SC21-7787-2
S5280-28

IBM 5280 Distributed Data System

DE/RPG Reference Manual

Program Number 5708-DE1

SC21-7787-2
S5280-28

IBM 5280 Distributed Data System

DE/RPG Reference Manual

Program Number 5708-DE1

Third Edition (June 1981)

This is a major revision of, and obsoletes, SC21-7787-1 and incorporates SN21-8196.

Because the changes and additions are extensive, this publication should be reviewed in its entirety.

This edition applies to release 3, modifications 0 of the IBM 5280 DE/RPG (Program 5708-DE1), and to all subsequent releases and modifications until otherwise indicated in new editions and technical newsletters. This publication contains examples of coded statements and the resulting operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual enterprise is entirely coincidental.

Use this publication only for the purpose stated in the *Preface*.

Changes are periodically made to the information herein; these changes will be reported in technical newsletters or in new editions of this publication. Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to annnounce such IBM products, programming, or services in your country.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Product Information Development, Dept. 997, 11400 Burnet Road, Austin, Texas 78758. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This publication describes the coded statements and the operating characteristics of the DE/RPG program for the IBM 5280 System. It includes both introductory and reference material for persons who program the system or design data entry applications. You should be familiar with the concepts involved in using DE/RPG programs.

This publication contains the following sections:

- Chapter 1 contains a general description of the DE/RPG program. The relationships among the source statements, the compiler, and the object program are described. This chapter also includes a comparison of the various programming strategies that can be used with DE/RPG and an example of coding for a simple data-entry job.
- Chapter 2 describes the required organization of statements within the source program. This chapter also contains information about the source entry program, which aids in the entry of source statements.
- Chapter 3 contains information about how to code source statements in DE/RPG.
- Chapter 4 explains how to code statements on the Z-specifications. The information is organized by statement type, and every entry is described.
- Chapter 5 explains how to code statements on the A-specifications. The information is organized by statement type, and every entry is described.
- Chapter 6 explains how to code statements on the C-specifications. All the entries for calculation statements are described.
- Chapter 7 contains introductory and detailed information about compile-time tables and self-check operations. Coding information for the tables and for self-check algorithms is included.
- Chapter 8 contains information about the DE/RPG compiler, including the data sets required for a compilation and the options available to the user.
- Chapter 9 describes the operating characteristics of a DE/RPG program. These characteristics include start-up and execution of the program. The modes of operation, key-initiated functions, and the status line are described. This chapter also describes production statistics for DE/RPG programs.
- Chapter 10 describes formats for zoned decimal data, packed decimal data, and binary data.
- The appendixes contain sequence and translation tables, brief summaries of data set characteristics and printer uses for DE/RPG, a listing of the compiler error messages, and the A-, Z-, and C-specifications forms.

DE/RPG is a language that allows you to create your own program by describing the job with statements. As a reference manual, this publication is intended to describe the variety of entries that can be used in each statement type to describe your job rather than to describe a particular application or to suggest techniques.

Related Publications

- *IBM 5280 Introduction to DE/RPG*, SC21-7803
- *IBM 5280 DE/RPG User's Guide*, SC21-7804
- *IBM 5280 System Concepts*, GA21-9352
- *IBM 5280 Operator's Guide*, GA21-9364
- *IBM 5280 Communications Reference Manual*, SC34-0247
- *IBM 5280-3270 Emulation Reference Manual*, SC34-0384

DE/RPG Coding and Debugging Material

- *Data Description Specifications*, GX21-9362
- *General Utility Specifications* GX21-9361
- *RPG Calculation Specifications*, GX21-9093

Contents

Chapter 1. Contents

Chapter 1. Introduction

INTRODUCTION

The IBM DE/RPG program product (DE/RPG) allows you to design programs to control the IBM 5280 System by describing your application or job with coded statements. These statements, called source statements, are used as input data by the DE/RPG compiler to produce an object program, which is written to diskette as a data set. You can load the object program data set into a storage partition when you want to run your job.

Source Statements

The source statements that can be used to describe jobs are divided into the following statement types:

- Job specification – for naming the job and specifying certain job characteristics
- Entry format – for specifying the sequence of record entry (data-entry jobs only) or the sequence of subroutine use during job execution
- Review format – for specifying which characteristic of a record identifies the appropriate display format during verify and update operations (data-entry jobs only)
- File description – for naming a file, describing its characteristics, and assigning an I/O device for the file
- Record description – for naming a record and describing record attributes
- Field description – for describing the characteristics of a field and the placement of the field within a record
- Table description – for identifying a table used during program execution
- Literal – for specifying fixed data within a record
- Calculation – for building subroutines
- Comment – for placing non-compiled comments or explanations in the source

In addition to the preceding statement types, compile-time table definitions and self-check algorithm specifications can be included as part of the source.

Source Program

Collectively, the statements that describe a job are a source program, which must be written to a diskette data set before the program can be compiled. An IBM-supplied Source Entry Program is provided on the diskette containing the compiler. The Source Entry Program is designed expressly for the entry of source programs and the creation of source program data sets for use by the compiler. The Source Entry Program is described in Chapter 2.

During program compilation, the compiler reads and interprets the source program that describes your job. The compiler allocates data areas for the files, records, and fields you specified and builds an object program that includes exactly the characteristics specified in the source program.

Coding Strategies

With DE/RPG, you can choose any one of three coding strategies to fit your job.

- You can specify the use of the transaction file.
- You can specify the use of the transaction file and use calculation statements for supplemental control of I/O.
- You can control the flow of your program entirely with calculation statements.

Transaction File

The transaction file supports data entry. When you use the transaction file, IBM-supplied routines are automatically provided by the compiler. These routines provide all the I/O control for the creation and access of the data set you assign as the transaction file. Additional access of the data set cannot be specified via calculation statements. (The transaction file routines provided by the compiler also support various modes of operation during program execution so that the same program can be used for verifying and updating the records in the data set.)

Transaction File with Calculation Statements

When you use the transaction file, you can also use calculation statements to control any I/O device function except those controlled by the IBM-supplied routines provided to accommodate data entry. For example, you can use calculation statements to control formatted printing and access to any diskette data set other than the data set assigned as the transaction file. Calculation statements can also be used to display information and to collect input from the keyboard/display station by using formats that are not specified for transaction file records.

Calculation Statements Only

When you do not use the transaction file, all I/O operations must be specified with calculation statements.

CHARACTERISTICS OF JOBS USING THE TRANSACTION FILE

Jobs that use the transaction file are usually data-entry jobs – that is, jobs with the express purpose of operator-entry of data from source documents to produce a diskette data set containing the entered data.

Verify and Update Records

The records in the transaction data set (produced by the transaction file) can be verified and updated by the operator using the same program that was used to enter the data.

Key-Initiated Functions

A comprehensive set of key-initiated functions can be used during the execution of jobs that use the transaction file. These functions enhance the key-entry of data and also enable the operator to locate a specific record by content or relative record number, to copy records from another data set to the transaction data set, and to insert records into the transaction data set. See *Chapter 9. Operating Characteristics* for additional information about the functions and the modes of operation of programs.

Production Statistics

Production statistics are accumulated by the system for jobs that use the transaction file. These statistics are maintained in special counters. Statistics can be accessed for a job or for a station. See *Production Statistics* in Chapter 9 for additional information.

CODING A MINIMUM JOB WITH THE TRANSACTION FILE

A simple job for data entry requires only a few statements in the source program. When the job involves the entry of only one kind of record, the following statements are required.

1. Job specification statement – to specify job characteristics
2. Entry format statement – to name the record to be entered
3. File description statement for the keyboard/display
4. Record description statement – to specify the record the fields belong to
5. Field description statements – to describe the characteristics of the fields
6. File description statement for the transaction file

An example of the coded statements for a simple data-entry job is shown in Figure 1. The source document for data in this job is a page from a city telephone directory. The entries in the directory are (from left to right) name, address, and telephone number. Only the names and numbers are desired as data; the address is to be ignored.

The example contains the minimum number of statements. Each statement contains a minimum number of entries.

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A.

Job		Keying Instruction	Graphic								Description	Page of
Operator	Date		Key									

Z

Sequence	Form Type	Name Type	Format ID (1-9, A0-Z9)	Job/Format/ Subroutine Name	Reserved	Repeat (1-9, N)	Mode (E/R)	AND(A)	Test Conditions: Position to be Tested (*POSnnnn)	Reserved	Condition	Character to Test for ('C')	Reserved	Next Format ID (0-9, A0-Z9)	Reserved	Options: Job Line / Entry Lines
1	Z	J		A					*POS		EQ	'C'				TFILE(C)
2	Z	1		B			E		*POS		EQ	'C'				
3	Z								*POS		EQ	'C'				
4	Z								*POS		EQ	'C'				
5	Z								*POS		EQ	'C'				
6	Z								*POS		EQ	'C'				

Options — Job Line: CFILE (data set); DATE (+DMY/+YMD); EDITC [(cuptd)]; ENTRATR (attr...); EXITATR (attr...); JOBOPT [(+NOPMT][+NOOPEN)]; PRTFILE (data set); SHARE (name); SHARER (name); STATUS (name); TFILE (data set [delfreq]); attr=BL,CS,HI,ND,RI,UL. Entry Lines: CLRL (number); EOJ [(job dev [+PASS])]; SLNO (line); WRITE (name).

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing: Checks=CHECK (code...) / Functions
1	A					F	D			37							DEVICE(CRT)
2	A					R	B							I			
3	A									30							
4	A									7			0				
5	A					F	C			37							DEVICE(DISK)
6	A																
7	A																
8	A																
9	A																
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

Checks=CHECK (code...): Auto Dup -AD; Auto Skip -AS; Blank Check -BC; Bypass -BY; Bypass on Verify -BV; Data Required -DR; Dup Disable -DD; Field Exit Required -FE; Lower Case -LC; Mandatory Entry -ME; Mandatory Fill -MF; Rt Adj-Blank Fill -RB; Right to Left -RL; Rt Adj-Zero Fill -RZ; Self Check -nxx (n: C,G (Check/Gen); xx-Modulus).

Functions: ADD (name); AUXDUP (name); AUXST (name); COMP (¹test fld1²@ fldn [indicator]); DSPATR (²attr...); EDTCDE (code float); ERROR (code [message]); EXSR (subroutine); INSERT (fld1²@ fldn); LOOK (table [index]); PMT (prompt); RANGE (low high); RANGET (table [index]); RESET (+TOTn); SEQ (¹test); SETOF (ind); SETON (ind); SHIFT (⁴shift); SUB (name); SUBST (table1 table2 [index]); TADD (+TOTn); TSUB (+TOTn); XCHK (table index1 index2); 'literal'.

¹test=EQ,GE,GT,LE,LT,NE. ²attr=BL,CA,CS,HI,ND,RI,UL. ³@=+,-,*,/. ⁴shift=A,D,H,N,V,W,X,Y.

*Number of sheets per pad may vary slightly.

Figure 1. Data Entry Job Examples

The following paragraphs explain each of the coded statements used in the example.

The job specification statement is on the first line on the Z-specifications. The Z in column 6 and the J in column 7 are required by the compiler for job specification statements. The job name is *A*, which is brief, but adequate for a name. The job is to use the transaction file as is indicated by the keyword TFILE, and the name of the transaction file is *C*.

Z-Specifications

The second line of the Z-specifications contains an entry format statement, as is indicated by the Z in column 6 and the E in column 21. An ID is required for all entry format statements, and the ID entered in this case is *1* (column 8). A name is required in the name field of every entry format statement. The name, *B*, is the name used for the record that is to be entered. Column 20 is blank indicating that this entry format is repeated indefinitely. Because column 20 is blank and because there are no other entry formats described, columns 45 and 46 can be blank. This entry format statement contains no keywords in the options field. Therefore, the default values for the starting line number and the number of lines to be cleared are used, and the data is written to diskette as a default record. (Default records consist of all input fields placed end-to-end in the order of entry.) The keywords that can be used with each statement type are described in Chapters 4 through 6.

A-Specifications

The first statement on the A-specifications in the example is the file description statement for the keyboard/display. The A in column 6 and the F in column 17 identify the statement type for the compiler. The *D* in column 19 satisfies the requirement for a name for the file. The length field specifies that the length of the record is 37. The DEVICE keyword with the parameter (CRT) assigns the keyboard/display to the file. Because the DSPSIZ keyword is not used, the minimum size of a display that can be used for this job is the default value, or the 480 character display.

The second line on the A-specifications is the record description statement, which is identified for the compiler by the A in column 6 and the R in column 17. The record name, *B* in column 19, is the same name as the name in the entry format statement. The / in column 38 defines the default value for a blank in the usage field for the field description statements that follow.

The next two lines are field description statements. These statements are identified to the compiler by the *A* in column 6 and the combination of the blanks in column 17 and entries (30 and 7) in the length field of the coding sheet. Neither of these fields is named in columns 19 through 24 because the fields are not used in any calculations and because the record is written as a default record (not reformatted). The first field described is 30 positions long as specified in columns 30 through 34 and is for character data (because there is an entry in the length field and column 37 is blank). The second field is 7 positions long (7 in column 34) and is for numeric data with no positions to the right of the decimal point (0 in column 37).

Both fields are input fields because the I in column 38 of the record description statement is used as the default for the blanks in column 38 of the field description statements. The blanks in column 35 of both field description statements allows specific defaults for keyboard shift during data entry. The first field is described for character data, and the default shift is alphabetic shift. The second field is for numeric data, and the default for numeric fields is numeric shift. The absence of entries in the line and position field (columns 39 through 44) and the absence of the starting line number keyword in the associated entry format statement (the second statement on the Z-coding sheet) sets the default for displaying the first data to line 3 and position 1.

Data for the first field displays in positions 1 through 30, and the data for the second field displays in positions 31 through 37, next to the first field of data.

The last statement in the source program is the file description for the transaction file. This file is named C, which matches the name specified in the parameter for the TFILE keyword in the job specification statement. The length of the logical records is 37 positions (columns 33 and 34). The DEVICE keyword is required to assign a device to the file. The device for a transaction file must always be a diskette drive. Because the BLKING keyword is not specified, the default blocking factors are used, and the data is written to diskette blocked and spanned.

How to Expand the Source Program

The source program described in the preceding paragraphs can easily be expanded to reformat the records written to the resulting data set. When the records for the transaction file are reformatted, the data remains the same (the same fields), but the order of the data fields within the records can be changed.

In the preceding example, the records in the resulting data set are organized with the name of the telephone subscriber first and the telephone number second, which is the order of the entry of data into the fields for each record. To change the order of the data in the data set records to number and then name, only the additions to the source program shown in Figure 2 are required.

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A

Job		Keying Instruction	Graphic	Description	Page of
Operator	Date		Key		

Z

Sequence (1–5)	Form Type (6)	Name Type (7)	Format ID (1-9, A0-Z9) (8–9)	Job/Format/Subroutine Name (10–17)	Reserved (18–19)	Repeat (1-9, N) (20)	Mode (E/R) (21)	AND(A) (22)	Test Conditions: Position to be Tested (*POSnnnn) (23–30)	Reserved (31–32)	Condition (33–34)	Character to Test for ('C') (35–37)	Reserved (38–44)	Next Format ID (0-9, A0-Z9) (45–46)	Reserved (47–54)	Options (55–80)
1	Z	J		A					*POS		EQ	' '				TFILE(C)
2	Z		1	B			E		*POS		EQ	' '				WRITE(H)
3	Z								*POS		EQ	' '				
4	Z								*POS		EQ	' '				
5	Z								*POS		EQ	' '				

Options:

Job Line
- CFILE (data set)
- DATE (*DMY/*YMD)
- EDITC ([cuptd])
- ENTRATR (attr . . .)
- EXITATR (attr. . . .)
- JOBOPT ([*NOPMT][*NOOPEN])
- PRTFILE (data set)
- SHARE (names)
- SHARER (names)
- STATUS (name)
- TFILE (data set [delfreq])
- attr=BL,CS,HI,ND,RI UL

Entry Lines
- CLRL (number)
- EOJ [(job dev [*PASS])]
- SLNO (line)
- WRITE (name)

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic	Source Document	Page of
Operator	Date		Key		

A

Sequence (1–5)	Form Type (6)	Comment (*) (7)	Reserved (8)	Indicator (for CHECK (BY, BV) or ERROR) (9–10)	Reserved (11–16)	Name Type (F/K/R/T) (17)	Reserved (18)	Dataset/Record/Field/Table Name (19–26)	Reserved (27–29)	Length (30–34)	Data Type (35)	Reserved (36)	Decimal Positions (0-9) (37)	Usage (I/O/B/W) (38)	Location: Line (39–41)	Location: Pos (42–44)	Editing (45–80)
1	A					F		D		37							DEVICE(CRT)
2	A					R		B						I			
3	A							K		30							
4	A							L		7			0				
5	A					F		C		37							DEVICE(DISK)
6	A					R		H									
7	A							L									
8	A							K									
9	A																
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

Editing:

Checks=CHECK (code . . .)

Auto Dup	=AD	Mandatory Entry	=ME
Auto Skip	=AS	Mandatory Fill	=MF
Blank Check	=BC	Rt Adj–Blank Fill	=RB
Bypass	=BY	Right to Left	=RL
Bypass on Verify	=BV	Rt Adj–Zero Fill	=RZ
Data Required	=DR	Self Check	=nxx
Dup Disable	=DD	n=C/G (Check/Gen)	
Field Exit Required	=FE	xx=Modulus	
Lower Case	=LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low high)
AUXST (name)	RANGET (table [index])
COMP (¹test fld1 ³@ . . . fldn [indicator])	RESET (*TOTn)
DSPATR (²attr . . .)	SEQ (¹test)
EDTCDE (code float)	SETOF (ind)
ERROR (code ['message'])	SETON (ind)
EXSR (subroutine)	SHIFT (⁴shift)
INSERT (fld1 ³@ . . . fldn)	SUB (name)
LOOK (table [index])	SUBST (table1 table2 [index])
	TADD (*TOTn)
	TSUB (*TOTn)
	XCHK (table index1 index2)
	'literal'

[1] test=EQ,GE,GT,LE,LT,NE
[2] attr=BL,CA,CS,HI,ND,RI,UL
[3] @=+,–,*,/
[4] shift=A,D,H,N,V,W,X,Y

*Number of sheets per pad may vary slightly.

Figure 2. Additions to Data Entry Job Example

The added code is circled in the example. The significance of the additions is explained in the following paragraphs.

The WRITE keyword is added in the entry format statement (second line on the Z-specifications). The parameter (H) names a record format in the transaction file. The record name *H* must be the *reformat* format for the same data that is included in a default record (all input fields and only input fields, which are coded with I or B in the usage field of the field description statements for the file to which the keyboard/display is assigned).

Names are added to the field description statements (third and fourth lines on the A-specifications). The names are required now because the fields must be identifiable later for reformatting the record. The order of entry stays the same as before and the entered data is still displayed in the same locations on the display screen.

Three new statements are added to the source program. The first new statement is the record description statement naming the record in the transaction file *H*. Next, the fields are named in the order they are to occur in the record. (The fields do not need to be completely described here, because they need be described only once in a program.)

With this reformatted record, the first data entered, which is still the telephone subscriber's name, occupies positions 8 through 37 in the records in the data set. The telephone number is positions 1 through 7 in each record.

Running the Program

After the program is compiled, the operator can load the resulting object program into a partition. Then the operator is prompted to select an appropriate initial mode of operation. The operator is prompted for data set names and device identification for the devices specified in the file description statements.

At the completion of the job, the operator must end the job via the End of Job key.

Chapter 2. Contents

Chapter 2. Source Program Data Set

DE/RPG SPECIFICATIONS

The source program data set is the input data for the DE/RPG compiler. Therefore, the records in the source program data set must contain data in a correct format.

The formats required by the compiler are reflected in the organization and placement of fields in the coding sheets. Each line on the coding sheets contains 80 columns, which correspond directly to the 80 positions in records required by the compiler.

Figure 3 shows the three kinds of specifications that you can use to describe your program. The Z-specifications and the A-specifications are both divided into two sections. Entries in the left side of these specifications are significant by position. The right field in both is for free-form entry of keywords.

All the fields in the C-specifications are significant by position.

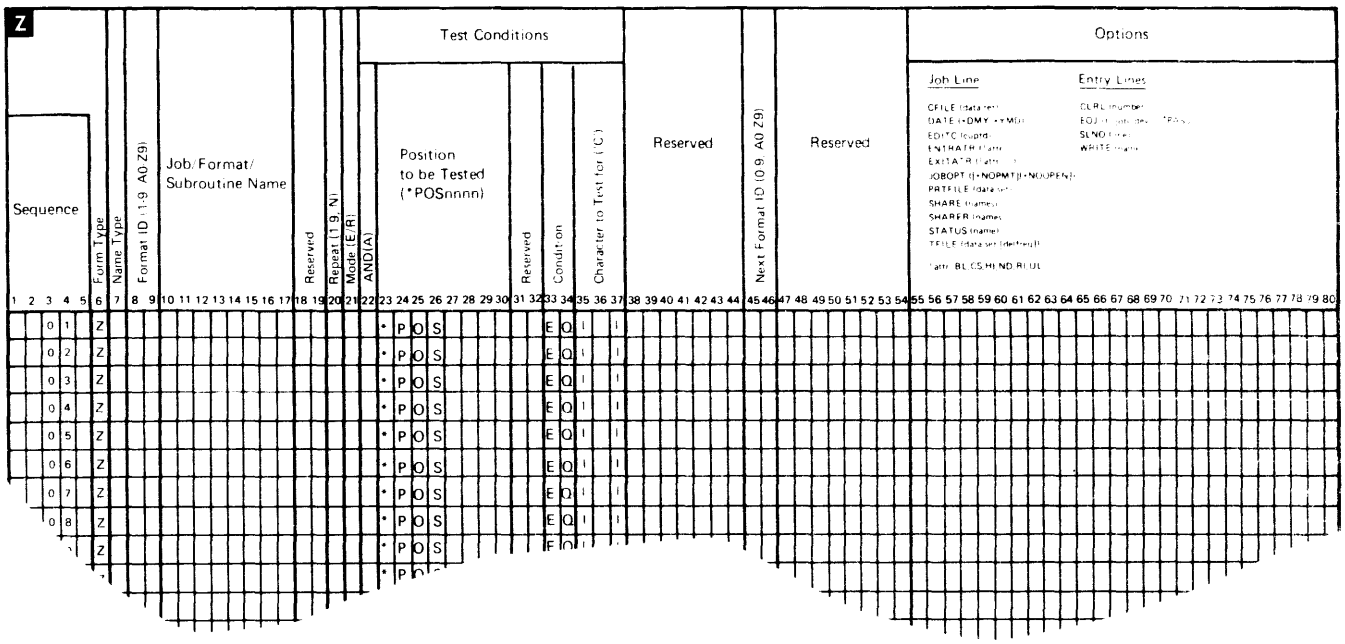
Z-Specifications

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

GX21-9361-0
Printed in U.S.A.

Job		Keying Instruction	Graphic								Description	Page of
Operator	Date		Key									



A-Specifications

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

GX21-9362-0
Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

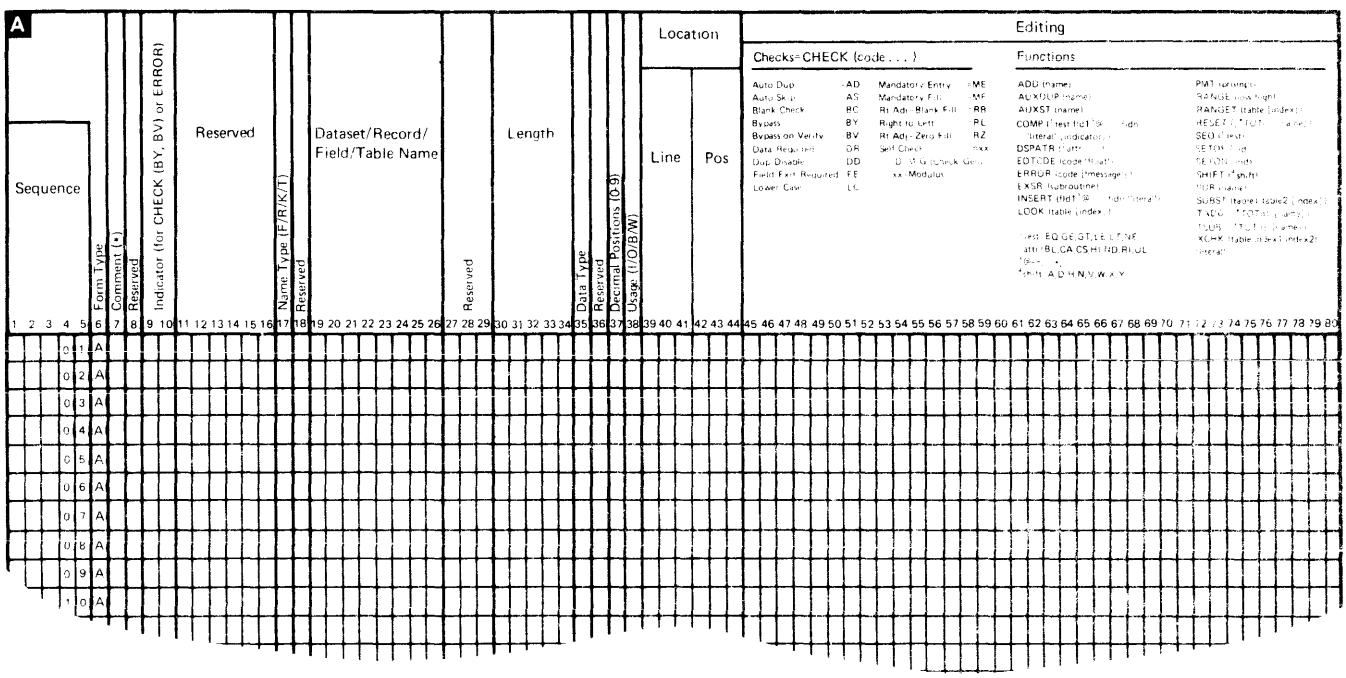


Figure 3 (Part 1 of 2). DE/RPG Specifications

C-Specifications

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 1 2 of __ Program Identification 75 76 77 78 79 80

C	Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
	3 4 5	6	7 8	9 10 11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61	62 63	64 65	66 67	68 69	70 71	72 73 74
	0 1	C																			
	0 2	C																			
	0 3	C																			
	0 4	C																			
	0 5	C																			
	0 6	C																			
	0 7	C																			
	0 8	C																			
	0 9	C																			
	1 0	C																			
	1 1	C																			
	1 2	C																			
	1 3	C																			
	1 4	C																			
	1 5	C																			
	1 6	C																			
	1 7	C																			
	1 8	C																			
	1 9	C																			
	2 0	C																			
		C																			
		C																			
		C																			
		C																			
		C																			

69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 72 71 70

Figure 3 (Part 2 of 2). DE/RPG Specifications

STATEMENT SEQUENCE

The statements in a source program data set must be in a sequence that is acceptable to the compiler program. The compiler reads and processes the source statements consecutively starting with the first statement. Comment statements are not processed by the compiler and can be specified ahead of (or following) any other source statement.

The following chart summarizes the required sequence of source statements and user-supplied definitions.

Sequence	Specifications Used	Statement or Definition
1	Z	Job specification statement
2	Z	Entry format and review format statements
3	A	File, record, field, and table description statements
4	C	Calculation statements
5	80-character record	Compile-time table definitions
6	80-character record	Self-check algorithm specifications

Z-Specifications

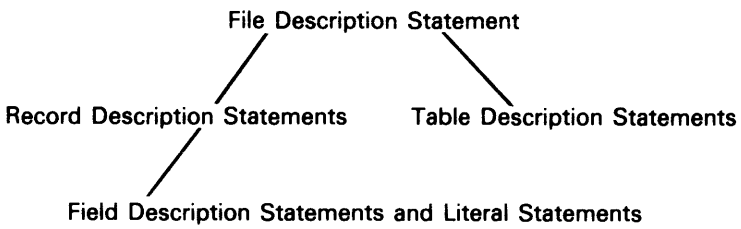
The first statement processed by the compiler must be a job specification statement. Every source program must contain one and only one job specification statement.

The remaining statements on the Z-specifications must follow the job specification statement. Every source program must contain at least one entry format statement. Review format statements (if used) can both precede and follow entry format statements because the entry format statements and the review format statements are processed separately by the compiler. However, review format statements are executed in the sequence in which they are specified.

A-Specifications

The statements on the A-specifications must follow the Z-specifications statements.

The statements on an A-specifications must be entered in the order represented by the hierarchy in the following chart, which shows file descriptions in the highest (first) position.



Field description statements and literal statements are always subordinate to a record description statement. Record description statements and table description statements are always subordinate to file description statements.

A file description statement must be the first A-specification statement processed. In general, a file description statement is followed by a record description statement, which in turn is followed by field description statements (and any literal statements) for the entire record. When the file contains two or more kinds of records, a record description statement and then the field description statements for that record should be added consecutively until all the record types for the file are described.

File description statements can be coded without any subordinate statements. For example, when the CFILE keyword is used in the job specification statement, a file description is required to assign a device (diskette drive) to the function.

C-Specifications

Statements on the C-specifications (if used) must follow the last statement on the A-specifications. Your programming needs determine the sequence of the calculation statements, but all calculation statements for executable operations must occur between a subroutine entry (BEGSR operation code) and a subroutine exit (ENDSR operation code).

Definitions of compile-time tables and self-check algorithm definitions (if used) must follow the last statement on the C-specifications (if used). The compile-time tables and self-check algorithms can be coded on any specifications or general purpose coding sheet, but not more than 80 columns can be coded on a line.

SOURCE ENTRY PROGRAM

The Source Entry Program is supplied by IBM and is contained on the diskette that contains the DE/RPG compiler. You can use the Source Entry Program to enter source statements into a source program data set. You can also use the Source Entry Program to update or to verify previously entered source statements.

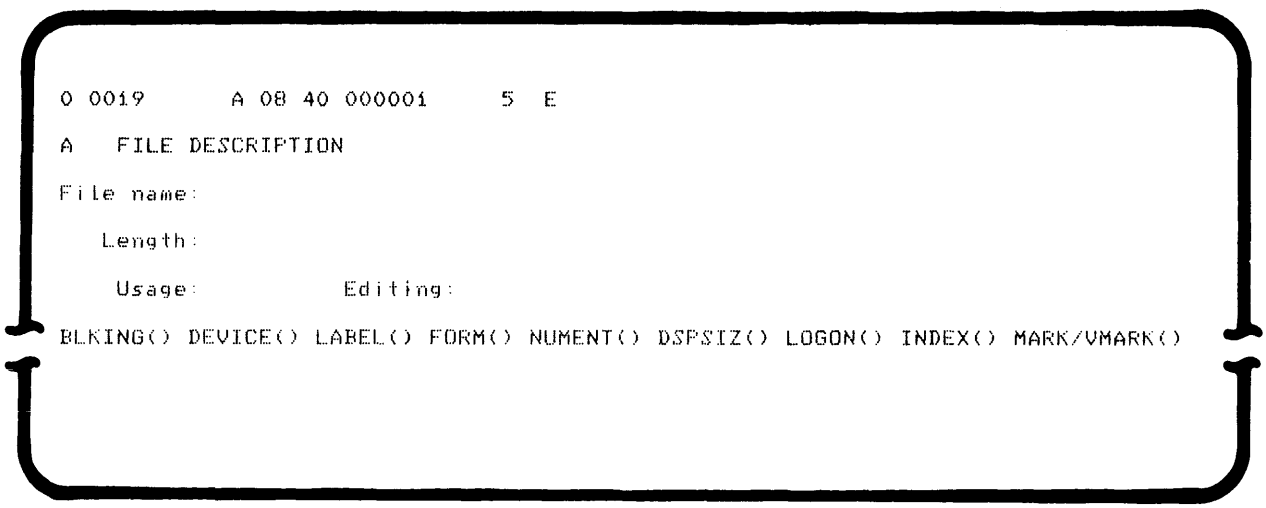
The Source Entry Program features 17 different formats and a menu. Each format is designed to help you enter a particular source statement type or a continuation line associated with a source statement type. The menu displays the names of the formats and simplifies the process of selecting a format.

The Source Entry Program is a DE/RPG program. The file through which you enter source statements is defined in the program as a transaction file. Thus all the modes of operation that are valid for transaction files are available through the Source Entry Program. The section *Operating Modes* in Chapter 9 describes these modes of operation. The *Operator's Guide* describes how to perform operations that are valid in each mode.

The Source Entry Program requires a minimum partition size of 13K. The partition size needed increases as the size of the output diskette increases. A partition size of 16K is required for a sector size of 1024. Increased partition size is required if the Print key or Review Second Data Set key is used.

SOURCE ENTRY PROGRAM FORMATS

The formats for each statement type display only the fields that are valid for that statement type. For example, the following format allows fields to be entered for a file description statement.



The current field is displayed in reverse image. You can either enter data into the field or you can skip or space over the field. As you enter each field, it is reformatted into the correct format for the compiler. Fields that must be right-adjusted are right-adjusted by the program. Also, any constant that is always required for a specific statement type is provided by the program. For example, when a job specification statement format is selected, the Z in position 6 and the J in position 7 of the record are automatically provided.

When a record has been entered, the Source Entry Program automatically displays the next appropriate format when a successive format is predictable. For example, a job specification statement is followed by the format for an entry format statement, and a record description statement is followed by a field description statement. The following chart shows the format IDs associated with each format. The chart also shows the format ID of the format that is automatically selected when the Enter key is pressed or when the Next Fmt (next format) key is pressed.

Format		After Enter	After Next Fmt
1.	Menu	N/A	N/A
2.	Job Specification	C3	C3
3.	Entry Format	3	4
C3.	Options continued	C3	3
4.	Review Format	4	1
5.	File Description	5	6
C5.	File Editing continued	C5	5
6.	Record Description	7	7
C6.	Record Editing continued	7	7
7.	Field Description	7	C7
C7.	Field Description continued	C7	7
8.	Comment	8	1
9	Calculation	9	1
0	Format 0 (compile-time data)	0	0
T0	Table Description	T0	1
S0	Lower Case Input (format 0)	S0	1
R8	Review Comments	R8	1

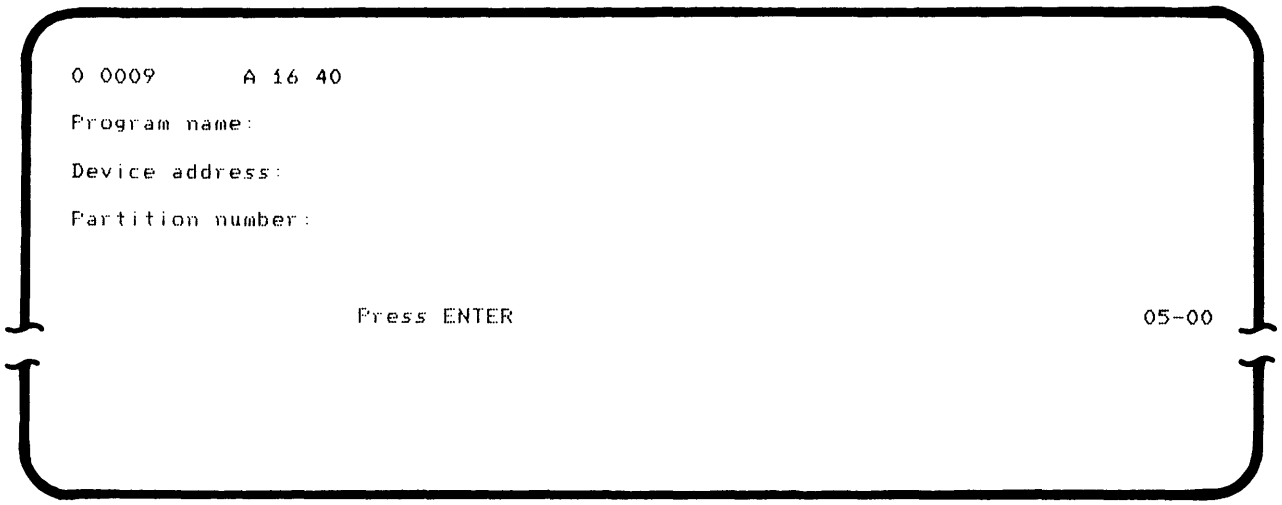
Note: Enter literal statements by using format 7 (field description) and format C7 (field description continued). Enter compile-time table definitions and data and self-check algorithm definitions by using format 0.

Regardless of the predetermined format succession, you can select any format or the menu at any time by using the Sel Fmt (select format) key.

The displays associated with the format IDs are shown in *Appendix E.*

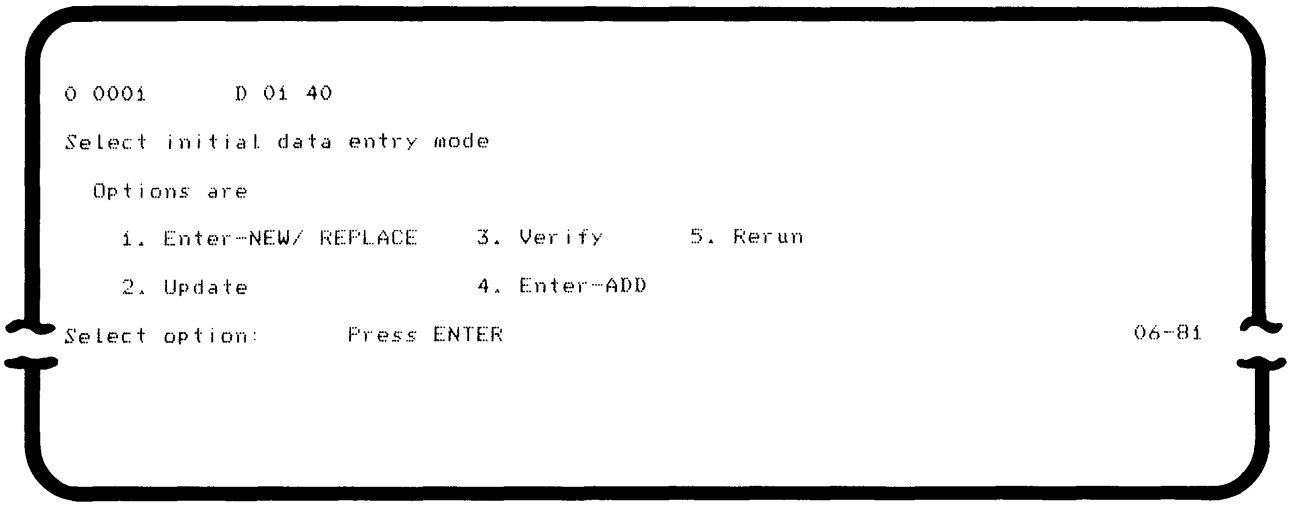
USING THE SOURCE ENTRY PROGRAM

Load the Source Entry Program by inserting the diskette containing the program into a diskette drive and responding to the load prompt.



The name of the Source Entry Program on the diskette with the compiler is SYSSEP.

When the program is loaded, the system prompts you to select an initial data entry mode.



Enter a 1 to write new source to an empty data set or to write over existing source in a data set. (All existing source will be lost when the new data is written.)

Enter a 2 to review and change previously entered source.

Enter a 3 to verify source for accuracy and to make corrections if necessary.

Enter a 4 to add source at the end of previously entered source in a data set.

Enter a 5 to review the source in rerun-display mode. (Normally, this option is not used when you are entering source programs.)

Next, the system prompts for file information about the source program data set.

```
0 0001      A 26 E2                E

Enter data for data set open

Data set name:     SYSIN

Device address:    4000


                    Press ENTER                                    06-82
```

Ensure that the diskette for the source program data set is inserted, then respond to the prompting message. The name you assign to the data set in response to this prompt will be the name of the source program data set.

If you selected option 1 (Enter (NEW/REPLACE)) on the previous display (screen 06-81) and the data set entered in 06-82 does not exist on the diskette, the program prompts you for the information needed to allocate the data set. If the diskette is volume protected, the program prompts you for the owner identifier.

Note: The record length of the source program data set must be 80 because DE/RPG source statements are 80 positions long.

If you selected enter mode on the initial data entry mode prompt, the first format displayed is the menu of formats for the entry of source statements.

```
0 0001      N 01 40            1  E
SELECT FORMAT: _  Your entry goes here.

  1 MENU                 5 FILE   DESCRIPTION   8 COMMENT

  2 JOB SPECIFICATION    6 RECORD DESCRIPTION   9 CALCULATION

  3 ENTRY FORMAT         7 FIELD  DESCRIPTION   0 FMT O FOR RECORD IMAGE

  4 REVIEW FORMAT        T TABLE  DESCRIPTION   S SHIFT LOWER CASE (FMT SO)
```

You can select a format from the menu, or you can use the Select Fmt key to select the format you want to use. If you are entering a new source program, the first format you should select is format 2 because the job specification statement must be the first statement in a source program. If you are updating or adding source statements, you can select whichever format is appropriate.

As you enter source statements (records) into the data set, the Source Entry Program inserts a deleted record after every five records that you enter. This allows space in the data set for records to be added later.

Line Numbering

When a source program is compiled, the DE/RPG compiler assigns line numbers to the source statements. These line numbers are shown on the compiler listing. When the compiler assigns these numbers, a continued statement is assigned the same number as the associated primary statement (even though each statement is a separate record in the source program data set). Because of the line numbering and the insertion of deleted records, the record numbers of the statements in the data set and the line numbers on the compiler listing will often be different.

Keyboard Functions

While the Source Entry Program is executing, all the keyboard functions that are valid for use with a transaction file are supported. These functions include:

- Search functions
- Second data set functions
- Record insert and delete functions
- Erase input functions
- Print function
- Page forward function

The search functions are useful when you are updating a source program to correct errors in the source statements. Although the line numbers on the compiler listing and the record numbers are not identical, the search functions can usually be used to find a record that is close to the desired source statement.

The second data set functions can be used to copy a group of source statements from an existing data set into the new data set when enter mode is being used. This can be helpful when more than one program use the same compile-time data tables or self-check algorithms.

Disable Auto Enter Function

Although the auto enter function is supported for entering source statements, this function should normally be disabled during source entry. This allows you to review each completed statement before it is written to the diskette. Also, when the auto enter function is disabled, you can correct an overrun in the keyword portion of the statement, as follows: You can reset the awaiting record advance condition (the cursor is flashing in the last record position) by pressing the character backspace (←) key, then you can enter a continuation character in the last position to allow a continuation line to be entered.

Chapter 3. Contents

Chapter 3. Coding Source Statements in DE/RPG

You can create a simple program to control your data entry process by coding directly with the Source Entry Program. That is, you can create the program by simply responding to prompting messages. When a more complex program is needed, the use of the A-, Z-, and the C-specifications is recommended.

This section describes the entries that you can make when you fill out the specifications or in response to the prompting messages that are displayed when you use the Source Entry Program. Because the prompting messages are easily related to the field heading on the specifications, the specifications are referred to during the statement descriptions.

PRIMARY, CONTINUATION, AND SECONDARY LINES

With the A- and Z-specifications, each source statement can usually be completed in a single line, called a primary line. Because of the nature of some applications, some source statements require more than one line. Two additional line types can be used as supplements to the primary line when additional coding space is needed. These supplemental line types are continuation and secondary lines.

- Continuation lines, in effect, extend the *Options* field (Z-specifications) or the *Editing* field (A-specifications).

- Secondary lines allow the specification of additional test conditions and alternatives.

The continuation lines and the secondary lines are coded by eliminating entries in certain fields that must be coded for primary lines. During program compilation, a continuation line must follow its associated primary line. Secondary lines must follow the primary line (or follow the continuation line, if a continuation line supplements the primary line). Only comment statements, which are ignored by the compiler, can be coded between a primary line of a source statement and associated continuation or secondary lines.

Continuation Characters

Continuation characters are required whenever an expression such as a keyword and its associated parameter cannot be started and completed on one line. In such a case the line preceding a continuation line must contain a continuation character, either a plus character (+) or a minus character (-), as the last nonblank character in the line. A plus character indicates that blanks preceding the first nonblank character in the following continuation line are to be ignored during compilation. A minus character indicates that any blanks preceding the first nonblank character are to be included as part of the statement. All blanks preceding the continuation character (+ or -) are included.

GENERAL CODING CONVENTIONS

Several conventions for coding with DE/RPG remain consistent regardless of the type of coding sheet or statement type. These common items include the rules and syntax requirements for:

- *Names*, which are labels you can assign to the job, files, records, fields, tables, and variables
- *Reserved words*, which are names that have special meanings in the system
- *Sequence numbers*, which assist in keeping statements in the proper order for compilation
- *Keywords and their parameters*, which identify specific attributes and functions
- *Constants*, which can be coded as fixed data supplied by the program

Names

When you name a job, file, record, or field, you can use any combination of allowed characters.

Names are always left-adjusted. The first character must be selected from the following: A through Z, $, #, or @. The remaining characters may be A through Z, 0 through 9, $, #, or @. (Embedded blanks are not allowed in names.)

Job names, file names, and record names can be from one to eight characters long. All other names consist of a maximum of six characters.

Named variables that are used in a program need to be defined only once regardless of the number of times they are used in the program. If a named variable is defined more than once, the attributes of the variable (such as the length and decimal positions) must always be defined the same way.

Reserved Words

Table names cannot begin with TAB because these characters at the beginning of a table name have a special meaning to the system. In addition, the following names have special meanings also:

- *TOT1 through *TOT9 are fields meant to be used as counters. The fields are 15 positions long with no decimal positions.
- *STAT01 through *STAT29 are fields used as counters to accumulate job and station statistics. The lengths of these fields vary. They are described in Chapter 9.
- UDATE is a system variable that contains the date. The field is six positions long with no decimal positions.

These names do not need to be defined in a program. However, if they are defined, the length and decimal positions specified must be the same as the values used by the system. The fields *TOT1 through *TOT9 can be used as a source of data or a destination. Avoid the use of the other names except as a source of data.

Sequence Numbers

Sequence numbers in source statements are optional and are not checked by the compiler. However, sequence numbers can be useful when you want to change, add, delete, or relocate a source statement. The only restriction is that the first two columns of the *Sequence* field cannot contain **.

Keywords

Keywords represent attributes and functions. Most keywords must be accompanied by a string of one or more parameters that further define the attributes or function. The string of parameters must follow the keyword and must be enclosed within a pair of parentheses.

The general syntax for a keyword and its parameter string is:

KEYWORD (*parameter string*)

When the parameter string contains two or more parameters, each consecutive parameter must be separated from the preceding parameter by at least one blank.

Two or more keywords can be specified on a single line. Any keyword that is specified without a parameter string must be separated from the following keyword by at least one blank.

Within these guidelines, the following three lines convey identical meanings to the compiler.

```
EOJ SLNO(3) CLRL(*NO)

EOJ SLNO(3)CLRL(*NO)

EOJ  SLNO ( 3 ) CLRL ( *NO )
```

The only required space or blank in the sequence shown is between EOJ and SLNO.

Whenever a keyword and its parameter are not started and completed on the same line, the use of a continuation character is required. See *Continuation Characters*, earlier in this chapter.

Constants

Numeric constants consist of any combination of the numbers 0 through 9. A decimal point and a sign (+ or -) can be included. The sign, if used, must precede the first number. If a sign is not used, the value of the constant is positive. A decimal point in a numeric constant indicates place values and is used in decimal aligned arithmetic. However, the decimal point is not included as part of the data. No more than nine decimal places can be specified for numeric values. Numeric constants cannot contain blanks and cannot be enclosed in apostrophes.

Character constants consist of any combination of characters, including blanks. Character constants must be enclosed in apostrophes. An apostrophe required as data within the constant must be represented by two apostrophes. Character constants are invalid in arithmetic operations.

COMMENT STATEMENTS

Comment statements are not compiled, but allow you to include explanations for source program documentation.

Comment statements can be placed anywhere in the source statement sequence.

Signify a comment by entering an asterisk (*) in column 7. The remaining positions of the line (columns 8 through 80) are then ignored by the compiler and are available for comments.

Comment statements cannot be specified within the specifications for compile-time tables or within the definition of a self-check algorithm.

STATEMENT DESCRIPTIONS

The next three chapters describe the statements that you use to code a program. These chapters describe the contents and meaning of each field in the statements. At the end of each section, a short example is included that shows how the statement is coded on the specifications or the source entry prompt.

Chapter 4. Contents

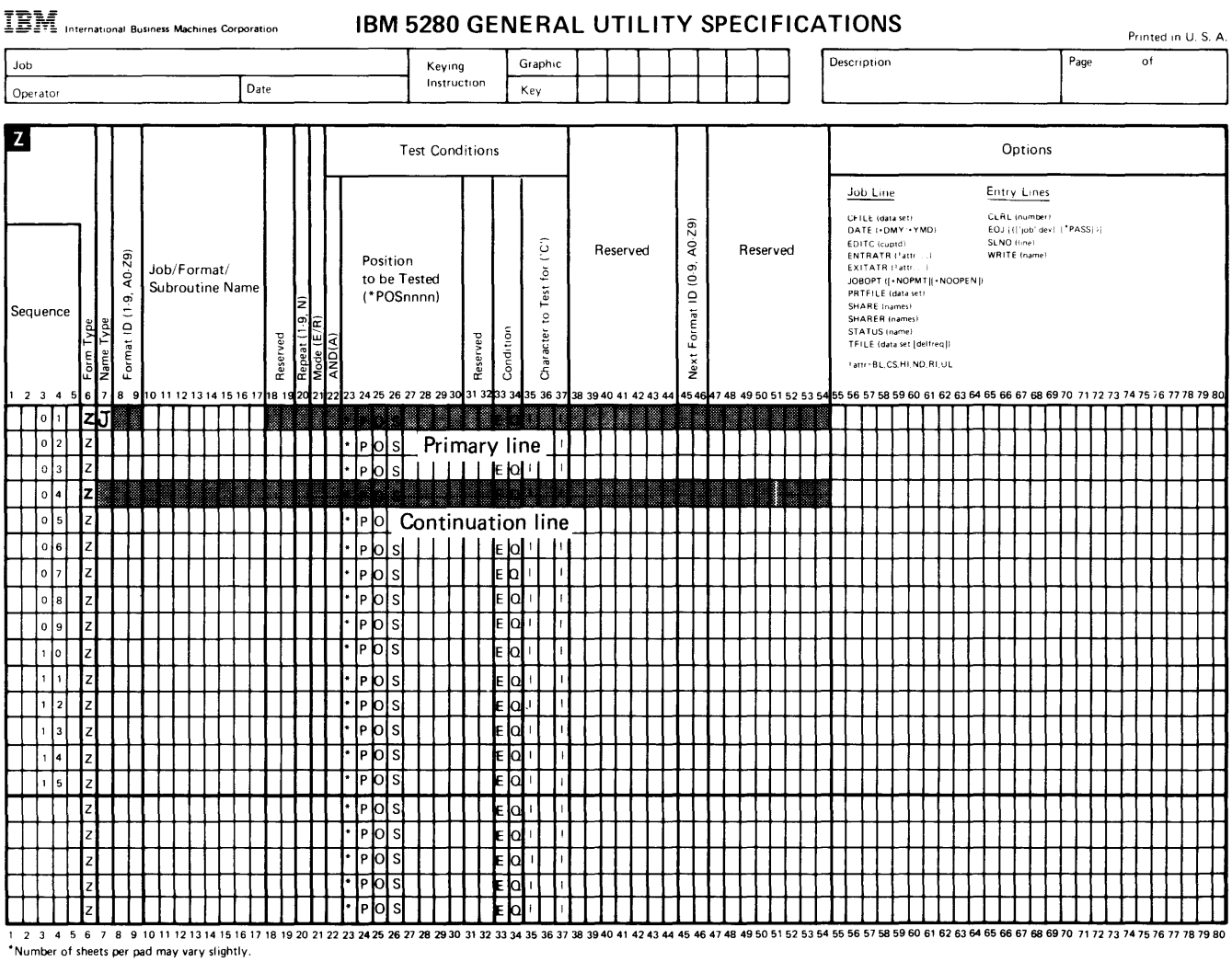
The Z-specification is used to code job specification statements, entry format statements, and review format statements.

JOB SPECIFICATION STATEMENT

The job specification statement names the job and specifies file-related functions that are to be used during the execution of the program. The job specification statement can also specify the date format and currency symbols to be used for the job.

During compilation, the job specification statement must be the first source statement in the sequence. The job specification statement can be preceded by comments, which are ignored by the compiler.

The fields that can be coded for job specification statements are shown as unshaded areas in the following illustration:



Sequence Number Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

Enter a Z in the *Form Type* field.

Name Type Field (Column 7)

Enter the letter J in the *Name Type* field.

Name Field (Columns 8 through 17)

The job name must be entered in the *Name* field starting in position 10. The job name is displayed on the status line when error conditions are indicated during program execution.

Options Field (Columns 55 through 80)

Only the following keywords and their associated parameters can be entered in the *Options* field. These keywords can be entered in any order.

- CFILE – copy file name
- DATE – date format
- EDITC – currency edit
- ENTRATR – input field attribute during entry
- EXITATR – input field attribute after entry
- JOBOPT – job options
- PRTFILE – print file name
- SHARE and SHARER – shared file names
- STATUS – I/O status
- TFILE – transaction file name

The *Options* field can be extended by continuation lines if additional coding space is needed on the specification line. To code a continuation line, use only the *Form Type* field and the *Options* field.

CFILE(*name*)

The CFILE keyword is required in the job specification statement if the copy function is to be used during program execution. The name parameter must be the name of the file from which the data is to be copied. During program execution, the operator is prompted for data set and device information so that the name of the execution-time data set can be different, if necessary, from the name used in the source program.

The keyword CFILE cannot be used unless the keyword TFILE is used. The length of the logical records in the data set for the transaction file and the data set for the copy file must be the same.

DATE(*DMY)
DATE(*YMD)
DATE(*MDY)

The DATE keyword is used to change the format of the contents of the system variable named UDATE. The default format for the date if the keyword is not used is MMDDYY (month, day, year).

The required parameter must be either *DMY, *YMD, or *MDY, which represent the formats DDMMYY (day, month, year), YYMMDD (year, month, day), and MMDDYY (month, day, year) respectively.

EDITC(*characters*)

The EDITC keyword is used to change edit characters for currency fields and the date. The edit characters are used in conjunction with the EDTCDE keyword.

The characters parameter is five positions long and must be enclosed in apostrophes. The first and second positions specify the currency symbol to be used. The third position specifies the decimal point character; the fourth position specifies the thousands separator character; and the fifth position specifies the date separator.

The default values of these positions are:

blank $	for currency symbol
.	for decimal point
,	for thousands separator
/	for date separator

If the defaults are acceptable, the EDITC keyword can be omitted. The date separator can be specified only as either / (slash) or . (period).

ENTRATR(*attributes*)

The ENTRATR keyword allows you to specify display attributes to be applied to each input field only during the entry of data into the field. Combinations of attributes can be specified in the parameter. The use of this keyword overrides any display attributes specified in field description statements after the cursor reaches the field and requires a blank display position ahead of and a blank display position immediately following the data displayed for each input field. When the entry of data for the field is completed, the display of the field is either returned to normal or changed to the attribute specified as the parameter for the EXITATR keyword (if used).

The following attributes can be specified.

BL – The displayed field blinks (flashes on and off).

CS – Column separators display between character positions. Column separators are fine vertical lines, which are between character positions and do not reduce display capacity.

HI – The displayed characters are highlighted (displayed with increased intensity).

ND – The field is not displayed, but data can be entered.

RI – The field is displayed with images reversed (dark characters are displayed on a light background).

UL – Each character position in the field is underlined.

The combination of HI, RI, and UL is invalid.

EXITATR(*attributes*)

The EXITATR keyword allows you to specify display attributes to be applied to each input field after the entry of data into each field is completed. In addition, this attribute is placed after the current field when the cursor enters the field; therefore, the attribute applies to the space following the field until the next attribute is encountered. The use of this keyword is valid only if the ENTRATR keyword is used. The attributes that can be specified are the same as the attributes available for ENTRATR. (Unless your display requires unusual attributes, there is no need to specify this keyword. The normal display is the default. Any other attributes might cause undesirable results on the display.)

Depending on the use of the DSPATR, ENTRATR, and EXITATR keywords, an input field could appear three different ways on the display, according to the location of the cursor. The effect of these keywords is:

Keyword	Where Specified	When in Control
DSPATR	Field line	From the initial display until the cursor reaches the field.
ENTRATR	Job line	While the cursor is in the field.
EXITATR	Job line	After the cursor leaves the field (moving forward *or* backward).

JOBOPT(*NOPMT)
JOBOPT(*NOOPEN)
JOBOPT(*NOPMT *NOOPEN)

At the start of execution of a DE/RPG program, the system normally displays the data set names and device assignments for each file to the operator and allows the operator to change them before the program executes. If you want to bypass these prompts, use the keyword JOBOPT with the parameter *NOPMT. When the prompts are bypassed, the system attempts to use the data set names and device IDs specified in the source statements during program execution without prompting the operator.

At the start of execution of a DE/RPG program, all files used by the program except communications files are normally opened. If you want to bypass opening the files until they are required in the program, use the keyword JOBOPT with the parameter *NOOPEN. If this is specified, an OPEN operation must be performed in a calculation subroutine before the file is accessed. In any case, transaction files are always opened when the program is started.

PRTFILE(*name*)

The PRTFILE keyword is required in the job specification statement if the key-initiated print function is to be used during program execution. The name parameter identifies the file. This file requires a file description statement to identify the printer.

The PRTFILE keyword cannot be used unless the TFILE keyword is used. The length of the logical records in the data set for the transaction file and the printer file must be the same. PRTFILE is not required when formatted printing is initiated with calculation statements and if the key-initiated print function is not to be used. Use of the PRTFILE keyword does not exclude formatted printing. However, the PRTFILE keyword and the calculation operation cannot refer to the same file.

SHARE(*name*)
SHARER(*name*)

The SHARE keyword is used to identify the files that can be accessed by other programs during the execution of this program. The *name* parameters must be file names. Files named as parameters for SHARE can be accessed by other programs for read, write, and update operations.

The keyword SHARER is the same as SHARE except that SHARER limits access by other programs to reading only and does not allow write or update operations.

When you specify SHARE or SHARER for a transaction file, you cannot insert into that file; for a keyed file, you cannot write to that file.

While the current program is active, files that are not named as parameters for the keyword SHARE or the keyword SHARER can be accessed only by the current program.

STATUS(*name*)

The STATUS keyword defines a variable that can be checked to determine the status of an I/O device after an I/O operation in error. If the STATUS keyword is specified, the I/O operation must be specified in a calculation subroutine. A resulting indicator must be specified in the calculation statement.

The name parameter implicitly defines a numeric field with a length of 4 and no decimal positions. After an error, the status codes returned in the field are the four-digit message identifiers, which are explained in the *System Message Manual*.

TFILE(*name*)
TFILE(*name number*)

This keyword is required for most data entry jobs to specify the use of the transaction file. The *name* parameter must be the name of a file that is described as a diskette file in a file description statement.

The second parameter, *number*, is optional and represents the number of data records to be written to the data set (during data-entry operations) between deleted records. When the specified number of data records has been written to the data set, a record that is flagged as a deleted record is written. This record is then followed by the specified number of data records, which is followed by another record that is flagged as a deleted record, and so on. The presence of the deleted records in a data set is beneficial if records are later inserted between existing records in the data set. The expansion of the data set to make room for the inserted records is absorbed by the deleted records, reducing the number of existing records that must be rewritten. Deleted records cannot be retrieved and are ignored during processing of the data set.

Note: Deleted records are only inserted after the specified number of records has been entered consecutively from the keyboard. If any records are deleted, updated, or copied by the operator before the specified number is satisfied, the counting of entered records is restarted and the number of data records between deleted records will vary.

When the TFILE keyword is used, a number of operator controlled data entry functions are automatically available to the program via the common function options SYSCFA and SYSHELP. These options are specified during system configuration, as described in the *System Control Programming Reference Manual*. The data entry functions provided are:

Auto duplicate	Record backspace
Auto enter	Record delete
Auto skip	Record insert
Clear screen	Rerun mode
Display verify record	Search content
Duplicate	Search record number
Edit release	Search sequential content
Enter mode	Search to end-of-data
Mark field	Select format
Next format	Update mode
Page forward	Verify mode
Record advance	

The use of these functions is described in Chapter 9 of this manual.

Example

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A.

Job		Keying Instruction	Graphic	Description	Page of
Operator	Date		Key		

Z

Sequence (1–5) | Form Type (6) | Name Type (7) | Format ID (1-9, A0-Z9) (8–9) | Job/Format/Subroutine Name (10–17) | Reserved (18–19) | Repeat (1-9, N) (20) | Mode (E/R) (21) | AND(A) (22) | Test Conditions: Position to be Tested (*POSnnnn) (23–30), Reserved (31–32), Condition (33–34), Character to Test for ('C') (35–37) | Reserved (38–44) | Next Format ID (0-9, A0-Z9) (45–46) | Reserved (47–54) | Options (55–80)

Options:

Job Line
CFILE (data set)
DATE (*DMY *YMD)
EDITC ([cupnd])
ENTRATR (attr ...)
EXITATR (attr ...)
JOBOPT ([*NOPMT][*NOOPEN])
PRTFILE (data set)
SHARE (names)
SHARER (names)
STATUS (name)
TFILE (data set [defined])
attr: BL,CS,HI,ND,RI,UL

Entry Lines
CLRL (number)
EOJ [(job dev [*PASS])]
SLNO (line)
WRITE (name)

Sequence	Form Type	Name Type	Job/Format/Subroutine Name	Position to be Tested	Condition	Character to Test for
1	Z			*POS	EQ	' '
2	Z	J	JOBNAME	*POS	EQ	' '
3	Z			*POS	EQ	' '
4	Z					
5	Z					
6	Z			*POS	EQ	' '
7	Z			*POS	EQ	' '
8	Z			*POS	EQ	' '
9	Z			*POS	EQ	' '
10	Z			*POS	EQ	' '
11	Z			*POS	EQ	' '
12	Z			*POS	EQ	' '
13	Z			*POS	EQ	' '
14	Z			*POS	EQ	' '
15	Z			*POS	EQ	' '
	Z			*POS	EQ	' '
	Z			*POS	EQ	' '
	Z			*POS	EQ	' '
	Z			*POS	EQ	' '
	Z			*POS	EQ	' '

This is all that is required to name the job, but if the job is to use the transaction file, the keyword TFILE must be used in the options field.

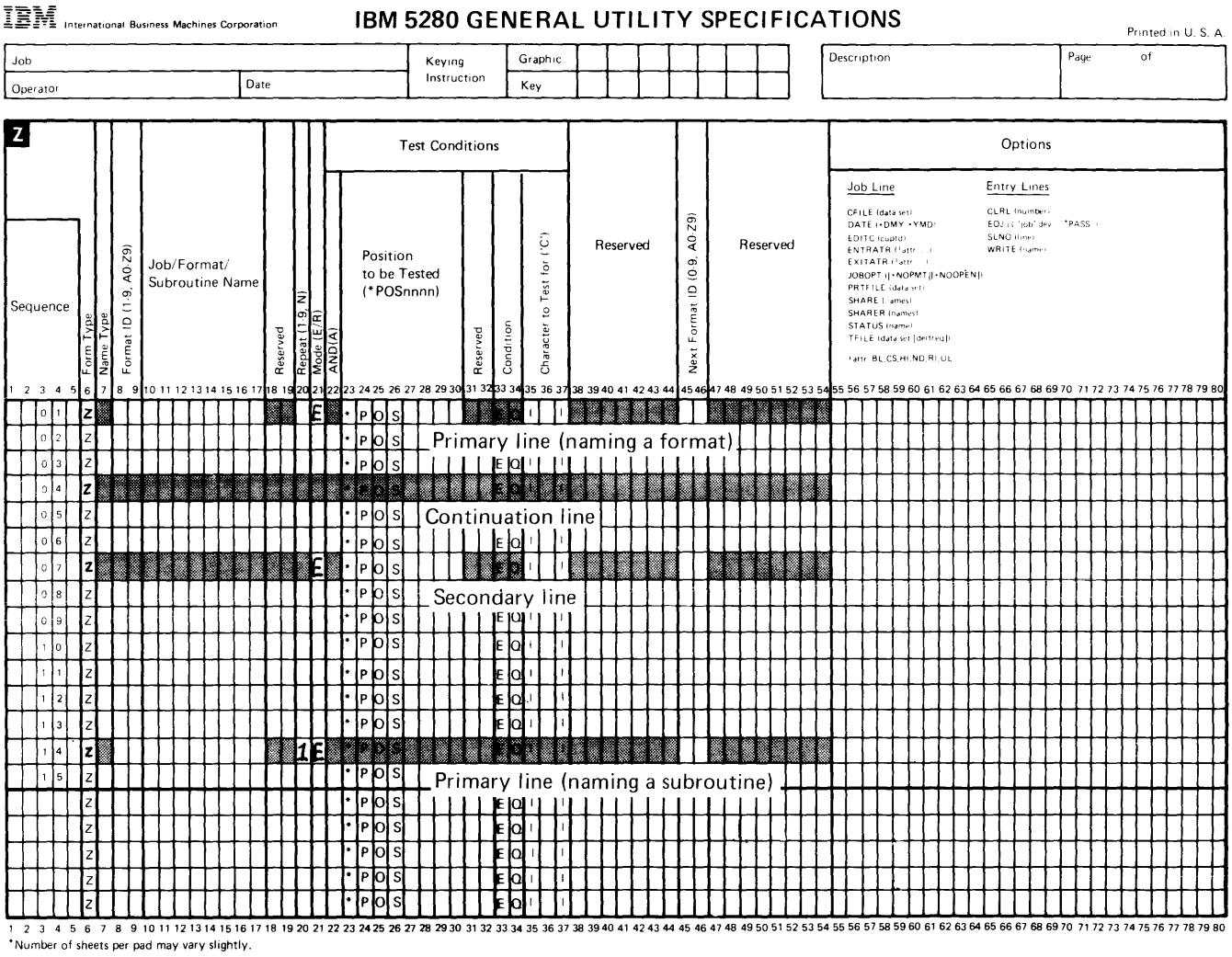
*Number of sheets per pad may vary slightly.

ENTRY FORMAT STATEMENTS

Entry Format statements identify record types for data-entry and can be used to assign the sequence of record entry or exits to calculation subroutines.

Entry format statements must be preceded by a job specification statement.

The fields that can be coded for entry format statements are shown as unshaded fields in the following illustration.



Sequence Number Field (Columns 1 through 5)

Enter the statement sequence numbers if desired.

Form Type Field (Column 6)

Enter a Z in the *Form Type* field.

Format ID Field (Columns 8 and 9)

An entry in the *Format ID* field is required in every entry format primary line. The ID of an entry format is used for both operator-controlled and program-controlled entry format selection. For data entry, the sequence of entry format use can be predetermined via the source statements but the operator can always modify the sequence during program execution through the use of the select format function and the next format function.

An ID can be either a one-character or a two-character entry and must be unique within the program. When a single character is used, the entry must be a number from 1 through 9. When a two-character ID is specified, the first character must be a letter (A through Z) and the second character must be a number (0 through 9).

The ID must start in column 8 for both one- and two-character IDs.

Name Field (Columns 10 through 17)

A name is required in the *Name* field of every entry format statement primary line. This name must be the name of a record that is described in a record description statement as part of the file assigned to the keyboard/display (CRT) or the name of a subroutine described on the C-specifications. Any program that uses a communications file (COMM or COMM3270) in which execution is controlled by calculation statements, must have at least one entry format statement that specifies a calculation subroutine. The name must start in column 10.

Mode Field (Column 21)

The letter E is required in the *Mode* field for entry format statements.

Repeat Field (Column 20), Test Conditions Fields (Columns 23 through 30 and 35 through 37), and Next Format ID Field (Columns 45 and 46)

The first record entry format or subroutine available during program execution is identified by the first entry format source statement processed during compilation. The order of use of entry formats during the remainder of a data-entry session is determined by one or more of the following three methods.

- Manual only - which requires the use of either the Sel Fmt (select format) key or the Next Fmt (next format) key on the keyboard to stop using the current format and to start using another.

- Automatic - which advances to the entry format identified by the *Next Format ID* field upon completion of the current format the number of times specified in the *Repeat* field or upon the use of the next format function.

- Conditionally automatic - which advances to the entry format identified by the *Next Format ID* field upon the equal comparison between a specified character and the data character in a specified position of the current record.

Entry format statements that name a calculation subroutine (in the *Name* field) must use the automatic method for advancing to the next entry format statement.

The entry format statement fields that affect the three methods of determining format sequencing are the *Repeat* field, the *Test Conditions* fields, and the *Next Format ID* field.

To specify the manual-only method of entry format sequencing, enter the letter *N* in the *Repeat* field or leave the *Repeat* field blank.

To code an automatic advance from one entry format to the next, enter into the *Repeat* field the number of times you want to use the format. Valid entries are any number from 1 through 9. Enter the ID of the entry format you want next in the *Next Format ID* field. Leave the *Test Conditions* fields blank. For entry format statements that name a calculation subroutine (in the *Name* field), the entry in the *Repeat* field must be 1.

For a conditionally automatic advance from one entry format to the next, you must use both the *Test Conditions* fields and the *Next Format ID* field. To test each record, enter the number 1 in the *Repeat* field. To suspend testing until a repeat count is satisfied, enter a number from 2 through 9 in the *Repeat* field.

The test is a comparison between the character, which is specified between apostrophes in the *Character to Test for* field, and the current record data character in the position specified in the *Position to be Tested* field. Specify the position in the form *POS*nnnn*, where *nnnn* is a number from 1 to 8192.

When the data in the specified position and the characters are the same, the entry format identified in the *Next Format ID* field is automatically selected to control entry of the next record. If the *Next Format ID* field is blank or if all the tests fail, the current format is repeated.

Multiple single-character tests can be specified, but the test conditions cannot be logically ANDed. All tests other than the first must be specified on secondary lines. Secondary lines for entry format statements must be blank in the *Format ID* field, the *Name* field, the *Repeat* field. The remaining fields can have entries as described previously.

When multiple tests are specified, the tests are attempted in the order in which they occur in the source program sequence. The first test found true stops the testing, and the entry format identified by the corresponding *Next Format ID* entry is used. If the test condition field is left blank and the *Next Format ID* field is specified, the specified format is selected if none of the previous conditions were satisfied.

Options Field (Columns 55 through 80)

Only the following keywords and their associated parameters can be entered in the *Options* field.

Keyword	Description
CLRL	Clear line
EOJ	End-of-job
SLNO	Starting line number
WRITE	Write in another format

CLRL(*number*)
CLRL(*NO)

The CLRL keyword allows you to limit the number of display lines that are cleared at the beginning of the current entry format.

The number parameter specifies the number of lines to be cleared, beginning with the starting line number specified by the SLNO keyword.

The *NO parameter retains the previous display, and any items displayed under the current format overlay the previous display.

If the keyword CLRL is not used, all of the lines from the previous display are cleared from the starting line to the bottom of the display.

EOJ
EOJ(*PASS)
EOJ('*name*' *device*)
EOJ('*name*' *device* *PASS)
EOJ(*name device*)
EOJ(*name device* *PASS)

The EOJ keyword can be specified with or without parameters. When EOJ is specified, the next format ID field must be left blank, and the end of the job essentially replaces a next format. EOJ can be entered in both primary and secondary lines of entry format statements. The EOJ keyword(s) is bypassed when the end-of-job key is used to end the job.

EOJ specified without parameters causes a normal end of job the same as the end-of-job key-initiated function.

EOJ(*PASS) ends the job but suppresses the option for production statistics.

EOJ('*name*' *device*) or EOJ (*name device*) chains to the next job, which is identified by the parameter string.

The first parameter, *name*, is the name of the object program data set as it appears on the diskette. This name is not necessarily the name entered in the *name* field of the job description statement. The name parameter can be a constant name (enclosed in apostrophes) or a variable name (not enclosed in apostrophes). The constant name parameter is specified at compile time, and the variable name parameter is specified at execution time. A variable name that begins with a blank causes a normal end of job.

The device parameter is required if the *name* parameter is specified. The device parameter can be specified in either of two forms:

- Two-character logical ID, which must be defined during system configuration
- Four-character physical device code for a diskette device, expressed as X'nnnn'

Refer to your system configuration records for the logical IDs and device codes that are valid on your system.

SLNO(*number*)

The number parameter for the keyword SLNO specifies the uppermost physical display line that the format can use. All display line references are based on this line as line 1. For example, if you specify SLNO(3) and assign a field to line 2 in a field description statement, the field is actually displayed on physical line 4.

The number parameter can be any whole number greater than or equal to 2 and not exceeding the number of lines on the display. (Physical line 1 is always reserved for displaying the status line.) If you plan to use the keyword PMT to display prompts, choose a value greater than or equal to 3, because prompting messages are displayed on line 2. If SLNO is not specified in an entry format statement, the default value of 2 is used.

WRITE(*NO)
WRITE(*name*)

The WRITE keyword can be used either to bypass writing the current record to the data set assigned as the transaction file or to specify a format other than the the record entry format for writing the data to the data set.

The *NO parameter bypasses the writing of the current data to diskette.

The *name* parameter must be the name of a record that is described in a record description statement for the transaction file.

When the keyword WRITE is not used, the current default record is written to the data set. The fields in the default record are in the sequence in which they were entered. All fields are adjacent with no spaces between fields. Only fields that are described as input fields (I or B in the *Usage* field in the field description statements) are included in the default record.

Example

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

GX21-9361-0
Printed in U. S. A.

Job		Keying Instruction	Graphic								Description	Page of
Operator	Date		Key									

Z

Sequence	Form Type	Name Type	Format ID (1-9, A0-Z9)	Job/Format/ Subroutine Name	Reserved	Repeat (1-9, N)	Mode (E/R)	AND(A)	Test Conditions: Position to be Tested (*POSnnnn)	Reserved	Condition	Character to Test for ('C')	Reserved	Next Format ID (0-9, A0-Z9)	Reserved	Options
01	Z								*POS		EQ	' '				
02	Z		A1	EXAMPLE1		N	E		*POS		EQ	' '		A2		
03	Z								*POS							
04	Z								*POS							
05	Z								*POS		EQ	' '				
06	Z								*POS		EQ	' '				
07	Z		A2	EXAMPLE2		5	E		*POS		EQ	' '		A3		
08	Z								*POS							
09	Z								*POS							
10	Z								*POS		EQ	' '				
11	Z								*POS		EQ	' '				
12	Z		A3	EXAMPLE3		1	E		*POS	45	EQ	'2'		A1		
13	Z						E		*POS	45	EQ	'3'		A2		
14	Z								*POS							
15	Z								*POS							
	Z								*POS		EQ	' '				
	Z		5	MATH		1	E		*POS		EQ	' '		A3		
	Z								*POS							
	Z								*POS							
	Z								*POS		EQ	' '				

Manual Advance

Automatic Advance

Conditionally Automatic Advance

Automatic Advance (from a subroutine)

Options

Job Line

CFILE (data set)
DATE (+DMY/+YMD)
EDITC (cuptd)
ENTRATR ('attr ')
EXITATR ('attr ')
JOBOPT ([+NOPMT][+NOOPEN])
PRTFILE (data set)
SHARE (names)
SHARER (names)
STATUS (name)
TFILE (data set [delfreq])

'attr=BL,CS,HI,ND,RI,UL

Entry Lines

CLRL (number)
EOJ ([{'job' dev} {*PASS}])
SLNO (line)
WRITE (name)

*Number of sheets per pad may vary slightly.

REVIEW FORMAT STATEMENTS

Records written to the transaction file contain data only. There are no identifiers automatically supplied to show the format used to create each record. During verify, update, and rerun modes of operation, a review format statement, which is optional, provides the means for identifying the entry format to use with a record read from the data set assigned as the transaction file.

The use of review format statements requires that the record types be identified by characters in the records. Therefore, the records written to the transaction file must be designed to accomodate this need for identification.

When you code a review format statement, specify both the character and the position to test. You also specify the ID of an associated entry format, which is the entry format to be used to display the record. (This can be a different format than the one used to create the record.)

When the test character is the same as the character in the test position, the record is displayed in the entry format specified. If the characters are different, the test indicated by the next review format statement is applied. When none of the tests specified yield an equal comparison, the entry format logical tests are applied. If these also fail, the record read from diskette is displayed in the last format used.

If a review format statement is left blank in the test conditions fields, all the records pass the test. No further tests are conducted, and the records are displayed in the format specified.

The tests are conducted in the sequence in which they are specified. Each time a new record is read from the data set assigned as the transaction file, testing starts with the first test specified.

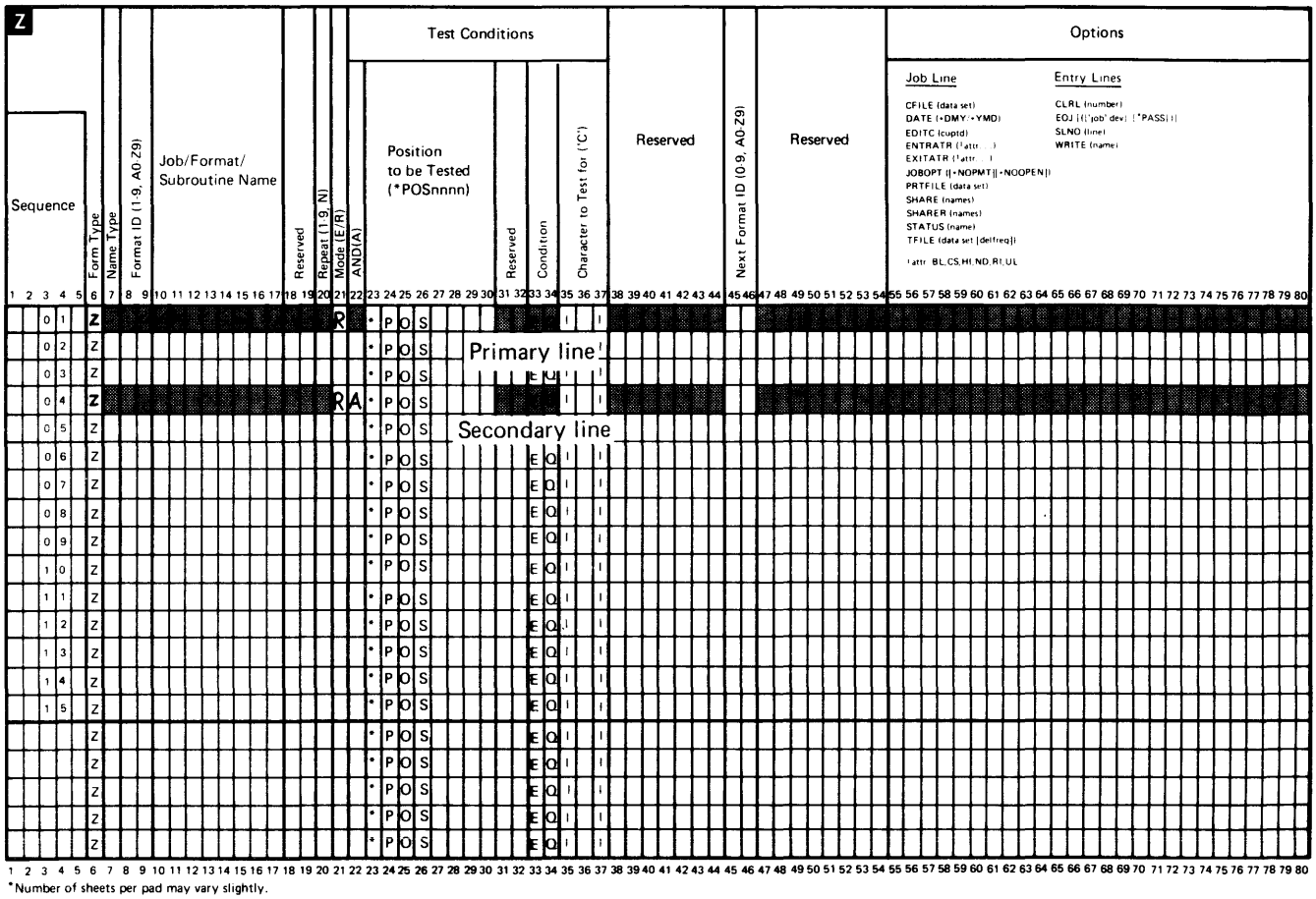
The fields that can be coded for review format statements are shown as unshaded fields in the following illustration.

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A.

Job		Keying Instruction	Graphic								Description	Page of
Operator	Date		Key									



*Number of sheets per pad may vary slightly.

Sequence Number Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

Enter a Z in the *Form Type* field.

Mode Field (Column 21)

The letter R is required in the *Mode* field for review format statements.

Test Conditions (Columns 23 through 30 and 35 through 37)

To specify a single test to be used on a record that is read from the data set assigned to the transaction file, use the *Position to be Tested* field and the *Character to Test for* field.

Specify the position to be tested as *POS*n* (with *n* as a number from 1 to 8192 representing the position in the record) in the *Position to be Tested* field. If you use the forms, the entry starts in column 23.

Specify the single character for which to test between a pair of apostrophes in the *Character to Test for* field.

Next Format ID Field (Columns 45 and 46)

In review format statements, the *Next Format ID* field means *use this format ID for this record*. Enter the ID of the entry format that is to be used to display the record. The ID specified cannot refer to an entry mode statement that specifies the keyword WRITE(*NO). The entry in the *Next Format ID* field must be either characters used as the ID of an entry format statement or a zero (0), which represents format 0. Format 0 displays the record as a string of one-position fields without special edits. The keyboard is in alphabetic shift when format 0 is used.

You can specify two tests to relate a record read from the data set assigned to the transaction file to the entry format used to display the record. This ANDing of tests requires a secondary line to supplement the primary line.

To AND two tests, leave the *Next Format ID* field on the primary line blank. The rules for the *Test Conditions* and *Next Format ID* fields in a secondary line are identical to those for the primary line for single test, except that you must enter the letter A in the *AND* field in the *Test Conditions* field.

When conditions are ANDed, both tests must be true for the record to be displayed in the specified format.

Example

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A.

Job		Keying Instruction	Graphic								Description	Page of
Operator	Date		Key									

Z

Sequence	Form Type	Name Type	Format ID (1-9, A0-Z9)	Job/Format/ Subroutine Name	Reserved	Repeat (1-9, N)	Mode (E/R)	AND/(A)	Test Conditions: Position to be Tested (*POSnnnn)	Reserved	Condition	Character to Test for ('C')	Reserved	Next Format ID (0-9, A0-Z9)	Reserved	Options: Job Line / Entry Lines
1 2 3 4 5	6	7	8 9	10 11 12 13 14 15 16 17	18	19	20	21	22 23 24 25 26 27 28 29 30	31 32	33 34	35 36 37	38 39 40 41 42 43 44	45 46	47 48 49 50 51 52 53 54	55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
01	Z															
02	Z								R *POS45		EQ	'2'		A1		
03	Z															
04	Z															
05	Z															
06	Z															
07	Z								R *POS45		EQ	'2'		A1		
08	Z								R *POS45		EQ	'3'		A1		
09	Z															
10	Z															
11	Z															
12	Z															
13	Z								R *POS45		EQ	'B'				
14	Z								AA *POS46		EQ	'5'		A4		
15	Z															
	Z															
	Z															
	Z															
	Z															
	Z															

If the record contains a 2 in position 45, use the format identified by A1 to display the record.

If the record contains either a 2 or a 3 in position 45, use the format identified by A1 to display the record.

If the record contains both a B in position 45 and a 5 in position 46, use the format identified by A4 to display the record.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

Chapter 5. Contents

Chapter 5. A-Specifications

The following statements can be coded on the A-specification:

- File description statement
- Record description statement
- Field description statement
- Table description statement
- Literal statement

FILE DESCRIPTION STATEMENTS

A file description statement describes a file that is to be used in the program. A file provides a logical path for passing data between an I/O device and the program and for controlling I/O operations. A file description statement must be coded for each data set used during program execution.

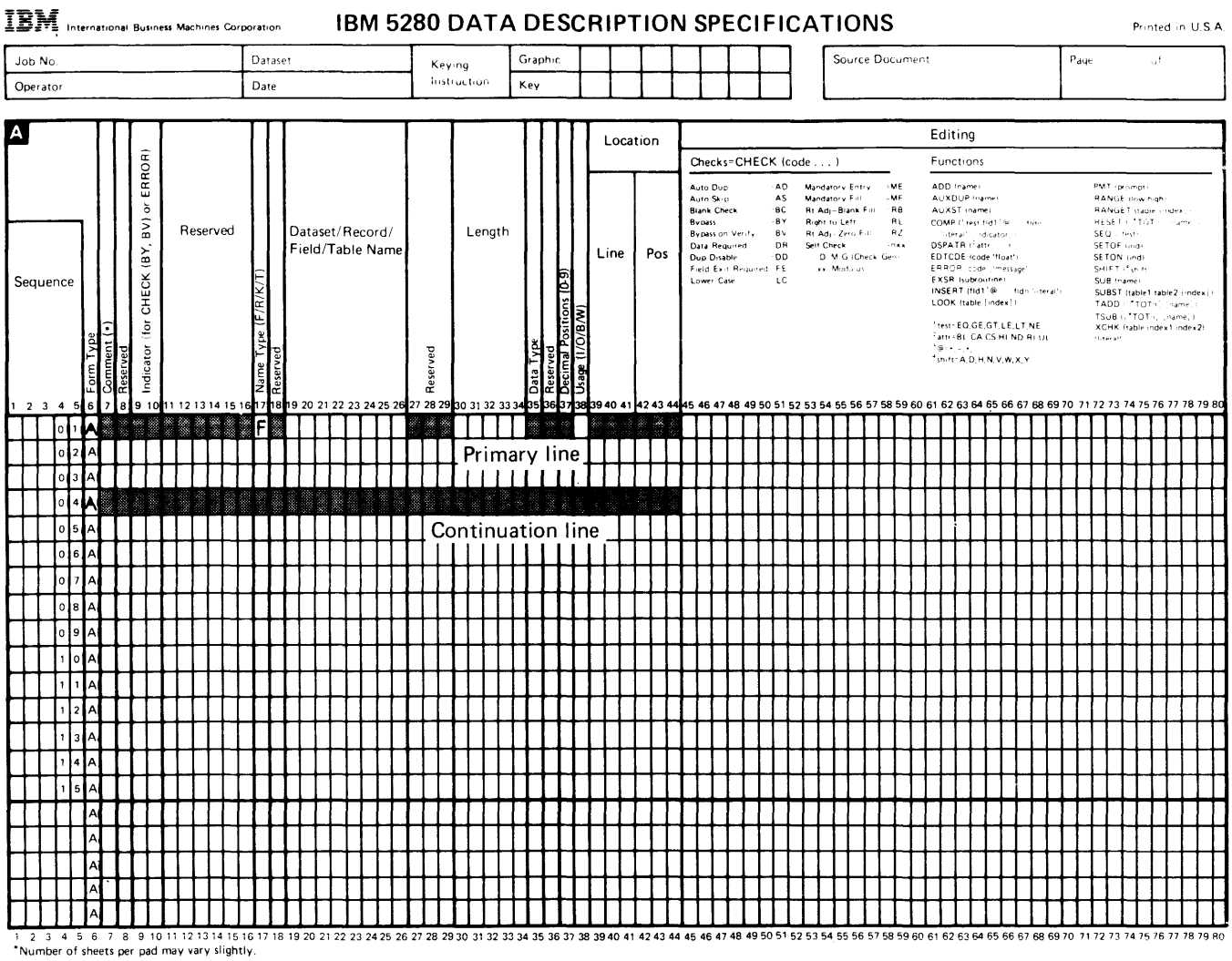
Only one keyboard/display file can be defined for each program. The maximum number of other files that can be defined for each program is 14. However, this number is reduced when some kinds of diskette files are used, as follows:

- For each indexed file that is defined in a program, the maximum number of files that can be described is reduced by one.
- For each multivolume file that is defined to use multiple diskette drives (specified by the DEVICE keyword), the maximum number of files that can be described is reduced by the number of drives used by the multivolume file, minus one.

Although a file description statement must be coded for each compile-time table used in a program, these statements are not counted as files and are not included in the maximum number of files defined in a program.

During program compilation, a file description statement must precede the record and field description statements (or table description statements).

The fields that can be coded for file description statements are shown as unshaded fields in the following illustration.



*Number of sheets per pad may vary slightly.

Sequence Number Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

Enter an A in the *Form Type* field.

Name Type Field (Column 17)

The letter F is required in the *Name Type* field for file description statements.

Name Field (Columns 19 through 26)

Every file description statement must have a name entered in the *Name* field. When the file is to function as the transaction file, the copy file, or the print file, the file name must be the same as the name used with the appropriate keyword in the job specification statement.

The name entered in the *Name* field of a file description statement is used as the data set label unless the LABEL keyword is used.

Length Field (Columns 30 through 34)

An entry in the *Length* field is required. The entry in the *Length* field represents the length of the logical records in the file. This length must be equal to or greater than the sum of the field lengths of all input and both fields (I or B specified in the *Usage* field) in the largest record in the file. Only numbers can be used. The length specified in a file description statement for a printer should be the width of a desired print line. The *Length* entry must be right adjusted.

The maximum entry for the *Length* field depends on the type of file being defined, as follows:

File Type	Maximum Entry
Diskette	8192
Printer	255
Magnetic stripe reader	128
Communications	512
Communications (IBM 3270)	1923

Usage Field (Column 38)

The *Usage* field can be used in file description statements to specify a default usage for subordinate field description statements. This field applies in file description statements associated with the keyboard/display only. The entry in the *Usage* field affects the source of data, the display of data in the fields, and the assembly of data into a record for the transaction file.

Valid entries in the *Usage* field are:

- I for an input field default
- O for an output field default
- B for a default that is for both input and output
- W for a work space field default

The *Usage* field can be left blank if a default value for the fields is not desired.

Data for input fields (I in the *Usage* field) can be entered by the operator, duplicated from the corresponding positions in the preceding record, auxiliary duplicated, or placed in the field via the INSERT keyword. Information entered by the operator can be changed by the operator during verify and update modes of operation.

In enter mode, execute mode, and verify mode, input fields are initially displayed as blanks. The data is displayed as it is entered or as it is verified. Fields that are filled by duplication or via the INSERT keyword are displayed in the sequence in which they occur in the source during compilation. During the update mode of operation, input fields are initially displayed and continue to be displayed for the entire record.

Data for output fields (O in the *Usage* field) or as work space (W in the *Usage* field) cannot be entered by the operator. Output fields can either be named or have data provided via a literal, but never both. A work space is an area in the program that is reserved for the program's use. A work space cannot be displayed or have data entered into it by the operator. A work space must have a name and can be initialized via the keyword INSERT and its parameter string. (The work space is initialized when the format is executed.) The data in a work space or in an output field cannot be changed directly by an operator, but can be changed as a result of an operation on data changed by the operator.

Fields for both input and output (B in the *Usage* field) are treated as input fields except during the execute mode of operation. During execute mode, the initial value of the field is displayed at the start of the format use. The operator has the opportunity to change the data in a key-entered field or to accept the data as displayed. All fields that are described as both input and output (B in the *Usage* field) must be named.

During enter mode, execute mode, verify mode, and update mode, fields that contain literals and named output fields are initially displayed. The data remains displayed for the remainder of the record operation, unless the data is intentionally overwritten on the display.

The contents of a work space are never displayed.

Editing Field (Columns 45 through 80)

Keywords and their parameters can be entered in the *Editing* field to provide specific information about the file characteristics and about the device assigned to the file.

The keywords that can be used in a file description statement are in the following chart:

Keyword	Description
BLKING	Blocking
CHECK	Keyboard level edits
DEVICE	Device type
DSPATR	Display attributes
DSPSIZ	Display size
FORM	Paper size
INDEX	Name of index file data set
LABEL	Name of data set on diskette
LOGON	SNA communications log on
MARK	Error mark
NUMENT	Number of entries
VMARK	Verify mark

BLKING (*DBL *format*)

The BLKING keyword applies only to diskette data sets and specifies the blocking factors to be used when writing to diskette.

The *DBL parameter specifies double buffering, that is, two storage spaces allocated to receive data during program execution. If *DBL is omitted, the default is used. The default for the transaction file and for keyed files, for which WRITE or UPDATE operations are specified, is double buffering, and the default for other files is single buffering.

The format parameter describes blocking and spanning characteristics for the diskette. The valid entries for the format parameter are:

- *FMTU for unblocked and unspanned, which starts each logical record (data record) at the beginning of a physical record (sector).
- *FMTS for blocked and spanned, which places logical records contiguously in the file.

Unblocked and unspanned organization can result in unused space on the diskette. Blocked and spanned organization uses diskette space more economically. See *Appendix C* for examples.

The default value for the *format* parameter is *FMTS.

CHECK(DD)

The CHECK keyword allows a keyboard level edit to be specified for data as it is entered. The parameter DD is the only keyboard level edit that can be specified in a file description statement. This parameter prohibits the use of the DUP key for records in the file.

DEVICE(*device address*)

The DEVICE keyword assigns a device to the file. This keyword is required in all file description statements except the file description statement associated with compile-time tables, which are included in the source program during compilation.

The device parameter is required. Valid entries for device are:

- COMM for communications
- COMM3270 for communications (3270 program interface)
- CRT for the keyboard/display
- DISK for a diskette drive
- MREAD for a magnetic stripe reader
- PRINTER for a printer

The address parameter specifies the address of the device. The address can be specified in either of two forms:

- Two-character logical ID, which must be defined during system configuration
- Four-character physical device code expressed as X'nnnn'

If a multivolume diskette file is to be processed, more than one address parameter will be issued. Up to eight different addresses can be specified. In this case, the number of addresses specified indicates the number of volumes to be processed. A sequential multivolume file can be processed on a single diskette drive. This is specified by coding the same address twice. Up to 99 volumes can be processed on the specified drive.

Refer to your system configuration records for the logical IDs and device codes that are valid on your system. The following logical IDs are recommended:

Device Entry	Logical ID
COMM	None required
COMM3270	None required
CRT	None required
DISK	Dn
MREAD	None required
PRINTER	Pn

Where n ranges from 1 to the number of devices (of a particular type) attached to the system.

The address parameter is invalid if the device parameter specifies CRT, COMM or COMM3270.

When JOBOPT(*NOPMT) is specified in the job specification statement, the address parameter is required. Otherwise, an entry for the address parameter is optional.

DSPATR(*attributes*)

The DSPATR keyword permits control of display characteristics via one or more of the following attributes. The attributes are effective for all the fields in the records in the file. DSPATR is valid only in file description statements for a keyboard/display file.

The following are the attributes that can be specified.

Attribute	Meaning
BL	The displayed record blinks (flashes on and off).
CS	Column separators display between character positions. Column separators are thin vertical lines which are displayed between character positions and do not reduce display capacity.
HI	The displayed characters are highlighted (displayed with increased intensity).
ND	The records are not displayed.
RI	The records are displayed with images reversed (dark characters are displayed on a light background).
UL	Each character position in the records is underlined.

The combination of attributes HI, RI, and UL is invalid. If this combination is specified, the records are not displayed.

DSPSIZ(*lines characters*)

The DSPSIZ keyword specifies the minimum display size that can be used during program execution. The keyword is optional, and can be used only in file description statements to which the keyboard/display is assigned. Both the lines parameter and the characters parameter are required.

The default display size is the 480-character display.

Valid entries for the parameters follow.

Lines	Characters	
6	80	for a 480-character display
12	80	for a 960-character display
24	80	for a 1920-character display

FORM(*length overflow indicator*)

The FORM keyword allows you to control page organization for printed output. FORM can be used only in file description statements in which DEVICE(PRINTER) is specified. The FORM keyword is not valid for a printer file that is specified for the PRTFILE keyword in the job specification statement.

The keyword FORM has three parameters, which must be specified in the indicated sequence.

The first parameter, *length*, is required and must be a whole number representing the number of possible print lines in the form. For example, if the form is 11 inches long and there are six lines to the inch, the length parameter should be 66. The valid range for the length parameter is 1 to 255.

The second parameter, *overflow*, represents the line number for the overflow line. When the overflow line is passed, the form is automatically advanced to the first line on the next page. The valid range for the overflow parameter is 1 to the value specified in the length parameter.

The last parameter, *indicator*, specifies the overflow indicator. This parameter can be specified only if the second parameter is used. The entry must be an indicator (01 through 99) which is set when the line number specified as the *overflow* parameter is passed. When the overflow indicator is specified, the automatic form advance to the first line is suspended. This allows the printing of totals at the bottom of a page and the heading information at the top of the next page via page control logic in the program.

Note: Remember that whenever an indicator is set on in a program it must be set off later in the program.

If DEVICE(PRINTER) is specified and the FORM keyword is not specified, a default value of 66 is assumed for the length parameter and a value of 60 is assumed for the overflow parameter. No indicator is assumed for the indicator parameter. If the FORM keyword is specified with only the length parameter, no end-of-page detection is performed and any end-of-page processing must be included in the program.

INDEX(*storage file*)

The INDEX keyword allows you to govern the organization and access of data sets that are explicitly read or written via calculation statements (C specifications). The INDEX keyword is used when:

- All the records in the file have a key field with the same field length and location.

- The records in the file do not have any key fields but an ADDROUT index has been produced by the sort program.

For more information on key index data sets and ADDROUT indexes, see *Appendix C.*

Either or both of the parameters can be specified. Storage, if used, must be a whole number (from 2 through 32767) representing the number of keys to be kept in storage as a second-level index for the file. The storage parameter cannot be specified for an ADDROUT index. If the records have key fields and the storage parameter is not specified, the default value is 5, which reserves the length of the key field plus three times five for a second-level index. The second-level index is built and maintained within the system to improve the speed of record access.

The file parameter, if used, must be the name of the key index data set or the ADDROUT index data set on diskette. The name can be specified as a data set label. For data set label conventions, see the LABEL keyword description in *File Description Statements* in this chapter. The index file cannot be defined elsewhere in the source program.

When the records in a data set contain key fields and the associated file description statement does not contain the keyword INDEX, or contains the keyword INDEX without the file parameter, the records in the file are assumed to be in key sequence. When the key field of a write operation would logically place a new record between two existing records, the new record is inserted in the correct sequence and all following records are rewritten until a deleted record is encountered.

When the records in a data set contain key fields and the associated file description statement contains the keyword INDEX with the *file* parameter, the index data set is used for record access. The records occur in the data record file in the order in which they are written, and the index records (comprised of the key field and the location of the record in the data record file) are in key sequence. Each record added to the data set results in two records; the data record, which is added at the end of the data set, and the index record, which is inserted into the proper position in the index file in key sequence.

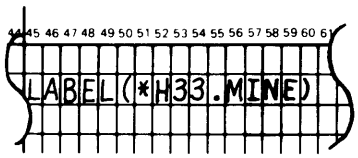
The address of the index data set is assumed to be the same as the address parameter of the data set that contains the data records. For multivolume files, the index data set are assumed to be on the same drive as the first volume of the file.

LABEL(*name*)

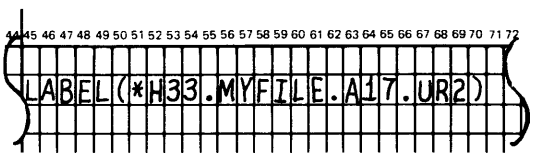
The LABEL keyword allows you to specify a data set label (or name) for use as a header label on a diskette when different from the *Name* field. The keyword is optional and, if used, requires a single parameter.

If you are creating an I exchange data set, the *name* parameter must be specified as it appears on a header label. The format for *name* can be either VOL.NAME or NAME, in which VOL is the volume ID specified in the form *Vaaaaa, and NAME is a simple data set name with a maximum length of 8 characters.

Within the volume ID specifier, the * identifies the *Vaaaaa* as a volume ID. The *V* can be any letter (A through Z), and the *aaaaa* is a string of up to five characters, which can be either letters (A through Z) or numbers (0 through 9).



If you are using a data set that is already created and is not I exchange, the data set name can be a simple name or a series of simple names connected with periods. A simple name can contain up to eight characters. The data set name can have a maximum length of 17 characters. The first character must be a letter (A through Z). The remaining characters can be letters (A through Z) or numbers (0 through 9).



The name provided can be overridden if JOBOPT(*NOPMT) is not specified in the job specification statement.

LOGON(*parameter*)

The LOGON keyword is for specifying log-on information to be sent to the host system when the communications file is opened. LOGON is valid only in file description statements assigning the device COMM. The required parameter can be either a character constant enclosed between apostrophes or the name of a field that contains the log-on information.

The maximum length of the field is 80 characters.

MARK(*POS*nnnn*)

The MARK keyword allows you to specify a position in the records for indicating that the operator used the Mark Field key in the record.

If the Mark Field key is used during the entry of the record, the character E is written into the specified position in the diskette record to show that the record contains errors. A solid rectangular block (■) is displayed in the leftmost position of the current field. The E remains in the diskette record until it is intentionally replaced.

The required parameter is *POS*nnnn*, where *nnnn* is a number representing the position to be marked.

The MARK keyword is valid in a file description statement for the transaction file only.

NUMENT(*number*)

The NUMENT keyword is used for two purposes. NUMENT can specify the number of entries in a table. If a file contains one or more tables, the NUMENT keyword is required. All the tables in a single file must have the same number of entries.

Also, the keyword NUMENT can specify the number of records to be used for dynamic allocation of record space for an output data set.

The required parameter, *number*, specifies the number of entries in the table or the number or records in the data set and must be entered as a whole number. The maximum *number* allowed is 16 777 215.

VMARK(*POS*nnnn*)

The VMARK keyword allows you to specify a position in the records for indicating that the record has been verified. When the record is verified, the character V is inserted into the record in the specified position. The V is removed whenever the record is rewritten to the data set. (Rewriting occurs if the record is updated.)

The required parameter is *POS*nnnn*, where *nnnn* is a number representing the position to be marked.

The VMARK keyword is valid in a file description statement for the transaction file only.

Example

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	A					F		NAMERO		30							DEVICE(CRT) DSPSIZ(12 80)
2	A	This statement describes a file named NAMERO. NAMERO is assigned to the keyboard/display.															
3	A	The minimum size display device that can be used is the 960-character display.															
4	A																
5	A																
6	A																
7	A																
8	A																
9	A																
10	A					F		INVOICE		230							DEVICE(DISK) BLKING(*DBL)
11	A	This statement describes a file named INVOICE. INVOICE is a diskette file (or data set) that															
12	A	contains 230-byte records. Double buffering is used with this file.															
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

Checks=CHECK (code . . .)

Auto Dup	AD	Mandatory Entry	ME
Auto Skip	AS	Mandatory Fill	MF
Blank Check	BC	Rt Adj–Blank Fill	RB
Bypass	BY	Right to Left	RL
Bypass on Verify	BV	Rt Adj–Zero Fill	RZ
Data Required	DR	Self Check	nxx
Dup Disable	DD	n: C/G (Check Gen)	
Field Exit Required	FE	xx: Modulus	
Lower Case	LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low high)
AUXST (name)	RANGET (table {index})
COMP ([1]test fld1 [1]@ fldn {indicator})	RESET (+TOTn)
DSPATR ([2]attr)	SEQ ([1]test)
EDTCDE (code {fcat})	SETOF (ind)
ERROR (code {,message})	SETON (ind)
EXSR (subroutine)	SHIFT ([4]shift)
INSERT (fld1 [1]@ fldn)	SUB (name)
LOOK (table {index})	SUBST (table1 table2 {index})
	TADD (+TOTn)
[1]test EQ,GE,GT,LE,LT,NE	TSUB (+TOTn)
[2]attr BL,CA,CS,HI,ND,RI,UL	XCHK (table index1 {index2}
[1]@ +, -, *, /	literal
[4]shift A,D,H,N,V,W,X,Y	

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

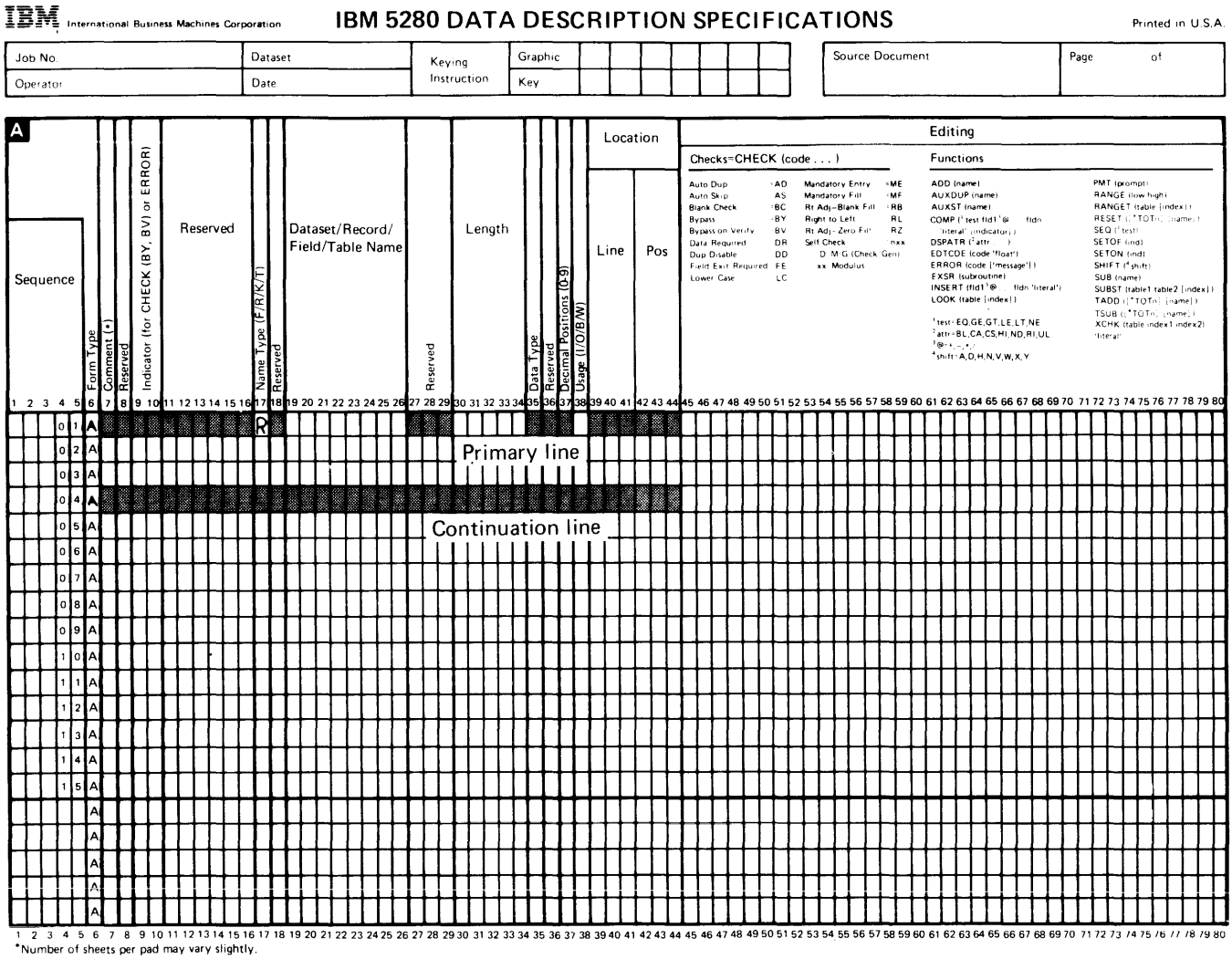
*Number of sheets per pad may vary slightly.

RECORD DESCRIPTION STATEMENTS

Record description statements name the record and describe characteristics that apply to the entire record.

During program compilation, a file description statement for the file to which the records are assigned must precede record description statements. A record description statement must precede the field description statements that describe the fields in the record, if any.

The fields that can be coded for record description statements are shown as unshaded fields in the following illustration.



*Number of sheets per pad may vary slightly.

Sequence Number Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

Enter an A in the *Form Type* field.

Name Type Field (Column 17)

The letter R is required in the *Name Type* field for record description statements.

Name Field (Columns 19 through 26)

Each record must be named. Follow the naming conventions stated in the *General Coding Conventions* section of this manual.

Length Field (Columns 30 through 34)

The length field is used to specify a record length for each communication file (COMM or COMM3270) record. The length specified determines the record length to be transmitted during a WRITE operation. The record length must be less than or equal to the length specified on the file description statement.

Other files use the length specified on the file description statement.

Usage Field (Column 38)

The *Usage* field can be used in record description statements to specify a default usage for subordinate field description statements. This field applies in record description statements associated with the keyboard/display only. The entry in the *Usage* field affects the source of data, the display of data in the fields, and the assembly of data into a record for the transaction file.

Valid entries in the *Usage* field are:

- I for an input field default
- O for an output field default
- B for a default that is for both input and output
- W for a work space field default

The *Usage* field can be left blank if a default value for the fields is not desired.

Data for input fields (I in the *Usage* field) can be entered by the operator, duplicated from the corresponding positions in the preceding record, auxiliary duplicated, or placed in the field via the INSERT keyword. Information entered by the operator can be changed by the operator during verify and update modes of operation.

Data for output fields (O in the *Usage* field) or as work space (W in the *Usage* field) cannot be entered by the operator. Output fields can either be named or have data provided via a literal, but never both. A work space is an area in the program that is reserved for the program's use. A work space cannot be displayed or have data entered into it by the operator. A work space must have a name and can be initialized via the keyword INSERT and its parameter string. The data in a work space or in an output field cannot be changed directly by an operator, but can be changed as the result of an arithmetic operation on data changed by the operator.

In enter mode, execute mode, and verify mode, input fields are initially displayed as blanks. The data is displayed as it is entered or as it is verified. Fields that are filled by duplication or via the INSERT keyword are displayed in the sequence in which they occur in the source during compilation. During the update mode of operation, input fields are initially displayed and continue to be displayed for the entire record.

Fields for both input and output (B in the *Usage* field) are treated as input fields except during the execute mode of operation. During execute mode, the initial value of the field is displayed at the start of the format use. The operator has the opportunity to change the data in a key entered field or to accept the data as displayed. All fields that are described as both input and output (B in the *Usage* field) must be named.

During enter mode, execute mode, verify mode, and update mode, fields that contain literals and named output fields are initially displayed. The data remains displayed for the remainder of the record operation, unless the data is intentionally overwritten on the display.

The contents of a work space are never displayed.

Editing Field (Columns 45 through 80)

The *Editing* field can be used to describe and to control characteristics that apply to the entire record. The group of keywords that can be used depends upon the device with which the file is associated.

The keywords that can be used in a record description statement are:

Keyword	Description
CHECK	Keyboard level edits
DSPATR	Display attributes
RECID	Record ID
SETOF	Set indicator off
SETON	Set indicator on
SKIPA	Skip after
SKIPB	Skip before
SPACEA	Space after
SPACEB	Space before

CHECK(DD)

The CHECK keyword allows a keyboard level edit to be specified for data as it is entered. The parameter DD is the only keyboard level edit that can be specified in a record description statement. This parameter prohibits the use of the Dup key for the record.

DSPATR(*attributes*)

The DSPATR keyword permits control of display characteristics via one or more of the following attributes, which are effective for all the fields in the record. The attributes specified by this keyword are combined with those specified in the file description statement (if any) to establish the display attributes to be used for all the fields in the record. DSPATR is valid only in record description statements associated with the file for the keyboard/display.

The following are the attributes that can be specified.

Attribute	Meaning
BL	The displayed record blinks (flashes on and off).
CA	Cancels display attributes propagated from the file description statement. The CA attribute can be used when an attribute or characteristic specified in the file description statement is not desirable. Other parameters can then be specified to control the display for the record. The parameter CA affects only the record for which it is coded.
CS	Column separators display between character positions. Column separators are thin vertical lines, which are displayed between character positions and do not reduce display capacity.
HI	The displayed characters are highlighted (displayed with increased intensity).
ND	The fields in the record are not displayed.
RI	The fields in the record are displayed with images reversed (dark characters are displayed on a light background).
UL	Each character position in the fields in the record are underlined.

The combination of attributes HI, RI, and UL is invalid. If this combination is specified, or if this combination is formed by the display attributes specified in the file description statement and this statement, the record is not displayed.

RECID(*POS*nnnn 'x')

The RECID keyword can be used in record description statements for any records (input) except those from the transaction file or the keyboard/display. RECID allows you to specify the identifying character and the position to test to identify the record type when records are read. The ability to identify record types is useful during record accesses via calculation statements.

The first parameter must be specified as *POS*nnnn* where *nnnn* is the character position within the record.

The second parameter must be specified as a single character enclosed in apostrophes.

SETOF*(indicator)*

The SETOF keyword can be used in record description statements for any input records except those from the transaction file or the keyboard/display. (For records for the transaction file or the keyboard/display, SETOF must be specified only in the appropriate field description statements.) This keyword is used in conjunction with the RECID keyword to identify to the program the type of record read. The effect of these keywords depends on the order in which the keywords are specified. When RECID is specified before the SETOF keyword and the record type specified by the RECID keyword is read, the indicator specified by the SETOF keyword is turned off. When SETOF is specified before RECID, the indicator is set off before the record type is checked and the results of the RECID keyword have no effect on the indicator. The indicator can be tested by operations in the calculation specifications and can be used to control which calculation operations are performed on the record.

The keyword SETOF allows you to turn off any one of the indicators (01 through 99). The single required parameter, *indicator*, must be the number of the indicator to be turned off.

SETON(*indicator*)

The SETON keyword can be used in record description statements for any input records except those from the transaction file or the keyboard/display. (For records for the transaction file or the keyboard/display, SETON must be specified only in the appropriate field description statements.) This keyword is used in conjunction with the RECID keyword to identify to the program the type of record read. The effect of these keywords depends on the order in which the keywords are specified. When RECID is specified before the SETON keyword and the record type specified by the RECID keyword is read, the indicator specified by the SETON keyword is turned on. When SETON is specified before RECID, the indicator is set on before the record type is checked and the results of the RECID keyword have no effect on the indicator. The indicator can be tested by operations in the calculation specifications and can be used to control which calculation operations are performed on the record.

The keyword SETON allows you to turn on any one of the indicators (01 through 99). The single required parameter, *indicator*, must be the number of the indicator to be turned on.

SPACEA(*number*)

The SPACEA keyword can be used only in record description statements for printer files to control the number of print lines spaced after the record is printed.

The number parameter must be a whole number from 0 through 3. The value 0 suppresses spacing so that two records can be printed on the same print line.

If the keyword is not used, and the keywords SPACEB, SKIPA, and SKIPB are also not specified, the printer automatically spaces one print line after each record.

SPACEB(*number*)

The SPACEB keyword can be used in record description statements for printer files to control the number of print lines spaced before the record is printed.

The number parameter must be a whole number from 1 through 3. If the keyword is not specified, no space-before-printing is performed.

SKIPA(*number*)

The SKIPA keyword can be used only in record description statements for printer files to control skipping to a new line after the record is printed.

The number parameter must be the line number to which you want to skip. The line number can be any whole number from 1 through the length of the form as specified in the first parameter for the FORM keyword in the associated file description statement.

SKIPB(*number*)

The SKIPB keyword can be used only in record description statements for printer files to control skipping to a new line before the record is printed.

The number parameter must be the line number on which the current record is to be printed. The line number can be any whole number from 1 through the length of the form as specified in the first parameter of the FORM keyword in the file description statement.

Example

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	A																
2	A					R		NAME									
3	A																
4	A																
5	A																
6	A																
7	A																
8	A																
9	A																
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

This statement describes a record named NAME. The relationship between NAME and a device is established by the file description statement that must preceed record description statements. The contents of NAME must be described by field description statements.

Checks=CHECK (code . . .)

Auto Dup	AD	Mandatory Entry	ME
Auto Skip	AS	Mandatory Fill	MF
Blank Check	BC	Rt Adj-Blank Fill	RB
Bypass	BY	Right to Left	RL
Bypass on Verify	BV	Rt Adj-Zero Fill	RZ
Data Required	DR	Self Check	nnn
Dup Disable	DD	n=C/G (Check/Gen)	
Field Exit Required	FE	xxx=Modulus	
Lower Case	LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low:high)
AUXST (name)	RANGET (table [:index])
COMP ([1]test fld1 [3]@ fldn [:indicator])	RESET (+TOTn)
DSPATR ([4]attr)	SEO ([1]test)
EDTCDE (code [:fnat])	SETOF (ind)
ERROR (code [:message])	SETON (ind)
EXSR (subroutine)	SHIFT ([4]shift)
INSERT (fld1 [3]@ fldn)	SUB (name)
LOOK (table [:index])	SUBST (table1:table2 [:index])
	TADD (+TOTn)
[1]test=EQ,GE,GT,LE,LT,NE	TSUB (+TOTn)
[2]attr=BL,CA,CS,HI,ND,RI,UL	XCHK (table:index1:index2)
[3]@=+,-,*,/	'literal'
[4]shift=A,D,H,N,V,W,X,Y	

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

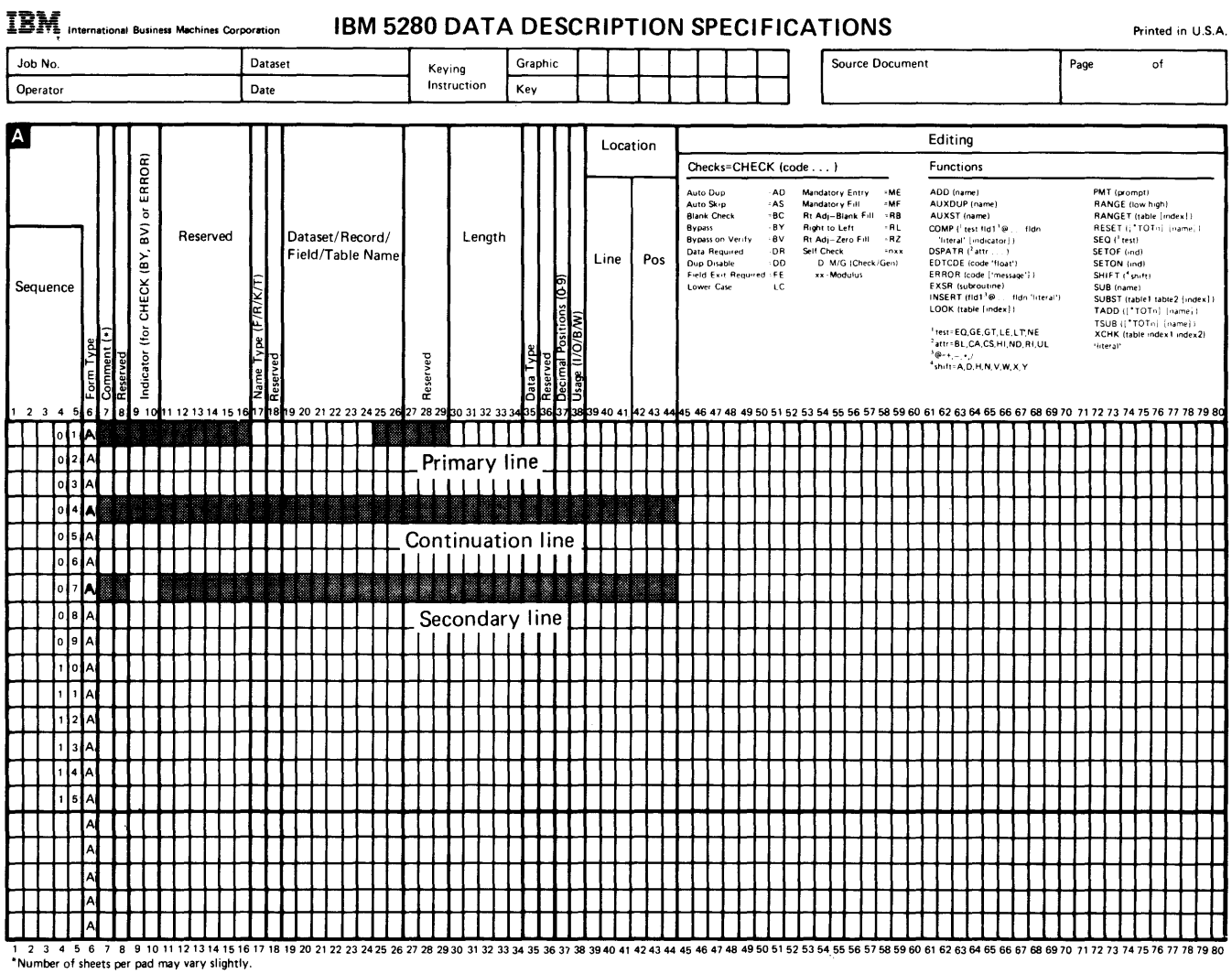
*Number of sheets per pad may vary slightly.

FIELD DESCRIPTION STATEMENTS

Field description statements describe characteristics such as field length, keyboard conditioning during data entry, and data organization for the field.

Field description statements must be preceded by a record description statement.

The fields that can be coded for field description statements are shown as unshaded fields in the illustration that follows.



Sequence Number Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

Enter an A in the *Form Type* field.

Indicator Field (Columns 9 and 10)

An entry in the *Indicator* field is valid only in a secondary line of a field description statement associated with the file for the keyboard/display. The *Indicator* field must be left blank except when the bypass function, the bypass during verification function, or the ERROR keyword is used. Refer to the CHECK keyword and the ERROR keyword descriptions for further information about using the *Indicator* field.

Name Type Field (Column 17)

The *Name Type* field must be left blank in most field description statements. The single exception is the field description for a key field, which requires the letter K in the *Name Type* field. Only one key field can be designated in a record. Key fields are valid only for diskette files that are not specified as the transaction file or the copy file.

Field Name Field (Columns 19 through 24)

The *Field Name* field may be left blank unless the contents of the field are to be used by your DE/RPG program:

- If the order of the fields in a record is changed from the entered sequence for writing to diskette (WRITE keyword in an entry format statement), all input fields (*I* or *B* entered for usage) must be named.
- If the field is used in a calculation or a logical operation, the field must be named.
- If the field is referred to as a source of data for auxiliary duplication, the field must be named.

The reserved words UDATE, *TOTn, and *STATnn can be specified in the name field. UDATE and *STATnn are valid only for output fields; they cannot be modified. Reserved words are predefined in the system. If a length is specified when a reserved word is specified in the name field, the length must match the predefined length for the reserved word. The section *General Coding Conventions* in Chapter 3 contains a list of the reserved words and their definitions.

Field names can be up to six characters long.

The reserved word *RTN can be used in the *Name* field in a field description statement for a keyboard/display file to provide a block of keyword functions within a format, yet keep the functions independent of a field. When *RTN is used in the *Name* field, the only other field that can contain entries is the *Editing* field. Only the keywords RESET, SETON, SETOF, and EXSR and their parameters are valid in the *Editing* field.

Length Field (Columns 30 through 34)

Every field described must have its length specified at least once in a job. Each field description statement in which the *Name* field is blank must have an entry in the *Length* field. Named fields need to have their lengths specified only once in a program. If the length of a named field is specified more than once, each length specification for that field must be the same.

Character fields can contain up to 256 positions. Character fields that are used as key fields must be limited to a maximum length of 28 positions. Numeric fields, which can be used in arithmetic operations, can contain up to 15 positions.

Only the digits 0 through 9 are allowed in the *Length* field; either leading zeros or leading blanks are acceptable. The entry in the *Length* field must be right adjusted.

Data Type Field (Column 35)

The entry in the *Data Type* field (except for P or B) specifies conditioning for the keyboard/display only. An entry is appropriate in field description statements for fields (designated by either I or B in the *Usage* field) associated with the keyboard/display.

The specification of P or B defines the conversion of data between internal and external forms during I/O. An entry is appropriate in field description statements associated with the following devices:

COMM
COMM3270
DISK (except for TFILE)
MREAD

The valid entries for the *Data Type* field are:

Entry	Condition
A	Alphabetic shift – Any character can be entered. The shift is positioned to the lower symbol on each key on all keyboards. The operator can use the shift key to enter the upper symbol on the keys.
B	Binary data – During input, data is converted from twos complement binary form to zoned decimal form. During output, data is converted from zoned decimal form to twos complement binary form. The length of the binary data is determined from the length of the field on which the *Data Type* B is coded. Field lengths of 4 or less are converted to 2 bytes of binary data. Field lengths of 5 to 9 are converted to 4 bytes of binary data. *Data Type* B can only be used with numeric fields that have a field length of 9 or less. (See Chapter 10, *Packed Decimal and Binary Data Formats.)*

Entry	Condition
C	Character check – The characteristics of the keyboard are determined by the parameter for the SHIFT keyword in the field description statement. The SHIFT keyword is required when this data type is specified. See the SHIFT keyword later in this section.
D	Digits only – Only the numbers 0 through 9 can be entered. The shift is positioned to the lower symbols on typewriter keyboards and is positioned to the upper symbols on both data entry keyboards and proof keyboards. The operator cannot override the shift. Negative numbers are displayed with the sign over the units position of the number.
H	Hexadecimal – Each character position requires two keystrokes and only the numbers 0 through 9 and the letters A through F can be entered. No shift key operation is required on data-entry keyboards.

A description of how data types D, N, S, and Y are displayed is in the field exit minus function of the section, *Program Execution*, in Chapter 9.

N	**Numeric shift – Any character can be entered. The shift is positioned to the lower symbols on typewriter keyboards and is positioned to the upper symbols on both the data entry keyboards and the proof keyboards. The operator can change the shift by using the Shift key.**
P	Packed decimal data – During input, data is converted from packed decimal form to zoned decimal form. During output, data is converted from zoned decimal to packed decimal form. The length of the packed data is determined from the length of the field on which the *Data Type* P is coded. The length of the packed decimal data can be calculated using the following formula: Length of packed decimal data = (L÷2) + 1 where L = length of the zoned decimal field *Note:* Any remainder in division is ignored. *Data Type* P can only be used with numeric fields. (See Chapter 10, *Packed Decimal and Binary Data Formats.)*

Entry	Condition
S	Signed numeric – Only the numbers 0 through 9 can be entered. The signed numeric data type implies right adjust with blank-fill, digits only, and field exit required. To leave the field, the operator must use a field exit key to right adjust and properly sign the data. If the operator uses the field advance or record advance function to exit the field, the data is not right-adjusted and signed. Positive values can be terminated by the Field Exit/Skip key, the Field+ key, or the Field Exit+ key. Negative values require the use of either the Field– key or the Field Exit– key. The keyboard shift is positioned to the lower symbols on typewriter keyboards and to the upper symbols on data entry and proof keyboards. When signed numeric data is entered, the system designates an additional display position, which is adjacent to and to the right of the data field. This additional display position is the sign position. It is blank for positive values and is set to - (minus sign) for negative values. However, the field is not lengthened for data written to diskette. Negative values are indicated on diskette when the zone portion of the low-order digit contains a hexadecimal D instead of a hexadecimal F.
V	Right half only – Only the characters that are defined at system generation can be entered. The shift is positioned to the lower right symbols on appropriate keyboards. The shift key allows entry of the upper right symbols.
W	Right half shift – Any character can be entered. The shift is positioned to the lower right symbols on appropriate keyboards. The operator can change the shift by using the shift keys.
X	Alphabetic only – Only the letters A through Z, comma, period, hyphen, and space can be entered. The shift is positioned to the lower symbols on all keyboards. The shift cannot be changed by the operator.
Y	Numeric only – Only the numbers 0 through 9, comma, period, plus, minus, and space can be entered. The shift is positioned to the lower symbols on typewriter keyboards and to the upper symbols on both the data entry keyboards and the proof keyboards. The shift cannot be changed by the operator.

Decimal Positions Field (Column 37)

An entry in the *Decimal Positions* field is valid in field description statements only if there is an entry in the *Length* field. The entry in the *Decimal Positions* field determines whether the described fields contents are character data or are numeric data, and if the field is numeric, the entry specifies the number of positions to the right of the decimal point.

The *Decimal Positions* field must contain a digit (0 through 9) for numeric fields and must be blank for character fields.

Usage Field (Column 38)

The *Usage* field applies in field description statements associated with the keyboard/display only. The entry in the *Usage* field affects the source of data, the display of data in the field, and the assembly of data into a record for the transaction file.

Valid entries in the *Usage* field are:

- I for an input field
- O for an output field
- B for a field that is for both input and output
- W for a work space field

The *Usage* field can also be left blank in field description statements for the keyboard/display. If a default value is entered in the *Usage* field of the corresponding record description or file description statement, then that value is used for the field as well. If the *Usage* field is left blank and there is no default from a file description or record description statement, an entry of I is assumed.

Data for input fields (I in the *Usage* field) can be entered by the operator, duplicated from the corresponding positions in the preceding record, auxiliary duplicated, or placed in the field via the INSERT keyword. Information entered by the operator can be changed by the operator during verify and update modes of operation.

Data for output fields (O in the *Usage* field) or as work space (W in the *Usage* field) cannot be entered by the operator. A work space must be named and can be initialized via the INSERT keyword and its parameter string. Output fields can either be named or have data provided via a literal, but never both. The data in a work space or in an output field cannot be changed directly by an operator, but can be changed as the result of an arithmetic operation on data changed by the operator.

In enter mode, execute mode, and verify mode, input fields are initially displayed as blanks. The data is displayed as it is entered or as it is verified. Fields that are filled by duplication or via the INSERT keyword are displayed in the sequence in which they occur in the source during compilation. During the update mode of operation, input fields are initially displayed and continue to be displayed for the entire record.

Fields for both input and output (B in the *Usage* field) are treated as input fields except during the execute mode of operation. During execute mode, the initial value of the field is displayed at the start of the format. The operator has the opportunity to change the data in a key entered field or to accept the data as displayed. All fields that are described as both input and output (B in the *Usage* field) must be named.

During enter mode, execute mode, verify mode, and update mode, fields that contain literals and named output fields are initially displayed. Once displayed, the data remains displayed for the remainder of the record operation, unless the data is intentionally overwritten on the display.

The contents of a work space are never displayed.

Location Field (Columns 39 through 44)

The entry in the *Location* field specifies the starting (leftmost) position for the placement of data for the associated device. If the device is the keyboard/display, the *Line* and *Position* fields apply. If the device is other than the keyboard/display, the entry in the *Location* field represents the starting position of the field.

An entry in the *Line* field specifies a display line number that is based on the starting line number being used for reference. (The starting line number is the value entered as the parameter for the SLNO keyword in entry format statements.) If the starting line number specified is 4 and the *Line* field entry is 2, the data is displayed on the fifth display line. (If the keyword SLNO is not used in the associated entry format statement, the default value for the starting line number is 2, so 2 is used as the reference for an entry in the *Line* field.)

If the *Line* and *Position* fields are not specified for a field, the starting position for the field defaults to the next available display position following the previous field. The first field defaults to position 1 of logical line 2.

Regardless of whether the SLNO keyword is specified, the defaults for the line and position entries are designed to protect the prompt line from being overlaid by the field being defined. The following table shows the relationships between the SLNO keyword, the line entries, and the physical line on the display.

SLNO Parameter	Line Entry	Physical Line
Blank	Blank	3
Blank	1	2
Blank	2	3
Blank	3	4
2	Blank	3
2	1	2
2	2	3
2	3	4
3	Blank	4
3	1	3
3	2	4
3	3	5

When devices other than the keyboard/display are assigned to the file associated with the fields, the *Location* field applies. The entry in the *Location* field specifies the starting position.

If the *Location* field is blank in this case, the field is started in the next available position. The sequence of the fields is determined by the order in which they occur in the source program.

Entries in the *Location* field and the *Line* and *Position* fields must be right adjusted. Either leading zeros or leading blanks are allowed.

Editing Field (Columns 45 through 80)

Entries in the *Editing* field are optional. An entry in this field can be a single keyword with a parameter string or multiple keywords, each with its parameter string. The *Editing* field can be extended by continuation lines.

In files for the keyboard/display, the order in which keywords are specified generally determines the sequence in which the functions are performed. However, some keywords have priority regardless of their positions.

The keywords with the highest priority are effective before data is entered into the field. These keywords are:

- PMT, for specifying the prompting message displayed on the second display line
- DSPATR, for specifying the display characteristics for the field

Next in priority are the keywords that control the source and placement of the entered data. These keywords are:

- AUXDUP, for controlled duplication from a named variable or field
- CHECK, for specifying keyboard level edits, automatic skipping, and automatic duplication from corresponding positions of the previous record
- INSERT, for placing constants, values from named fields, or the results of arithmetic operations into the field

If the current field is named, the next function is to store the data in the named variable. This will make the data accessible by name to keywords specified on the current field, such as EXSR.

The keywords with the next priority represent functions that occur only after the entry of data into the field is completed. The sequence in which these keywords are entered determines the sequence with which the represented functions are executed. This group includes the following:

- COMP, for specifying a comparison between the entered data and a programmed value
- SEQ, for checking the sequence of the contents of the field against the last field sequence checked
- RANGE, for determining whether the value in the field is within a specified range
- AUXST, for operator-controlled storing of a value for later availability for AUXDUP
- ADD and SUB, for using the data from the field for decimal aligned arithmetic
- TADD, TSUB, and RESET, for using the data from the field for accumulation in reserved counters

In addition to the keywords in the preceding list, this group also includes the keywords for table use (SUBST, RANGET, LOOK, and XCHK) and for indicator manipulation (SETON and SETOF).

The SUBST keyword causes the following sequence of execution:

1. After data has been keyed and if the current field is named, the data is stored in the named variable.
2. The keywords that appear *before* the SUBST keyword are executed in the sequence specified.
3. The substitution function is performed.
4. If the current field is named, the substituted data is stored in the named field.
5. The keywords that appear *after* the SUBST keyword are executed in the sequence specified.

Most keywords in field description statements are for the keyboard/display. Refer to the following chart as a guide for keyword applicability to device types.

Keywords	Keyboard/ Display	Diskette (for TFILE)	Diskette (non-TFILE)	Printer	Communications
ADD	X	.	.	.	.
AUXDUP	X	.	.	.	.
AUXST	X	.	.	.	.
CHECK	X	.	.	.	.
COMP	X	.	.	.	.
DSPATR	X	.	.	.	.
EDTCDE	*	.	X	X	X
ERROR	X	.	.	.	.
EXSR	X	.	.	.	.
INSERT	X	.	.	.	.
LOOK	X	.	.	.	.
PMT	X	.	.	.	.
RANGE	X	.	.	.	.
RANGET	X	.	.	.	.
RESET	X	.	.	.	.
SEQ	X	.	.	.	.
SETOF	X	.	.	.	.
SETON	X	.	.	.	.
SHIFT	X	.	.	.	.
SUB	X	.	.	.	.
SUBST	X	.	.	.	.
TADD	X	.	.	.	.
TSUB	X	.	.	.	.
XCHK	X	.	.	.	.

The X indicates that the keyword can be used.
The . indicates that the keyword cannot be used.
The * means that the keyword can be used in a field description statement only if all fields in the record are output fields and if the record is written to the display (WRITE operation via calculation statements). This is the only keyword allowed for fields contained in records that are written by using the WRITE operation code.

Many keywords for field description statements cannot be used in combination with certain entries in the *Usage* field, in the *Data Type* field, or with certain other keywords and parameters. These restrictions are identified in the section *Keyword Conflicts and Compatibilities* following the descriptions of all the keywords.

If the reserved word *RTN is specified in the *Name* field, the only keywords that can be specified in the *Editing* field are RESET, SETON, SETOF, and EXSR.

Descriptions of the keywords follow. The keywords are described in alphabetic sequence.

ADD(*counter*)

The ADD keyword causes decimal aligned addition of the data in the current numeric field to a counter. The addition occurs only after data entry for the field is completed.

The required parameter, *counter*, must be the name of a numeric field described in the program source.

AUXDUP(*source*)

The AUXDUP keyword allows auxiliary duplication of data from a named source into the current field. The current field may be either named or unnamed.

The required parameter, *source*, must be the name of a defined field.

Both character and numeric data can be auxiliary duplicated, and care should be exercised to ensure that the current field and the source have equal lengths.

During auxiliary duplication, the data is copied to the current field as a string of characters. When the current field length is less than the length of the source, characters that have no corresponding position in the current field are dropped. When the current field is longer than the source, the positions to the right of the data from the source contain unpredictable characters.

The function represented by AUXDUP occurs when either the automatic duplication function is enabled or the DUP key is pressed for the current field.

If the DUP key is pressed after several characters have been manually entered in the current field, the data in the remaining corresponding positions of the source is copied into the current field. The source and the current field are always left end aligned whether field processing progresses left to right (normal) or right to left (with CHECK(RL) specified).

AUXST(*target*)

The AUXST keyword allows you to copy the data from a field and to store the data as a named variable for use in subsequent auxiliary duplication operations. The field from which the data is copied can be either named or unnamed. The auxiliary storage operation is delayed until the data entry and editing for the field is completed.

The required parameter, *target*, must be a field name.

If the target is not defined elsewhere in the program, the compiler will assign the length and decimal positions for the current field.

Either character or numeric data can be stored, and care must be exercised to ensure that the field specified as the target has the correct length.

Data is copied to the target from left to right as a string of characters. When the length of the target is less than the length of the occupied field, characters that have no corresponding positions in the target are dropped. When the length of the target exceeds the length of the copied field, the remaining positions in the target field are unchanged.

The function represented by AUXST occurs only when the automatic duplication function is enabled.

CHECK(*parameters*)

The CHECK keyword allows keyboard level edits to be specified for data as it is entered. CHECK requires at least one of the following parameters.

Parameter	Meaning
AD	Automatic duplication – provides automatic duplication of the current field from the corresponding positions in the preceding record. Automatic duplication can occur only if the automatic duplication function is enabled. During automatic duplication, data type and CHECK keyword edits are ignored. (Automatic duplication produces blanks in execute mode.)
AS	Automatic skip – provides automatic blanking of the current field. An automatic skip can occur only if the automatic duplication function is enabled.
BC	Blank check – prohibits keying of blanks as data in a field. However, the entire field can be blank via either SKIP key or a FIELD EXIT key operated in the first position.
BY	Bypass – bypasses the data field. Any data in the field remains unchanged. A field can be conditionally bypassed if the condition (and indicator) and CHECK(BY) are specified in a secondary line of the field description statement. The secondary line must follow any continuation lines. The only entries allowed in the secondary line are an entry in the *Sequence* field, an indicator number in the *Indicator* field, the required A in the *Form Type* field, and CHECK(BY) in the *Editing* field. In review mode, if the field already contains data, it does not have to be rekeyed.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10 11 12 13 14 15 16		17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40	41 42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	A																
2	A							ZIPCD		5	D			I	3	12	PMT(ENTER ZIP CODE) CHECK(BC)
3	A			23													CHECK(BY)
4	A																
5	A																
6	A																
7	A																
8	A																
9	A																
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																
	A																

Checks=CHECK (code . . .)

Auto Dup	:AD	Mandatory Entry	:ME
Auto Skip	:AS	Mandatory Fill	:MF
Blank Check	:BC	Rt Adj–Blank Fill	:RB
Bypass	:BY	Right to Left	:RL
Bypass on Verify	:BV	Rt Adj–Zero Fill	:RZ
Data Required	:DR	Self Check	:nxx
Dup Disable	:DD	n:C/G (Check/Gen)	
Field Exit Required	:FE	xx:Modulus	
Lower Case	:LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low high)
AUXST (name)	RANGET (table [index])
COMP ([1]test fld1[3]@ fldn [indicator])	RESET (+TOTn)
DSPATR ([2]attr)	SEQ ([1]test)
EDTCDE (code float)	SETOF (ind)
ERROR (code ['message'])	SETON (ind)
EXSR (subroutine)	SHIFT ([4]shift)
INSERT (fld1[3]@ fldn)	SUB (name)
LOOK (table [index])	SUBST (table1 table2 [index])
	TADD (+TOTn)
[1]test=EQ,GE,GT,LE,LT,NE	TSUB (+TOTn)
[2]attr=BL,CA,CS,HI,ND,RI,UL	XCHK (table index1 index2)
[3]@=+,-,*,/	'literal'
[4]shift=A,D,H,N,V,W,X,Y	

The top line describes a field named ZIPCD with a length of five characters. When the field is available for entry, the message ENTER ZIP CODE is displayed on the prompt line, and only digits (0 through 9) can be entered via the keyboard. Entered data displays in the third available line of the display starting in the twelfth position.

The second line is a secondary line. It specifies that if indicator 23 is on, the field is bypassed. The prompting message is not displayed and the entire field is unchanged.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

Parameter	Meaning
BV	Bypass on verify – suspends verification of data in a field. The data is accepted without verification. A field can be conditionally bypassed during verification if the condition (an indicator) and CHECK(BV) are specified in a secondary line of the file description statement. The secondary line must follow the same rules as described above for BY.
DD	Duplication disable – prohibits the use of the DUP key for the current field.
DR	Data required – requires that at least one nonblank character be entered in the field. In review mode, if the field already contains data, it does not have to be rekeyed.
FE	Field exit required – requires the use of a non-data key to advance from the field. Normally, when the last character in a field is entered, the field is exited automatically.
LC	Lowercase – allows both uppercase and lowercase characters to be entered and displayed if the typewriter keyboard is used. Without LC specified, all characters are treated as uppercase characters. LC is ignored if the typewriter keyboard is not being used.
ME	Mandatory entry – requires that at least one character be entered. A blank satisfies the requirement for a character. In review mode, the field must be keyed even if it already contains data.
MF	Mandatory fill – specifies that if a field is filled with blanks when the cursor moves into it and a character is entered, the field must be filled. The cursor can move into the field forward or backward. If the field is not filled with blanks when the cursor moves into it (forward or backward), no mandatory fill checking is done. If no character is entered, the field can be normally skipped.
*RB	Right adjust with blank fill – when specified on a field of two or more positions, shifts the data to the rightmost positions of the field and fills the remaining positions with blanks when a field exit key is used before the field is full.
*RZ	Right adjust with zero fill – when specified on a field of two or more positions, shifts the data to the rightmost positions of the field and fills the remaining positions with zeros when a field exit key is used before the field is full.

**Note:* The Skip key will not perform right-adjust if used to exit the field. See the description of the Skip function in Chapter 9, *Key-Initiated Functions*.

Parameter	Meaning
Mxx	Self check (modulus xx) – imposes a check of the entered data according to the self check algorithm specified by XX. If XX equals 10 or 11, the self check algorithm for IBM modulus 10 or IBM modulus 11 is used. Other algorithms can be user supplied. See *Self Check Processes* in Chapter 7 for algorithm coding information.
GXX	Self check generate (modulus XX) – generates the necessary check digits to complete a self check number sequence. See *Mxx* above.

Note: In the IBM modulus 11 operation, basic numbers that require a check digit of 10 cannot be used as self check numbers. The accounting system must be adjusted to eliminate such numbers from codes that are used in the generation of check digits or in numbers that are to be self checked. If these numbers are used, an error will be displayed to the operator.

RL	Right to left – places the entered data such that the rightmost position is filled first and successive characters fill to the left. If the Mark field key is used when the field is being entered, the mark is placed in the leftmost position of the field.

COMP(*test data indicator*)

The COMP keyword allows you to specify comparisons of the data in the current field with a constant or a named field or expression. The comparison is made after data entry in the current field is completed.

The test and data parameters are required, and the use of the indicator parameter is optional.

The test parameter must be chosen from the following.

Parameter	Meaning
EQ	The data in the current field is equal to the specified data.
NE	The data in the current field is not equal to the specified data.
LT	The data in the current field is less than the specified data.
GT	The data in the current field is greater than the specified data.
LE	The data in the current field is less than or equal to the specified data.
GE	The data in the current field is greater than or equal to the specified data.

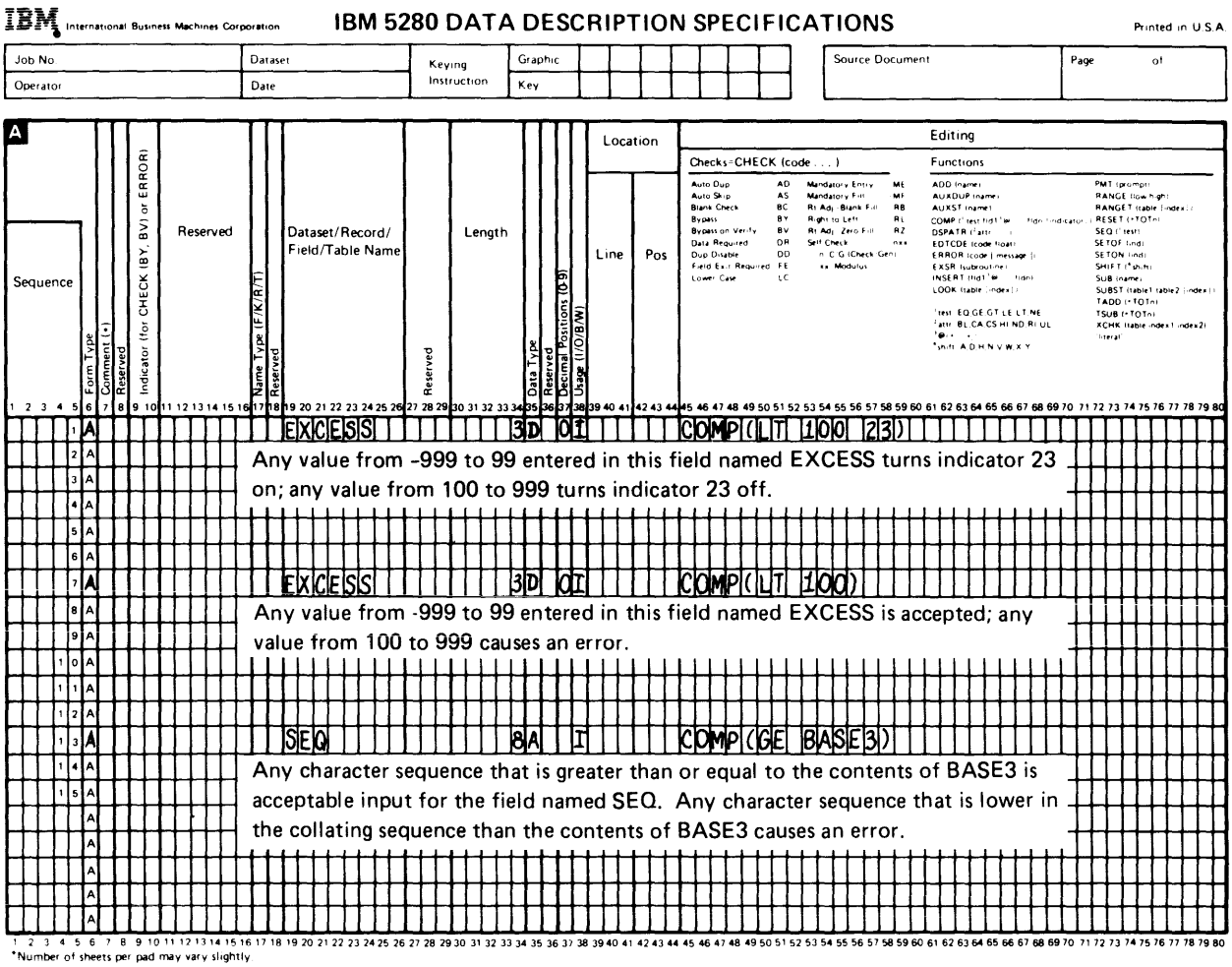
The data parameter, which specifies the data with which the current data is compared, can be:

- The name of a defined field or variable
- A constant
- An arithmetic expression involving addition, subtraction, multiplication, and division, using named fields or variables, and constants

The arithmetic symbols that can be used are +, -, * and / in any combination. The expression is evaluated from left to right with multiplication and division performed before addition and subtraction.

The last and optional parameter, *indicator*, can be specified as a number 01 through 99. When this parameter is used and the comparison passes, the specified indicator is turned on. When the comparison fails, the indicator is turned off.

When an indicator is not specified, comparison failures cause an error condition on the system. When an indicator is specified, setting of the error condition is suspended. Comparisons on character data are performed using the EBCDIC collating sequence (see Appendix A). Comparisons on numeric data are algebraic.



DSPATR(*attributes*)

The DSPATR keyword allows you to control display characteristics for each field via one or more attributes. Use of this keyword requires a display position before and immediately following the field. However, when adjacently displayed fields are controlled by display attributes, they can share a single position between them.

The attributes that can be specified are:

Attribute	Meaning
BL	Causes the displayed data to blink (flash on and off).
CA	Cancels display attributes propagated from the file description and record description statements. The CA parameter can be used when an attribute or characteristic specified in either the file description and record description statement is not desirable. Other parameters can then be used to control the display for the field. The parameter CA affects only the field for which it is coded.
CS	Causes column separators between character positions. Column separators are thin vertical lines, which are displayed between character positions and do not reduce display capacity.
HI	Causes the displayed characters to be highlighted (displayed with increased intensity).
ND	The field is not displayed.
RI	Causes the characters in the field to be displayed with image reversed (dark characters are displayed on light backgrounds).
UL	Causes each character position in the field to be underlined or underscored.

The combination of attributes HI, RI, and UL is invalid. If this combination is specified, or if this combination is formed by the display attributes specified in file description statement, record description statement, and this statement, the field is not displayed.

Note: When a display attribute is used in a prompt line or an overlaying format, be careful because the display attributes of a field require an additional position before and immediately following the field. If the field with display attributes begins in column 1, the display attributes require the use of column 80 in the previous line.

EDTCDE(*code 'float'*)

The EDTCDE keyword allows you to specify editing (punctuation) to be applied to data in numeric fields. The use of this keyword is limited to output operations that are explicitly coded with calculation statements.

The required parameter, *code*, must be a single character, which represents a set of editing characteristics. The following chart contains the valid entries for *code* and the associated edit characteristics.

Edit Code	Commas	Decimal Point	Sign for Negative Balance			Zero Suppress	Zero Balance to Print as:
			No Sign	CR	-(Minus)		
1	Yes	Yes	No sign			Yes	Zero
2	Yes	Yes	No sign			Yes	Blank
3		Yes	No sign			Yes	Zero
4		Yes	No sign			Yes	Blank
A	Yes	Yes		CR		Yes	Zero
B	Yes	Yes		CR		Yes	Blank
C		Yes		CR		Yes	Zero
D		Yes		CR		Yes	Blank
J	Yes	Yes			–	Yes	Zero
K	Yes	Yes			–	Yes	Blank
L		Yes			–	Yes	Zero
M		Yes			–	Yes	Blank
X[1]							
Y[2]						Yes	
Z[3]						Yes	Blank

[1]The X edit code removes the plus sign from the field for positive values. Because the system does this for you, normally you do not have to specify this code.

[2]The Y edit code suppresses the leftmost zero of a date field that is three to six digits long, and it suppresses the two leftmost zeros of a field that is seven positions long. The Y edit code also inserts slashes (/) between the month, day, and year according to the following pattern:

nn/n
nn/nn
nn/nn/n
nnn/nn/nn

[3]The Z edit code removes the sign (plus or minus) from a numeric field and suppresses leading zeros of a numeric field.

The float parameter is optional and can be used to specify that a floating currency symbol is desired or that the character positions to the left of the first significant digit are protected by asterisks. The entry must be enclosed in apostrophes. The valid entries for float are a currency symbol and *.

Use a currency symbol to specify a floating currency symbol, which places the currency symbol to the left of the first significant digit. The characters entered here (such as the dollar sign) must be the same as the character specified in the EDITC keyword on the job statement.

Use an asterisk (*) to specify that positions to the left of the first significant digit are to be filled with asterisks.

The characters actually used during an EDTCDE operation are determined by the parameters for the EDITC keyword in the job specification statement.

Note: When the currency symbol is specified, it requires two positions preceding the field when the record is printed. These two positions must be considered when the *Location* field (columns 39 through 44) is coded.

The following chart contains examples of the results of EDTCDE operations on various fields. The default values for the EDITC keyword are assumed.

Edit Codes	Positive Number– Two Decimal Positions	Positive Number– No Decimal Positions	Negative Number– Three Decimal Positions[1]	Negative Number– No Decimal Positions[1]	Zero Balance– Two Decimal Positions	Zero Balance– No Decimal Positions
Unedited	1234567	1234567	00012ƀ	00012ƀ	000000	000000
1	12,345.67	1,234,567	0.120	120	0.00	0
2	12,345.67	1,234,567	0.120	120		
3	12345.67	1234567	0.120	120	0.00	0
4	12345.67	1234567	0.120	120		
A	12,345.67	1,234,567	0.120CR	120CR	0.00	0
B	12,345.67	1,234,567	0.120CR	120CR		
C	12345.67	1234567	0.120CR	120CR	0.00	0
D	12345.67	1234567	0.120CR	120CR		
J	12,345.67	1,234,567	0.120–	120–	0.00	0
K	12,345,67	1,234,567	0.120–	120–		
L	12345.67	1234567	0.120–	120–	0.00	0
M	12345.67	1234567	0.120–	120–		
X[2]	1234567	1234567	00012ƀ	00012ƀ	000000	000000
Y[3]			0/01/20	0/01/20	0/00/00	0/00/00
Z[4]	1234567	1234567	120	120		

[1]The ƀ represents a blank. This may occur if a negative zero does not correspond to a printable character.

[2]The X edit code removes the plus sign from the field for positive values. Because the system does this for you, normally you do not have to specify this code.

[3]The Y edit code suppresses the leftmost zeros of a date field that is three to six digits long and it suppresses the two leftmost zeros of a field that is seven positions long. The Y edit code also inserts slashes (/) between the month, day, and year according to the following pattern:

nn/n
nn/nn
nn/nn/nn
nnn/nn/nn

[4]The Z edit code removes the sign (plus or minus) from a numeric field and suppresses leading zeros of a numeric field.

ERROR(*code 'message'*)

The ERROR keyword produces conditions similar to those associated with edit errors, which require operator action to resume the data entry process. When the ERROR keyword is used in a secondary line in conjunction with the *Indicator* field, you can indicate invalid conditions that are detected by your program logic by locking the keyboard, posting an error code on the status line, and flashing the status line. When the indicator specified in the *Indicator* field is on, the message is displayed.

The first parameter, *code*, is required and must be a two-digit number from 01 through 99. This number is displayed as the error code in the form 98XX, with XX the value entered for code.

The second parameter, which is optional, specifies a message, which is displayed on the status line when the operator presses the Help key while the station is in the error condition. The message parameter can be a character string of up to 39 character positions (enclosed in apostrophes) to provide operator guidance or to further define the error.

EXSR(*subroutine*)

The EXSR keyword provides branching to a named subroutine, which is coded with calculation statements on the C specifications.

The subroutine parameter is the name of the calculation subroutine. A subroutine is named by the statement that contains the BEGSR operation code.

The specified subroutine cannot perform any input/output operations to the keyboard/display file nor can it open any files because the keyboard/display is used when a file is opened.

INSERT(*source*)

INSERT(*expression*)

INSERT(*constant*)

The INSERT keyword allows the program to supply the data for the current field. The data can be the contents of another field, the result of an arithmetic expression, or a constant.

The source parameter can be the name of any named field. The data from a source can be either character data or numeric data, but it must match the field being defined.

Character data from a named field is left adjusted. When the current field is longer than the source field, the current field is padded to the right with blanks. When the current field is shorter than the source, the rightmost characters of the source are dropped.

Numeric data from a named field is right adjusted, decimal aligned, and rounded up as necessary. If the current field is longer than the source field, the current field is padded with zeros. The sign of the numeric data is also carried into the current field.

Production statistic counters can be used as a source of numeric data. (See *Production Statistics* in Chapter 9 for more information about these counters.) The reserved words *TOTn and UDATE can also be used as a data source. The contents of these reserved words is treated as numeric data with no positions to the right of the decimal.

Within an arithmetic expression parameter, addition, subtraction, multiplication, and division can be used in any sequence. The arithmetic expression is evaluated from left to right with multiplication and division preceding addition and subtraction. The valid symbols for the arithmetic operations are +, –, *, and /. The values in the expression can be specified as numeric constants and by names of defined numeric fields, including the special counters, *TOT1 through *TOT9, which are, in this application, 15 position numeric fields with no decimal positions. The use of parentheses and blanks within the expression is not allowed. The result of the evaluation of the arithmetic expression becomes the contents of the current field with decimal alignment and signs. With decimal alignment, truncation can occur on the left, and rounding can occur on the right.

In addition, the parameter for INSERT can be fixed data or a constant. The constant can be either numeric or character data. Numeric constants must have numeric fields as destinations. Character constants must have character fields as destinations.

A character constant is copied to the current field left adjusted. A numeric constant is inserted with decimal alignment. A decimal point in a numeric constant does not become part of the data in the current field.

IBM International Business Machines Corporation — **IBM 5280 DATA DESCRIPTION SPECIFICATIONS** — Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic	Source Document	Page of
Operator	Date		Key		

A

Checks=CHECK (code . . .): Auto Dup =AD, Auto Skip =AS, Blank Check =BC, Bypass =BY, Bypass on Verify =BV, Data Required =DR, Dup Disable =DD, Field Exit Required =FE, Lower Case LC, Mandatory Entry =ME, Mandatory Fill =MF, Rt Adj–Blank Fill =RB, Right to Left =RL, Rt Adj–Zero Fill =RZ, Self Check =nxx (n=C/G (Check/Gen), xx=Modulus)

Functions: ADD (name), AUXDUP (name), AUXST (name), COMP (¹test fld1 ³@ fldn [indicator]), DSPATR (²attr), EDTCDE (code float), ERROR (code ['message']), EXSR (subroutine), INSERT (fld1 ³@ fldn), LOOK (table [index]), PMT (prompt), RANGE (low high), RANGET (table [index]), RESET (•TOTn), SEQ (¹test), SETOF (ind), SETON (ind), SHIFT (⁴shift), SUB (name), SUBST (table1 table2 [index]), TADD (•TOTn), TSUB (•TOTn), XCHK (table index1 index2), 'literal'

¹test: EQ,GE,GT,LE,LT,NE
²attr=BL,CA,CS,HI,ND,RI,UL
³@=+,–,•,/
⁴shift=A,D,H,N,V,W,X,Y

Sequence (1–5)	Form Type (6)	Dataset/Record/Field/Table Name (19–26)	Length (30–34)	Decimal Positions (37)	Usage (38)	Editing (45–62)	The value in SAMPLE after each operation.
1	A						
2	A	SAMPLE	10		I	INSERT('-123.456')	- 1 2 3 . 4 5 6 bb
3	A						
4	A						
5	A	SAMPLE	5		I	INSERT('-123.456')	- 1 2 3 .
6	A						
7	A						
8	A	SAMPLE	10	5	I	INSERT(-123.456)	0 0 1 2 3 4 5 6 0 $\bar{0}$
9	A						
10	A						
11	A	SAMPLE	4	2	I	INSERT(-123.456)	2 3 4 $\bar{6}$
12	A						
13	A	SAMPLE	1	1	I	INSERT(.0)	0
14	A						
15	A						

The sign of the value in a numeric field occupies the zone portion of the rightmost digit in the field.

The arrow (↓) indicates the position of the decimal point.

*Number of sheets per pad may vary slightly.

LOOK(*tablename indexname*)

The LOOK keyword allows you to check the data in the field for an equal match against entries in a table.

The required parameter, *tablename*, must be the name of a table containing the entries against which the data in the field is compared. Those entries must have the same length as the field being compared. When an equal comparison exists, processing continues. When no match exists, an error is indicated.

The indexname parameter is optional. When this parameter is used, it must be the name of a numeric field with no positions to the right of the decimal. The field may be described as a workspace in a field description statement. If it is not described, a length of five positions with no decimal positions is assigned by the compiler. After each LOOK operation, the number of the table entry that matches the data in the field is stored in the field named in *indexname*. When no match exists, the field is set to 1. The name used cannot exceed six character positions, and the indexname parameter must be the second parameter.

IBM International Business Machines Corporation — IBM 5280 DATA DESCRIPTION SPECIFICATIONS — Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic	Source Document	Page of
Operator	Date		Key		

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	A																
2	A							ITEMNO		4	N		0	I			LOOK(SAMPLE POCKET)
3	A							POCKET		2			0	W			
4	A																
5																	
6																	
7																	
8	A																
9	A																
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

Checks=CHECK (code . . .)

Auto Dup	AD	Mandatory Entry	ME
Auto Skip	AS	Mandatory Fill	MF
Blank Check	BC	Rt Adj, Blank Fill	RB
Bypass	BY	Right to Left	RL
Bypass on Verify	BV	Rt Adj, Zero Fill	RZ
Data Required	DR	Self Check	mxx
Dup Disable	DD	n C/G (Check/Gen)	
Field Exit Required	FE	xx-Modulus	
Lower Case	LC		

Functions

ADD (name)
AUXDUP (name)
AUXST (name)
COMP (*test fld1 *fld2 fldn [indicator])
DSPATR (*attr . . .)
EDTCDE (code [fcat])
ERROR (code [,message])
EXSR (subroutine)
INSERT (fld1 *fld2 . . . fldn)
LOOK (table [,index])
*test-EQ,GE,GT,LE,LT,NE
*attr-BL,CA,CS,HI,ND,RI,UL
*fld=+, -, */
*shift-A,D,H,N,V,W,X,Y
PMT (*prompt)
RANGE (low,high)
RANGE (table [,index])
RESET (*TOTn)
SEQ (*test)
SETOF (ind)
SETON (ind)
SHIFT (*shift)
SUB (name)
SUBST (table1 table2 [,index])
TADD (*TOTn)
TSUB (*TOTn)
XCHK (table,index1,index2)
*literal

SAMPLE is a table containing the values shown. If the operator enters 0017 in the field named ITEMNO, the position of 0017 in SAMPLE is placed in the index named POCKET.

SAMPLE
0015
0016
0017
0023
0114
2379
6328

POCKET: 03

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

PMT(*message*)

The PMT keyword allows you to display a prompting message while a field is available to receive data. The prompting message is displayed on the second physical display line starting in position 1 and continues on the following lines depending on message length.

The required parameter, *message*, can be any message with a maximum length of 200 character positions. Any displayable character and spaces are valid in the message. If the message includes right parentheses, apostrophes, or is all spaces, the entire message must be coded as a character constant, which must be enclosed in apostrophes. An apostrophe must then be represented by two apostrophes. Otherwise, apostrophes are not required. If the message is not coded as a character constant, heading and/or trailing spaces are ignored.

Note: The prompt message is cleared when the next field is entered.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence | Form Type | Comment (*) | Reserved | Indicator (for CHECK (BY, BV) or ERROR) | Reserved | Name Type (F/K/R/T) | Reserved | Dataset/Record/Field/Table Name | Reserved | Length | Data Type | Reserved | Decimal Positions (0-9) | Usage (I/O/B/W) | Location: Line, Pos | Editing

Checks=CHECK (code . . .)

Auto Dup	:AD	Mandatory Entry	:ME
Auto Skip	:AS	Mandatory Fill	:MF
Blank Check	:BC	Rt Adj–Blank Fill	:RB
Bypass	:BY	Right to Left	:RL
Bypass on Verify	:BV	Rt Adj–Zero Fill	:RZ
Data Required	:DR	Self Check	:nxx
Dup Disable	:DD	n=C/G (Check/Gen)	
Field Exit Required	:FE	xx=Modulus	
Lower Case	:LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low high)
AUXST (name)	RANGET (table [index,])
COMP (²test fld1¹@ fld2 [indicator])	RESET (+TOTn)
DSPATR (²attr . . .)	SEQ (¹test)
EDTCDE (code float)	SETOF (ind)
ERROR (code ['message'])	SETON (ind)
EXSR (subroutine)	SHIFT (⁴shift)
INSERT (fld1¹@ fld2)	SUB (name)
LOOK (table [index,])	SUBST (table1 table2 [index1])
	TADD (+TOTn)
¹test=EQ,GE,GT,LE,LT,NE	TSUB (+TOTn)
²attr=BL,CA,CS,HI,ND,RI,UL	XCHK (table index1 index2)
³@=+,-,*,/	literal
⁴shift=A,D,H,N,V,W,X,Y	

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Line	Pos	Editing
1	A										1A			I.			PMT('THIS EXAMPLE SHOWS THE USE OF +
2	A																APOSTROPHES (''') TO ENCLOSE PROMPTI+
3	A																NG MESSAGES.')
4	A																
5	A																
6																	THIS EXAMPLE SHOWS THE USE OF APOSTROPHES (') TO ENCLOSE PROMPTING MESSAGES.
7																	
8																	
9																	
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

The prompt as displayed on line 2.

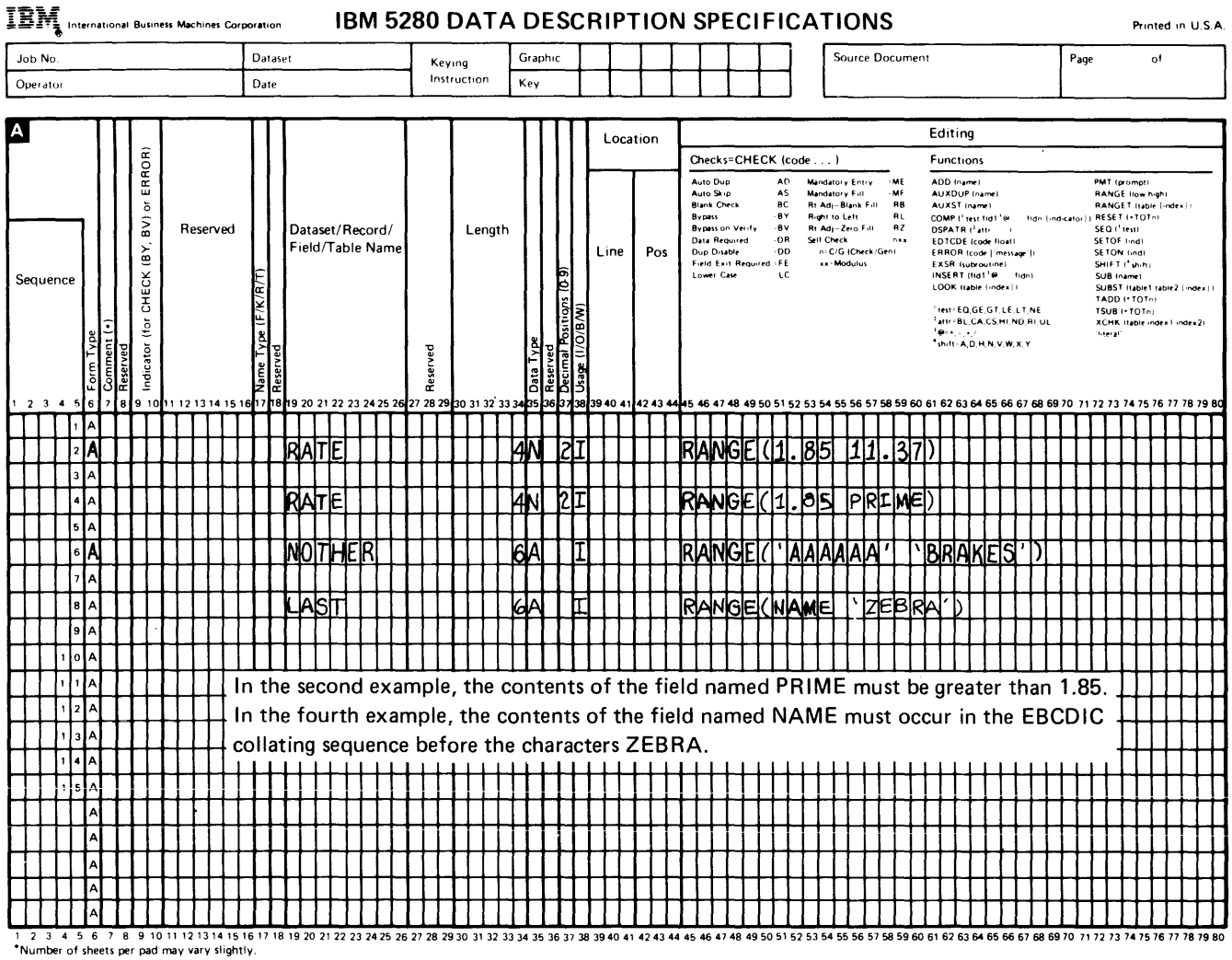
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

RANGE(*low high*)

The RANGE keyword allows the specification of high and low limit values for comparison with the data entered in the field. The data must be greater than or equal to the low-limit value and less than or equal to the high-limit value. If the data value is outside the specified range, an error condition is indicated and an error code is displayed in the status line. When the data falls within the range, processing continues.

Both parameters are required. The low and high values of the range must be specified in the order indicated. Each parameter value may be entered as a numeric constant enclosed in apostrophes, a character constant, or the name of either a numeric field or a character field containing the value. The EBCDIC collating sequence is used during range checking of character data. Numeric data is decimal aligned, then algebraically compared.



RANGET(*tablename indexname*)

The RANGET keyword allows you to check the data in the field against ranges of values contained in a table.

Note: In DE/RPG, a table is the same as an array in RPG.

The required parameter, *tablename*, must be the name of a table that contains pairs of entries which constitute the low and high limits of ranges. After the data for the field is entered, the data is tested against the ranges formed by the pairs of table entries. When the data is within one of the ranges, processing continues normally. When a range that includes the value of the data is not found, an error is indicated.

The indexname parameter is optional. When this parameter is used, it must name a numeric field with no positions to the right of the decimal. The numeric field may be described in a field description statement. If the field is not described, the compiler assigns a length of five positions for the field (with no decimal positions).

The EBCDIC collating sequence is used during the range checking for both character and numeric data. After each RANGET operation, the number of the pair of table entries that formed the range within which the entered data was found is stored in the field named in *indexname*. When the field data does not fit one of the ranges, the variable is set to 1.

The indexname parameter must be the second parameter.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location Line	Location Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	A																
2	A							QUANT		3			0	I			RANGET(TQUANT RANGE)
3	A							RANGE		2			0	W			
4	A																

Checks=CHECK (code . . .)

Auto Dup	AD	Mandatory Entry	ME
Auto Skip	AS	Mandatory Fill	MF
Blank Check	BC	Rt Adj, Blank Fill	RB
Bypass	BY	Right to Left	RL
Bypass on Verify	BV	Rt Adj, Zero Fill	RZ
Data Required	DR	Self Check	nnn
Dup Disable	DD	n, C, G (Check Gen)	
Field Exit Required	FE	xx, Modulus	
Lower Case	LC		

Functions

ADD (name)
AUXDUP (name)
AUXST (name)
COMP (test high low indicator)
DSPATR (attr . . .)
EDTCDE (code float)
ERROR (code message)
EXSR (subroutine)
INSERT (high low)
LOOK (table index high low)
(test: EQ GE GT LE LT NE
attr: BL CA CS HI ND RI UL
*on . . .
*shift: A D H N V W X Y)
PMT (prompt)
RANGE (low high)
RANGET (table index)
RESET (*TOT)
SEQ (test)
SETOF (ind)
SETON (ind)
SHIFT (*shift)
SUB (name)
SUBST (table1 table2 index)
TADD (*TOT)
TSUB (*TOT)
XCHK (table index1 index2)
(literal)

TQUANT is a table containing the values shown below. Each pair of values represents the low and high values of a range.

If the operator enters 075 in the field named QUANT, the value is found in the third range in TQUANT, and 03 is placed in the index named RANGE.

If the operator enters 079 in the field named QUANT, the value is not found in any range in TQUANT, 01 is placed in the index named RANGE, and an error is indicated on the status line (with the keyboard locked).

1	5	A
		A
		A
		A
		A
		A

1 2 3 4 5 6 7 8 9 10 11
*Number of sheets per pad ma

TQUANT

001
016
030
059
061
078
093
225
300
779

4 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

RESET(*name*)

The RESET keyword places zeros in all positions of the named field.

The required parameter name, must identify one of the counters (*TOT1 through *TOT9) or a numeric field.

SEQ(*test*)

The keyword SEQ allows the specification of a sequence check of the data in the current field against the data in the field previously sequence checked. The field to be checked should not be longer than 16 positions. If it is, only the first 16 positions are checked.

The required parameter, *test*, must be chosen from the following:

Parameter	Meaning
EQ	The current data must be equal to the previous data.
NE	The current data must not be equal to the previous data.
GE	The current data must be greater than or equal to the previous data.
LE	The current data must be less than or equal to the previous data.
GT	The current data must be greater than the previous data.
LT	The current data must be less than the previous data.

The test is conducted after the data is fully entered. The data is tested using the EBCDIC collating sequence (see *Appendix A*).

When a test is completed, the current data replaces the previous data as a reference for the next sequence check. When a test fails, the system posts an error message. The first time a sequence check is performed, there is no previous data for a comparison and the test always passes.

SETOF(*indicator*)

The keyword SETOF allows you to turn off any one of the indicators (01 through 99).

The single required parameter, *indicator*, must be the number of the indicator (01 through 99).

SETON(*indicator*)

The keyword SETON allows you to turn on any one of the indicators (01 through 99).

The single required parameter, *indicator*, must be the number of the indicator (01 through 99).

SHIFT(*codes*)

The SHIFT keyword is required in field description statements in which the letter C is specified for *Data Type*. This keyword allows you to program the keyboard conditioning for each position in the field.

The required parameter, *codes*, is a string of characters, one for each character position in the field. The following characters are valid.

Character	Meaning
A	Alphabetic shift – Any character can be entered. The shift is positioned to the lower symbol on each key on all keyboards. The operator can use the Shift key to enter the upper symbol on the keys.
D	Digits only – Only the numbers 0 through 9 can be entered. The shift is positioned to the lower symbols on typewriter keyboards and is positioned to the upper symbols on both data entry keyboards and proof keyboards. Negative numbers are displayed with the sign over the number. The operator cannot override the shift.
H	Hexadecimal – Each character position requires two keystrokes and only the numbers 0 through 9 and the letters A through F can be entered. No shift key operation is required on data-entry keyboards.
N	Numeric shift – Any character can be entered. The shift is positioned to the lower symbols on typewriter keyboards and is positioned to the upper symbols on both the data entry keyboards and the proof keyboards. The operator can change the shift by using the Alphabetic Shift key.
V	Right half only – Only the characters that are defined at system generation can be entered. The shift is positioned to the lower right symbols on World Trade keyboards. The shift key allows entry of the upper right symbols.
W	Right half shift – Any character can be entered. The shift is positioned to the lower right symbols on the World Trade keyboards. The operator can change the shift by using the shift keys.
X	Alphabetic only – Only the letters A through Z, comma, period, hyphen, and space can be entered. The shift is positioned to the lower symbols on all keyboards. The shift cannot be changed by the operator.
Y	Numeric only – Only the numbers 0 through 9, comma, period, plus, minus, and space can be entered. The shift is positioned to the lower symbols on typewriter keyboards and to the upper symbols on both the data entry keyboards and the proof keyboards. The shift cannot be changed by the operator.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing: Checks=CHECK (code . . .)	Editing: Functions
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34 35	36	37	38	39	40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	A																	
2	A							CODE		6	C		I				SHIFT(XXDDDH)	
3	A																	
4	A																	
5	A																	
6	A																	
7	A																	
8	A																	
9	A																	
10	A																	
11	A																	
12	A																	
13	A																	
14	A																	
15	A																	
	A																	
	A																	
	A																	
	A																	
	A																	

Checks=CHECK (code . . .)

Auto Dup	AD	Mandatory Entry	ME
Auto Skip	AS	Mandatory Fill	MF
Blank Check	BC	Rt Adj–Blank Fill	RB
Bypass	BY	Right to Left	RL
Bypass on Verify	BV	Rt Adj–Zero Fill	RZ
Data Required	DR	Self Check	nxx
Dup Disable	DD	n: C/G (Check/Gen)	
Field Exit Required	FE	xx: Modulus	
Lower Case	LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low high)
AUXST (name)	RANGET (table [index])
COMP (1 test fld1 1 @ fldn [indicator])	RESET (+TOTn)
DSPATR (2 attr)	SEQ (1 test)
EDTCDE (code float)	SETOF (ind)
ERROR (code ['message'])	SETON (ind)
EXSR (subroutine)	SHIFT (4 shift)
INSERT (fld1 1 @ fldn)	SUB (name)
LOOK (table [index])	SUBST (table1 table2 [index])
	TADD (+TOTn)
1 test: EQ,GE,GT,LE,LT,NE	TSUB (+TOTn)
2 attr: BL,CA,CS,HI,ND,RI,UL	XCHK (table index1 index2)
3 @: +, –, *, /	'literal'
4 shift: A,D,H,N,V,W,X,Y	

When entered, the six-character field named CODE must contain alphabetic characters in the first two positions (XX), digits only in the next three positions (DDD), and a hexadecimal number in the last position (H).

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

SUB(*counter*)

The SUB keyword causes decimal aligned subtraction of the data in the current numeric field from a counter. The subtraction occurs only after the data entry for the field is completed.

The required parameter, *counter*, must be the name of a numeric field described in the program source.

SUBST(*tablename1 tablename2 indexname*)

The keyword SUBST uses two tables with corresponding entries. After the data is entered in the field, the first table is searched for an entry that matches (equal comparison) the data in the field. When the match is found, the data in the field is replaced by the corresponding entry from the second table.

The tablename1 and tablename2 parameters are required and must be the names of tables. The table named in *tablename1* must contain the entries for comparison to the data entered in the field. The table named in tablename2 contains the entries that are substituted for the data in the field. Tablename1 must be specified first.

The third parameter, *indexname*, is optional. When this parameter is specified, it must be the name of a numeric field. The numeric field may be described in a field description statement. If the field is not described, the compiler assigns a length of five positions to the field (no decimal positions). After each SUBST operation, the field named in indexname contains the number of the table entry that matched the data entered in the field, which is also the number of the entry substituted from the second table specified.

If a match between the data in the field and an entry in the first table does not exist, an error is indicated, and the variable is set to 1.

The length that you specify for the current field must be long enough to receive the substituted entry from the table named in the second parameter. The length of the original entry, which is used as the reference during the search of the table named by the first parameter, is determined by the length of the entries in the first table.

The length of the entries in the second table (tablename2) should be greater than, or equal to, the length of the entries in the first table (tablename1).

See *Editing Field*, Chapter 5 for the sequence of execution if SUBST is specified.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic	Source Document	Page of
Operator	Date		Key		

Editing — Checks=CHECK (code . . .)

Auto Dup =AD, Auto Skip =AS, Blank Check =BC, Bypass =BY, Bypass on Verify =BV, Data Required =DR, Dup Disable =DD, Field Exit Required =FE, Lower Case =LC, Mandatory Entry =ME, Mandatory Fill =MF, Rt Adj–Blank Fill =RB, Right to Left =RL, Rt Adj–Zero Fill =RZ, Self Check =nxx (n=C/G (Check/Gen), xx=Modulus)

Functions

ADD (name), AUXDUP (name), AUXST (name), COMP (¹test fld1 ³@ fldn [indicator]), DSPATR (²attr), EDTCDE (code float), ERROR (code ['message']), EXSR (subroutine), INSERT (fld1 ³@ fldn), LOOK (table [index]), PMT (prompt), RANGE (low high), RANGET (table [index]), RESET (*TOTn), SEQ (¹test), SETOF (ind), SETON (ind), SHIFT (⁴shift), SUB (name), SUBST (table1 table2 [index]), TADD (*TOTn), TSUB (*TOTn), XCHK (table index1 index2), 'literal'

¹test=EQ,GE,GT,LE,LT,NE
²attr=BL,CA,CS,HI,ND,RI,UL
³@=+,–,*,/
⁴shift=A,D,H,N,V,W,X,Y

Sequence (1–5)	Form Type (6)	Dataset/Record/Field/Table Name (19–26)	Length (30–34)	Data Type (35)	Decimal Positions (37)	Usage (I/O/B/W) (38)	Editing (45–80)
1	A						
2	A	EMPNAM	25	N		I	PMT(ENTER THE 4-DIGIT EMPLOYEE NUMB+
3	A						ER) SUBST(NOTAB NAMTAB INDEX)
4	A	INDEX	2		0	W	

The tables, NOTAB and NAMTAB contain the following entries.

NOTAB	NAMTAB
1000	WALTER JAMES ABRAHMS
1001	EDWART TIMOTHY BIRGSTRESS
1002	OREM CASH
1003	FREDRICK MOSS HAINES
1005	MARYJO KASTEM
1009	EDITH FAY MURRAY

If the operator enters 1003 and presses the Field Exit key, the described field, EMPNAM, and the workspace, INDEX, contain the following values.

EMPNAM FREDRICK MOSS HAINES

INDEX 04

*Number of sheets pe

TADD(*counter*)

The TADD keyword adds the contents of the current field into a counter. The contents of the current field is considered a string of numbers, and no decimal alignment is performed. The addition is performed only after the entry of data into the field is completed.

The required parameter, *counter*, must be specified as *TOT*n* with *n* equal to 1 through 9.

In enter mode, backspace operations into or past the field cause the amount added into the counter to be subtracted from the counter contents.

In update or verify mode, backspace operations do not affect the counter contents. The contents of the counter are updated by adding the difference between the initial and final values of the field when the field is reprocessed in the forward direction.

TSUB(*counter*)

The TSUB keyword subtracts the contents of the current field from a counter. The contents of the current field is considered a string of numbers, and no decimal alignment is performed. The subtraction is performed only after the entry of data into the field is completed.

The required parameter, *counter*, must be specified as *TOT*n* with *n* equal to 1 through 9.

In enter mode, backspace operations into or past the field cause the amount subtracted from the counter to be added into the counter contents.

In update or verify mode, backspace operations do not affect the counter contents. The contents of the counter are updated by subtracting the difference between the initial and final values of the field when the field is reprocessed in the forward direction.

XCHK(*tablename indexname1 indexname2*)

The XCHK keyword allows you to compare the values contained in a pair of named fields to pairs of entries in a table. The named fields could contain the index values from previously completed LOOK or RANGET operations that included the optional indexname parameters or be any other variable names. Therefore, the XCHK keyword allows compatibility checking of data entered in separate fields.

Three parameters are required (*tablename*, *indexname1*, and *indexname2*) and these parameters must be specified in the sequence indicated.

The tablename parameter must be the name of a table. Each table entry must contain a value. The entries are used in pairs.

The indexname1 and indexname2 parameters must be the names of numeric fields. The fields can be the fields used as the indexes for previously completed table operations. If the field is not described elsewhere in the program, the compiler assigns a length of five positions to the field (zero decimal positions). The lengths assigned to the fields should be equal to each other and to the length of the table entries.

During the execution of the XCHK function, the current values in *indexname1* and *indexname2* are used as masks in a search of the named table. When a pair of table entries that contains both values is found, processing continues. If no match is found, an error is indicated.

Job No	Dataset	Keying Instruction	Graphic / Key	Source Document	Page of
Operator	Date				

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing: Checks-CHECK (code . . .)	Editing: Functions
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80	
1	A																	
2	A							COLOUT		3	A			I			PMT(ENTER BODY COLOR)	
3	A																LOOK(COL OUT)	
4	A							COLIN		3	A			I			PMT(ENTER INTERIOR COLOR)	
5	A																LOOK(COL IN)	
6	A																XCHK(MATCH OUT IN)	
7	A							IN		1			0	W				
8	A							OUT		1			0	W				

1 2 3 4
*Number o

This example of an XCHK operation uses two tables, COL and MATCH. COL is searched for a matching entry for each field (LOOK operation). The number of the entry that matches the entered data is placed in the specified workspace, which serves as an index field. Then, MATCH is searched for a pair of entries that match the contents of the two index fields.

COL

RED
BLK
ORG
WHT

With the table contents as shown, an orange interior is available only with an orange body. Also a red interior is not available with an orange body.

MATCH	Equivalent Color Pairs
1 1	Red/Red
1 2	Red/Black
1 4	Red/White
2 1	Black/Red
2 2	Black/Black
2 4	Black/White
3 2	Orange/Black
3 3	Orange/Orange
3 4	Orange/White
4 1	White/Red
4 2	White/Black
4 4	White/White

Keyword Conflicts and Compatibilities

Many keywords for field description statements cannot be used in combination with certain entries in the *Usage* field, certain entries in the *Data Type* field, or certain other keywords and parameters. If *RTN is specified in the *Name* field, the only keywords that are valid in the *Editing* field are RESET, SETON, SETOF, and EXSR. The other valid and invalid combinations are indicated in the following charts.

The Xs represent invalid combinations. For example, CHECK(DR) cannot be used for output fields or workspaces (coded with O or W in the *Usage* field). Also notice that combining DR (data required) and BY (bypass) as parameters for CHECK is invalid. Also, DR cannot be specified twice for the same field.

Check Parameters	Usage Field Entry				Data Type Field Entry									
	I	O	W	B	A	C	D	H	N	S	W	X	Y	V
LC	.	X	X	.	.	.	.	.	.	.	.	.	.	.
AD	.	X	X	.	.	.	.	.	.	.	.	.	.	.
AS	.	X	X	.	.	.	.	.	.	.	.	.	.	.
BC	.	X	X	.	.	.	.	.	.	.	.	.	.	.
BY	.	X	X	.	X	X	X	X	X	X	X	X	X	X
BY-I	.	X	X	.	.	.	.	.	.	.	.	.	.	.
BV	.	X	X	.	.	.	.	.	.	.	.	.	.	.
DR	.	X	X	.	.	.	.	.	.	.	.	.	.	.
FE	.	X	X	.	.	.	.	.	.	.	.	.	.	.
ME	.	X	X	.	.	.	.	.	.	.	.	.	.	.
MF	.	X	X	.	.	.	.	.	.	X	.	.	.	.
RB	.	X	X	.	.	X	.	.	.	.	.	.	.	.
RZ	.	X	X	.	.	X	.	.	.	.	.	.	.	.
Gxx	.	X	X	.	.	.	.	.	.	.	.	.	.	.
Mxx	.	X	X	.	.	.	.	.	.	.	.	.	.	.
RL	.	X	X	.	X	X	X	X	X	X	.	X	X	.

Check Parameters	Check Parameters															
	LC	AD	AS	BC	BY	BY-I	BV	DR	FE	ME	MF	RB	RZ	Gxx	Mxx	RL
LC	X	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.
AD	.	X	X	.	X	.	.	.	.	.	.	.	.	.	.	.
AS	.	X	X	.	X	.	.	.	.	.	.	.	.	.	.	.
BC	.	.	.	X	X	.	.	.	.	.	.	.	.	.	.	.
BY	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BY-I	.	.	.	.	X	X	.	.	.	.	.	.	.	.	.	.
BV	.	.	.	.	X	.	X	.	.	.	.	.	.	.	.	.
DR	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.
FE	.	.	.	.	X	.	.	.	X	.	.	.	.	.	.	.
ME	.	.	.	.	X	.	.	.	.	X	.	.	.	.	.	.
MF	.	.	.	.	X	.	.	.	.	.	X	X	X	.	.	.
RB	.	.	.	.	X	.	.	.	.	.	X	X	X	.	.	X
RZ	.	.	.	.	X	.	.	.	.	.	X	X	X	.	.	X
Gxx	.	.	.	.	X	.	.	.	.	.	.	.	.	X	.	.
Mxx	.	.	.	.	X	.	.	.	.	.	.	.	.	.	X	.
RL	.	.	.	.	X	.	.	.	.	.	.	X	X	.	.	X

Check Parameters	Keywords																						
	ADD	AUXDUP	AUXST	COMP-I	COMP	DSPATR	EDTCDE	ERROR	EXSR	INSERT	LOOK	SETON/SETOF	PMT	RANGE	PANGET	RESET	SEQ	SHIFT	SUB	SUBST	TADD	TSUB	XCHK
LC	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
AD	.	X	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	X	.	.	.
AS	.	X	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
BC	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
BY	X	X	X	X	X	.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BY-I	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
BV	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
DR	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
FE	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
ME	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
MF	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
RB	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
RZ	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
Gxx	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
Mxx	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
RL	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.

BY-I refers to CHECK(BY) used as the exclusive entry in Editing field on a secondary line with an indicator specified.

COMP-I refers to the use of COMP, the compare keyword, including the optional parameter to set an indicator when the comparison fails. Use of this parameter prevents the normal error condition caused by comparison failures.

X indicates an invalid combination.

Keywords	Usage Field Entry				Data Type Field Entry										Check Parameters																Keywords																						
	I	O	W	B	A	C	D	H	N	S	W	X	Y	V	LC	AD	AS	BC	BY	BY-I	BV	DR	FE	ME	MF	RB	RZ	Gxx	Mxx	RL	ADD	AUXDUP	AUXST	COMP-I	COMP	DSPATR	EDTCDE	ERROR	EXSR	INSERT	LOOK	SETON/SETOFF	PMT	RANGE	RANGET	RESET	SEQ	SHIFT	SUB	SUBST	TADD	TSUB	XCHK
ADD	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
AUXDUP	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	X	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	X	.	.	.	.	.
AUXST	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
COMP-I	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
COMP	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
DSPATR	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
EDTCDE	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ERROR	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
EXSR	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
INSERT	.	X	.	.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	.	X	.	X	X	.	X	X	.	X	X	.	.	X	X	.	X	X	.	X	.	.	X
LOOK	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
SETON/SETOFF	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
PMT	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.
RANGE	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	X	.	.	.	.	.	.	.	.	.
RANGET	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	X	.	.	.	.	.	.	.	.
RESET	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
SEQ	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	X	.	.	.	.	.	.
SHIFT	.	X	X	.	X	.	X	X	X	X	X	X	X	X	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	X	.	.	.	.	.
SUB	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
SUBST	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	X	X	X	.
TADD	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.
TSUB	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.
XCHK	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.

BY-I refers to CHECK(BY) used as the exclusive entry in Editing field on a secondary line with an indicator specified.

COMP-I refers to the use of COMP, the compare keyword, including the optional parameter to set an indicator when the comparison fails. Use of this parameter prevents the normal error condition caused by comparison failures.

X indicates an invalid combination.

TABLE DESCRIPTION STATEMENTS

Table description statements name the tables and specify the length of the entries in the tables. A table can contain either character entries or numeric entries, but not both. Data for a table can be supplied with the source program (compile-time table) or in a data file when the program is executed (execution-time table).

Table description statements must be preceded by file description statements for the files in which they reside. Files that contain tables can contain only tables, and all the tables in a single file must contain the same number of entries. If a file contains data for more than one table, each record in the file contains data for all the tables. For example, if a file contains data for the three tables A, B, and C, the records in the file would contain the table entries as follows:

Record 1	A1	B1	C1
Record 2	A2	B2	C2
Record n	An	Bn	Cn

Tables cannot be described as a part of the file to which the keyboard/display is assigned.

Note: The term *table* as used here refers to the function provided by arrays in RPG subroutines. The term *table entry* refers to an entry in a table and coincides with the term *array element* in RPG subroutines.

The fields that can be coded in table description statements are shown as unshaded fields in the following illustration.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/R/K/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
01	A					T											
02	A																
03	A																
04	A																
05	A																
06	A																
07	A																
08	A																
09	A																
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

Checks=CHECK (code . . .)

Auto Dup	=AD	Mandatory Entry	=ME
Auto Skip	=AS	Mandatory Fill	=MF
Blank Check	=BC	Rt Adj–Blank Fill	=RB
Bypass	=BY	Right to Left	=RL
Bypass on Verify	=BV	Rt Adj–Zero Fill	=RZ
Data Required	=DR	Self Check	=nxx
Dup Disable	=DD	D M/G (Check Gen)	
Field Exit Required	=FE	xx = Modulus	
Lower Case	=LC		

Functions

ADD (name)
AUXDUP (name)
AUXST (name)
COMP ('test fld1'\@ fldn 'literal' (indicator))
DSPATR ('attr . . .)
EDTCDE (code 'float')
ERROR (code ['message'])
EXSR (subroutine)
INSERT (fld1'\@ . . . fldn 'literal')
LOOK (table [index])
PMT (prompt)
RANGE (low high)
RANGET (table [index])
RESET ([*TOTn] [name])
SEQ ['test]
SETOF (ind)
SETON (ind)
SHIFT (*shift)
SUB (name)
SUBST (table1 table2 [index])
TADD ([*TOTn] [name])
TSUB ([*TOTn] [name])
XCHK (table index1 index2)
'literal'

'test=EQ,GE,GT,LE,LT,NE
'attr=BL,CA,CS,HI,ND,RI,UL
'@+-_*,/
*shift=A,D,H,N,V,W,X,Y

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

Sequence Number Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

Enter an A in the *Form Type* field.

Name Type Field (Column 17)

The letter T is required in the *Name Type* field for table description statements.

Name Field (Columns 19 through 26)

Every table must be named. Table names cannot start with TAB and cannot be more than six characters in length.

Length Field (Columns 30 through 34)

The *Length* field describes the length of an entry in the table. (All entries in a single table are the same length.) An entry in the length field is required.

The *Length* field can contain only numbers. The number must be right adjusted, and either leading blanks or leading zeros are allowed. A numeric table entry cannot exceed 15 positions. Character entries cannot exceed 256 positions.

Decimal Positions Field (Column 37)

The entry in the *decimal positions* field determines the nature (character or numeric) of the table entries.

Leave the decimal positions field blank for tables with character entries. Enter a number from 0 through 9 to indicate the number of positions to the right of the decimal point for numeric table entries.

Example

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	A																
2	A					T		ATABLE		6			2				
3	A																
4	A																
5	A																
6	A																
7	A																
8	A																
9	A																
10	A																
11	A																
12	A																
13	A																
14	A																
15	A																
	A																
	A																
	A																
	A																
	A																

Checks=CHECK (code . . .)

Auto Dup	AD	Mandatory Entry	ME
Auto Skip	AS	Mandatory Fill	MF
Blank Check	BC	Rt Adj Blank Fill	RB
Bypass	BY	Right to Left	RL
Bypass on Verify	BV	Rt Adj Zero Fill	RZ
Data Required	DR	Self Check	nnn
Dup Disallw	DD	n C G (Check Germ)	
Field Exit Required	FE	xx Modulus	
Lower Case	LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low high)
AUXST (name)	RANGET (table [index])
COMP ('test fld1 'tst fld2 [ind or indicator])	RESET (*TOT#)
DSPATR ('attr [ind])	SEQ ('test)
EDTCDE (code [foat])	SETOF (ind)
ERROR (code [message])	SETON (ind)
EXSR (subroutine)	SHIFT ('shift)
INSERT (fld1 'fld2 [ind])	SUB (name)
LOOK (table [index])	SUBST (table1 table2 [index])
	TADD (*TOT#)
'test: EQ,GE,GT,LE,LT,NE	TSUB (*TOT#)
'attr: BL,CA,CS,HI,ND,RI,UL	XCHK (table index1 index2)
'fld2: *, -,	'literal'
'shift: A,D,H,N,V,W,X,Y	

This statement names a table ATABLE. This table contains six-position numeric entries with two decimal positions. The number of entries in this table is described by the NUMENT keyword or the file description statement to which this statement is subordinate.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

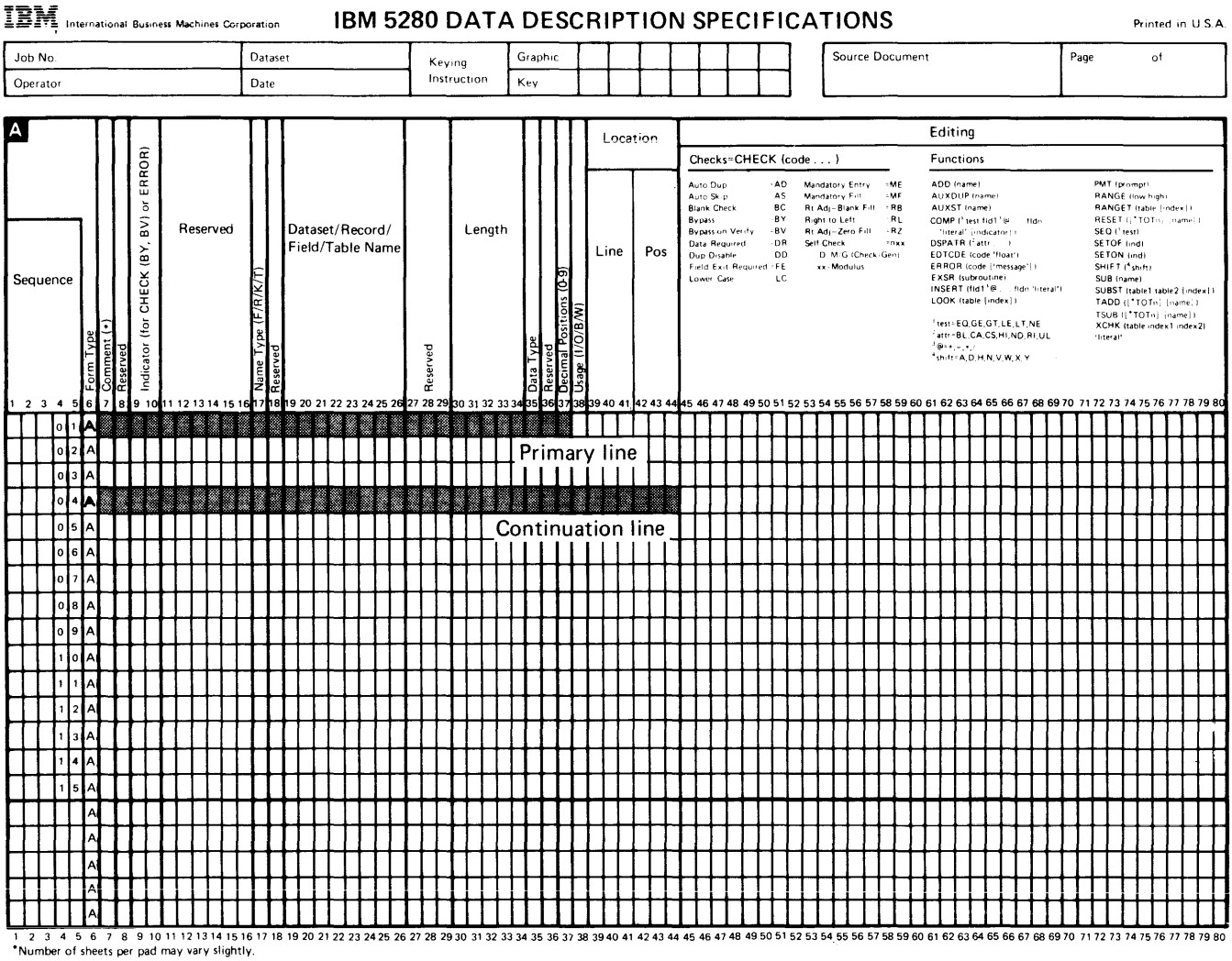
*Number of sheets per pad may vary slightly.

LITERAL STATEMENTS

Literal statements are the means for specifying and positioning character constant data for output to any file except the transaction file. Like field description statements, all literal statements must be preceded by a record description statement.

The fixed prompts resulting from literal statements do not become part of a diskette record.

The fields that can be coded for literal statements are shown as unshaded fields in the following illustration.



*Number of sheets per pad may vary slightly.

Sequence Number Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

Enter an A in the *Form Type* field.

Usage Field (Column 38)

The *Usage* field applies only to statements associated with the file to which the keyboard/display is assigned. Literal statements require the letter O in the *Usage* field.

Location Field (Columns 39 through 44)

The entry in the *Location* field specifies the starting (leftmost) position for the placement of data for the associated device. If the device is the keyboard/display both the *Line* field and the *Position* field apply. If the device is other than the keyboard/display, the *Location* field applies, and the entry in the *Location* field represents the starting position of the field.

An entry in the *Line* field specifies a display line number that is based on the starting line number being used for reference. (The starting line number is the value entered as the parameter for the SLNO keyword in entry format statements.) If the starting line number specified is 4 and the *Line* field entry is 2, the data is displayed on the fifth physical display line. (If the keyword SLNO is not used in the associated entry format statement, the default value for the starting line number is 2, so 2 is used as the reference for an entry in the *Line* field.)

If the *Line* and *Position* fields are not specified for a field, the starting position for the field defaults to the next available display position following the previous field. The first field defaults to position 1 of logical line 1.

Regardless of whether the SLNO keyword is specified, the defaults for the line and position entries are designed to protect the prompt line from being overlaid. See the *Location Field* description under *Field Description Statements* for a table that shows the relationships between the SLNO keyword, the line and position entries, and the physical line and position on the display.

When devices other than the keyboard/display are assigned to the file associated with the fields, the *Location* field applies. The entry in the *Location* field specifies the starting position.

If the *Location* field is blank in this case, the field is started in the next available position. The sequence of the fields is determined by the order in which they occur in the source program.

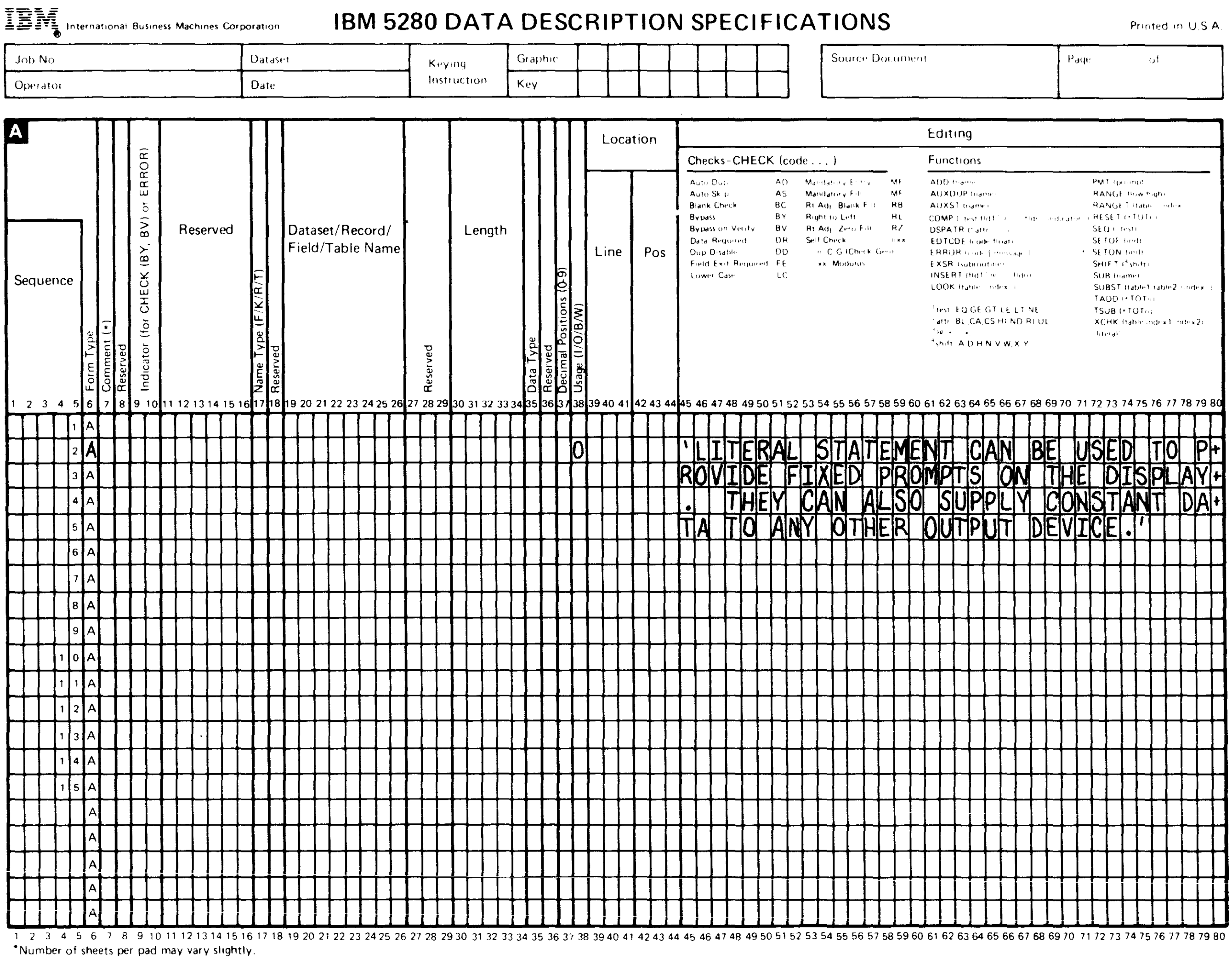
Entries in the *Location* field and the *Line* and *Position* fields must be right adjusted. Either leading zeros or leading blanks are allowed.

Editing Field (Columns 45 through 80)

The fixed prompt message or data is specified in the *Editing* field. The data or message must be preceded and followed by an apostrophe. An apostrophe used in the message must be represented by a pair of apostrophes.

The keyword DSPATR and its parameter string can be added to the *Editing* field to provide emphasis to fixed prompt message. Refer to the DSPATR keyword description in *Field Description Statements* in this chapter.

Example



Chapter 6. Contents

Chapter 6. Calculation Specifications

I/O (input and output) operations and data manipulations can be coded with calculation statements. The calculation operations are a subset of RPG operations and can be coded on the calculation specifications.

Many of the operations that you code in calculation statements can be performed on data tables. These tables are described to DE/RPG through file description statements and table description statements on the A specification. The functions provided through data tables parallel the functions provided through arrays in RPG. Because the operations specified through calculation statements are a subset of RPG operations, this section uses the term *array* to refer to data tables that are defined on the A-specifications. The term *array element* parallels the term *data table entry*.

You can use calculation statements to build subroutines for performing operations that cannot be coded on the A-specifications. The subroutines can be designed for use during a data-entry job or can be organized to control an interactive job or a background job.

The calculation statements can be grouped in the following categories:

- Subroutine beginning and ending statements
- Execute subroutine operation
- Branching operations
- Arithmetic operations
- Compare operations
- Move operations
- Bit operations
- Table search operations
- I/O operations
- Indicator setting operations

Calculation source statements must be organized into subroutines. Each subroutine must begin with a statement that identifies the entry to the subroutine. Each subroutine must also contain one statement to identify the end of the subroutine. When a subroutine ends, program execution continues with the operation that follows the operation that transferred control to the subroutine.

All calculation statements must be included within a subroutine.

CODING CONVENTIONS FOR C-SPECIFICATIONS

The coding of certain fields on the calculations forms specifications (C-specifications) remains consistent regardless of the operation coded. The coding positions of these fields and their uses follow:

Columns	Use
1 through 5	Sequence field
6	Form type field (C)
7	Normally blank (* for a comment line)
8	Blank
9 through 17	Conditioning indicators fields
28 through 32	Operation code field
60 through 74	Comments field
75 through 80	Reserved

Sequence Field (Columns 1 through 5)

Enter the statement sequence number, if desired.

Form Type Field (Column 6)

The form type field for all calculation statements must contain the letter C, which identifies the calculation statement to the compiler.

Conditioning Indicators Fields (Columns 9 through 17)

The use of the conditioning indicators fields is not required, but can be used to specify up to three indicators (01 through 99) that can be tested during program execution to determine whether a specific operation should be performed.

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic								Card Electro Number
Programmer	Date		Punch								

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Not	And	Not	And	Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
3 4 5	6	7 8	9 10	11	12 13	14	15 16	17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74						
0 1	C		1 1		2 3		N 4 7		To satisfy the conditions specified, indicators 11 and 23 must be on and indicator																
0 2	C								47 must be off.																
0 3	C																								
0 4	C																								
0 5	C																								
0 6	C																								
0 7	C																								

The three fields consist of columns 9 through 11, 12 through 14, and 15 through 17. The numbers of the indicators to be tested can be entered in columns 10 and 11, 13 and 14, and 16 and 17. If all the columns contain blanks, no test is to be conducted. Columns 9, 12, and 15 specify whether the indicators must be on (blank in these columns) or off (an N in these columns). Column 9 is associated with the indicators in columns 10 and 11; column 12 with columns 13 and 14; and column 15 with columns 16 and 17. When the condition tested for is true, the function represented by the statement is executed. If any test fails, the function is bypassed.

All calculation statements except three can be conditioned by indicator tests. The exceptions are the two subroutine beginning and ending statements, BEGSR (begin subroutine) and ENDSR (end subroutine), and one of the branching operations, TAG.

Operation Field (Columns 28 through 32)

An operation code is required for each statement. Only the operation codes described in this section can be used. The operation code must start in column 28.

Comments Field (Columns 60 through 74)

Columns 60 through 74 are ignored by the compiler and can be used for comments or explanations. An entire line can be specified as a comment by entering an asterisk in column 7. See *Comment Statements* in Chapter 3.

SUBROUTINE BEGINNINGS AND ENDINGS

The operation codes in this category identify the beginnings and the ends of subroutines and are required for all calculation subroutines. No calculation statements for a subroutine are allowed to precede the beginnings of the subroutine or allowed to follow the end of the subroutine. The operation codes are:

Op Code	Meaning
BEGSR	Identifies the entry point into a subroutine.
ENDSR	Identifies the end of a subroutine.

BEGSR Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Blank	Required	BEGSR	Blank	Blank	Blank

The *Factor 1* field must contain the name of the subroutine. The name can be from one to six character positions long, must start in column 18, and can contain no imbedded blanks.

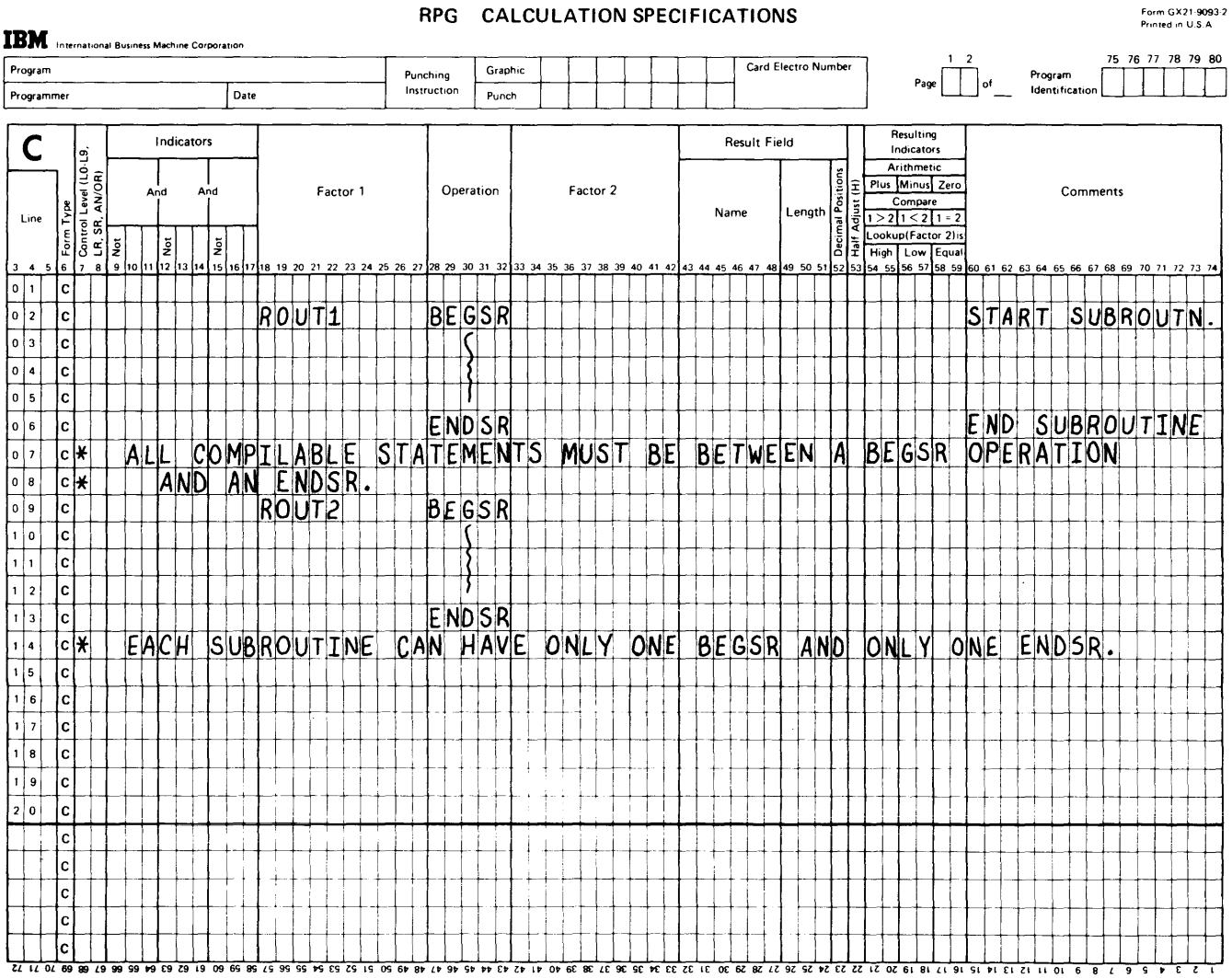
The subroutine name identifies the beginning of the calculation subroutine. By specifying this name, you can specify a subroutine to be executed in field description statements (EXSR keyword), in calculation statements (EXSR operation code), and by an entry format statement naming the subroutine rather than a record.

ENDSR

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Blank	Optional	ENDSR	Blank	Blank	Blank

An entry in *Factor 1* is optional and must be either blank or a name. The name, if entered, can be from one to six character positions long, must start in column 18, and cannot contain imbedded blanks. This name can be used as the destination of a branch from within the calculation subroutine.

Example



RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page ___ of ___ Program Identification

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup(Factor 2)is (High / Low / Equal)	Comments
01	C											
02	C			ROUT1	BEGSR							START SUBROUTN.
03	C											
04	C											
05	C											
06	C				ENDSR							END SUBROUTINE
07	C	*		ALL COMPILABLE STATEMENTS MUST BE BETWEEN A BEGSR OPERATION								
08	C	*		AND AN ENDSR.								
09	C			ROUT2	BEGSR							
10	C											
11	C											
12	C											
13	C				ENDSR							
14	C	*		EACH SUBROUTINE CAN HAVE ONLY ONE BEGSR AND ONLY ONE ENDSR.								
15	C											
16	C											
17	C											
18	C											
19	C											
20	C											

EXECUTE SUBROUTINE OPERATION

The execute subroutine operation (EXSR) suspends the execution of the current calculation subroutine and causes the execution of another calculation subroutine. When the execution of that subroutine is completed, execution resumes with the next operation following EXSR operation.

One calculation subroutine can transfer control to (via an EXSR operation) a second subroutine, which can call a third subroutine, and so on. However, an attempt to call a suspended subroutine is an invalid operation. A suspended subroutine is one that has transferred control to another subroutine or to an entry format and has not been resumed and completed. A subroutine cannot call itself.

EXSR Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	EXSR	Required	Blank	Blank

The *Factor 2* field must contain the name of the subroutine to be executed. This name must be the factor 1 entry for a BEGSR operation. The name must begin in column 33.

Example

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page ___ of ___ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
01	C												
02	C				ROUT1	BEGSR							START SUBROUTN.
03	C												
04	C												
05	C												
06	C					ENDSR							END SUBROUTINE
07	C	*			ALL COMPILABLE STATEMENTS MUST BE BETWEEN A BEGSR OPERATION								
08	C	*			AND AN ENDSR.								
09	C				ROUT2	BEGSR							
10	C												
11	C												
12	C												
13	C					ENDSR							
14	C	*			EACH SUBROUTINE CAN HAVE ONLY ONE BEGSR AND ONLY ONE ENDSR.								
15	C												
16	C												
17	C												
18	C												
19	C												
20	C												
	C												
	C												
	C												
	C												
	C												

EXECUTE SUBROUTINE OPERATION

The execute subroutine operation (EXSR) suspends the execution of the current calculation subroutine and causes the execution of another calculation subroutine. When the execution of that subroutine is completed, execution resumes with the next operation following EXSR operation.

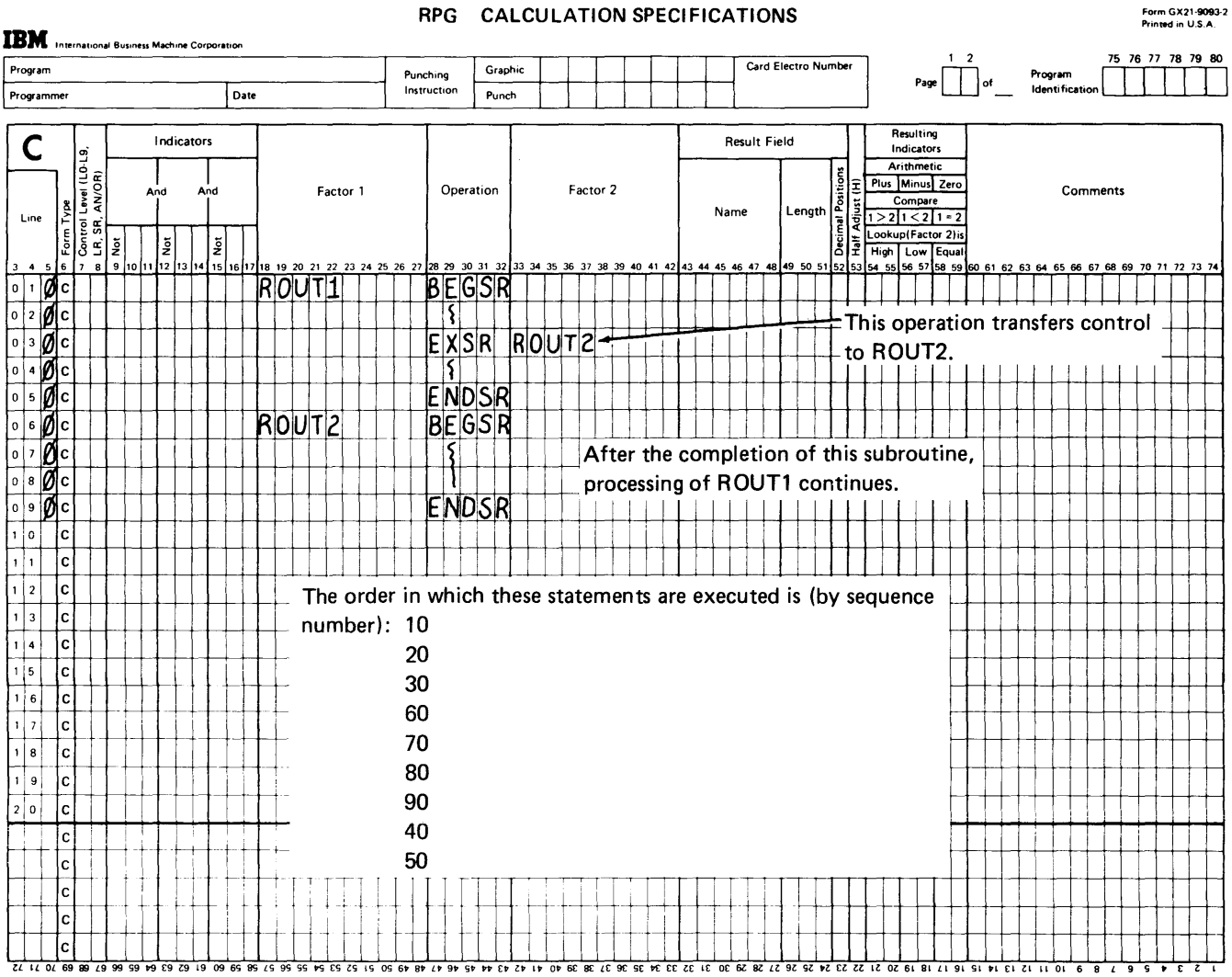
One calculation subroutine can transfer control to (via an EXSR operation) a second subroutine, which can call a third subroutine, and so on. However, an attempt to call a suspended subroutine is an invalid operation. A suspended subroutine is one that has transferred control to another subroutine or to an entry format and has not been resumed and completed. A subroutine cannot call itself.

EXSR Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	EXSR	Required	Blank	Blank

The *Factor 2* field must contain the name of the subroutine to be executed. This name must be the factor 1 entry for a BEGSR operation. The name must begin in column 33.

Example


RPG CALCULATION SPECIFICATIONS
Form GX21-9093-2
Printed in U.S.A.
IBM International Business Machine Corporation
Program
Programmer
Date
Punching Instruction
Graphic
Punch
Card Electro Number
Page
of
Program Identification
C
Line
Form Type
Control Level (L0-L9, LR, SR, AN/OR)
Indicators
And
And
Not
Factor 1
Operation
Factor 2
Result Field
Name
Length
Decimal Positions
Half Adjust (H)
Resulting Indicators
Arithmetic
Plus
Minus
Zero
Compare
1 > 2
1 < 2
1 = 2
Lookup(Factor 2)is
High
Low
Equal
Comments
ROUT1
BEGSR
EXSR ROUT2
ENDSR
ROUT2
BEGSR
ENDSR
This operation transfers control to ROUT2.
After the completion of this subroutine, processing of ROUT1 continues.
The order in which these statements are executed is (by sequence number): 10
20
30
60
70
80
90
40
50

BRANCHING OPERATIONS

The branching operations allow a change in the sequence of execution of operations within a subroutine; execution continues at a specified point in the subroutine. The operation codes in this group are:

Op Code	Meaning
GOTO	Causes a branch to a labeled destination
TAG	Identifies a destination label

Note: The CABxx operations provide conditional branching operations. See *Compare Operations*, later in this section.

GOTO Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	GOTO	Required	Blank	Blank

The GOTO operation causes a branch to a destination in the same subroutine. The GOTO operation cannot branch to a destination outside the subroutine. The destination can either precede or follow the position of the GOTO operation.

The *Factor 2* field must contain the destination label. The destination label can be either a name in the *Factor 1* field of the ENDSR statement for the subroutine or the name used as the factor 1 entry for a TAG operation. The name can be from one to six characters long, can contain no imbedded blanks, and must start in column 33.

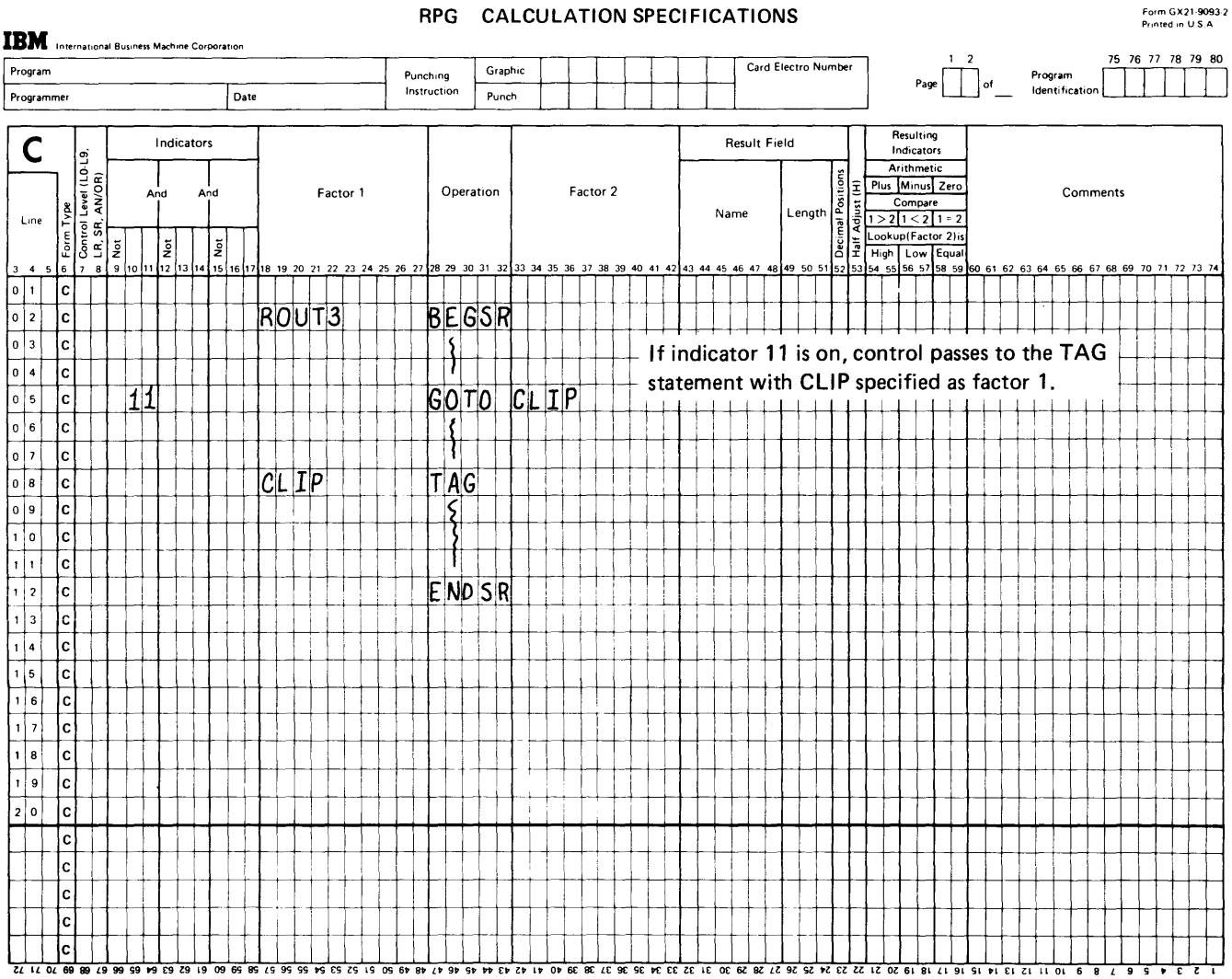
TAG Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Blank	Required	TAG	Blank	Blank	Blank

The only function of a TAG operation is the identification of a name as a destination label. A destination label represents a point within a subroutine to which a branch can occur. The TAG operation cannot be used to identify an entry point into a subroutine.

The *Factor 1* field must contain a name, which can be used as a destination label by a branch operation. The name can be from one to six characters long, must start in column 18, and cannot contain imbedded blanks. The same name cannot be used for more than one TAG operation in the same program. That is, the same name cannot be used for TAG operations in different subroutines within the same program.

Example



ARITHMETIC OPERATIONS

Arithmetic operations can be performed only on numeric data. Decimal alignment is performed for all arithmetic operations and the results are always signed numbers. If the result field specified for an arithmetic operation is not long enough to contain the results of the operation, the value is decimal aligned and the leftmost digits are truncated to the length of the result field. If fewer decimal positions are specified for the result field than are present in the value, the rightmost decimal positions are truncated. If the value is shorter than the result field, the value is decimal aligned and padded with zeros, as appropriate. Neither condition is indicated to the program.

This category of operations consists of the following operation codes:

Op Code	Meaning
ADD	Factor 1 + factor 2 = result
Z-ADD	0 + factor 2 = result
SUB	Factor 1 - factor 2 = result
Z-SUB	0 - factor 2 = result
MULT	Factor 1 x factor 2 = result
DIV	Factor 1 ÷ factor 2 = result
MVR	Remainder from DIV operation = result

The use of fields for factors 1 and 2, result, half adjust, and resulting indicators are described in subsections that follow the descriptions of the operations.

ADD Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	ADD	Required	Required	Optional

The add operation adds the contents of the fields named in *Factor 1* and *Factor 2* and places the sum in the named result field. Statements for an add operation require entries in *Factor 1*, *Factor 2*, and *Result* fields.

Z-ADD Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	Z-ADD	Required	Required	Optional

The zero and add operation adds the contents of the field named in factor 2 to zero and places the sum in the named result field. Entries are required in *Factor 2* and *Result* fields. *Factor 1* must be blank.

SUB Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	SUB	Required	Required	Optional

The subtract operation subtracts the contents of factor 2 from the contents of factor 1 and places the difference in the named result field. Entries are required in *Factor 1*, *Factor 2*, and *Result* fields.

Z-SUB Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	Z-SUB	Required	Required	Optional

The zero and subtract operation subtracts the contents of the field named in factor 2 from zero and places the difference in the named result field. Entries are required in the *Factor 2* and *Result* fields. *Factor 1* must be blank.

MULT Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	MULT	Required	Required	Optional

The multiply operation multiplies the contents of factor 1 by the contents of factor 2 and places the product in the named result field. Entries are required in *Factor 1*, *Factor 2*, and *Result* fields.

DIV Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	DIV	Required	Required	Optional

The divide operation divides the contents of factor 1 by the contents of factor 2 and places the quotient in the named result field. Entries are required in *Factor 1*, *Factor 2*, and *Result* fields. If the contents of factor 1 is zero the result of the divide operation is zero. The contents of factor 2 cannot be zero. If an MVR operation follows the DIV operation, the *Half Adjust* field in the DIV operation must be blank.

MVR Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	MVR	Blank	Required	Optional

The move remainder operation places the remainder from the immediately preceding DIV(divide) operation in the result field. There can be no statements between the DIV statement and the MVR statement. An entry in the *Result* field is required, and the *Factor 1*, *Factor 2*, and *Half Adjust* fields must be blank. (The *Half Adjust* field in the previous divide operation must also be blank.) The value of the remainder is such that, without truncation or rounding, the following equation is true:

remainder = dividend - (divisor x quotient)

Factor 1 (Columns 18 through 27) and Factor 2 (Columns 33 through 44)

Entries in the *Factor 1* and *Factor 2* fields can be numeric constants or the names of numeric fields or array elements.

Numeric constants must be specified as numbers, which can be signed (+ or -) and can include a decimal point. A numeric constant can contain 10 positions maximum, including both the sign and the decimal point.

The name of an array element must be specified as *name,position*, in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 10 characters.

The contents of fields or array elements named in *Factor 1* and *Factor 2* are unchanged by the arithmetic operations unless the same name is used as the *Result* field.

Entries in *Factor 1* and *Factor 2* must start in the leftmost columns (18 and 33). When a sign is entered as part of a constant, the sign must be in the leftmost column of the field.

Result Field (Columns 43 through 52)

All arithmetic operation statements require an entry in the *Result* field. The *Result* field can name a defined numeric field or array element or can define a new numeric field. The result of the arithmetic operation replaces the contents of the field named in the *Result* field. Because the location of the decimal point is the primary consideration for aligning data in the named field, truncation of the data can occur on either end or on both ends of the field.

For example, if the result field is defined as eight positions with two decimal positions, placing the value 1234.567 in the field would result in the truncation of the third decimal position (7). The result field would contain 001234.56. If the value 7654321 were placed in this result field, the leftmost digit (7) would be truncated and the result field would contain 654321.00. Because of this, a result field should always be defined large enough to contain all the meaningful positions of the values that might be placed in the field by arithmetic operations.

If an array element is specified in the result field, it must be specified as *name,position*, in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *Name* and *Position* must be separated by a comma (no blanks) and the total length cannot exceed six characters.

When the name of a defined field is used (defined elsewhere in the program), only the name is required. The name must start in column 43. The *Length* field (columns 49 through 51) and the *Decimal Positions* field (column 52) can be left blank. If entries are made in these fields, the entries must match the length and decimal position entries for the original definition for the field.

When the *Result* field is used to define a new field, entries are required in *Length*, and *Decimal Positions* fields.

The length of a numeric field cannot exceed 15 positions. The entry in the length field must be right adjusted.

The entry in the *Decimal Positions* field describes the number of positions to the right of the decimal point. A number from 0 through 9 is required to designate a numeric field.

Half Adjust Field (Column 53)

The entry in the *Half Adjust* field determines whether truncation occurs on the right end of the result field or rounding occurs in the rightmost position. The only valid entries are blank to indicate no half adjust and the letter H to indicate that half adjusting is required. Half adjusting on a divide (DIV) operation is not allowed if the divide operation is followed by a move remainder (MVR) operation.

Resulting Indicators (Columns 54 through 59)

The *Resulting Indicators* field can be coded so that specified indicators reflect the condition (positive, negative or zero) of the arithmetic result. The indicators can be used to condition subsequent operations.

The three conditions represented are:

- Result positive, that is, greater than zero (columns 54 and 55)
- Result negative, that is, less that zero (columns 56 and 57)
- Result zero (columns 58 and 59)

After the arithmetic operation and half adjust operation (if specified) are performed, all the indicators are turned off, then the indicator that corresponds to the condition of the arithmetic result is set on.

Valid entries in the *Resulting Indicators* field are the numbers 01 through 99 for indicators or blank if no indicator is specified.

Example of Arithmetic Operations

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic								Card Electro Number
Programmer	Date		Punch								

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup(Factor 2)is (High / Low / Equal)	Comments
01	C		(1)	A	ADD	1	A	3	Ø			
02	C		(2)	B	ADD	C	V	5	2			
03	C											
04	C		(3)		Z-ADD	C	V					
05	C											
06	C		(4)	C	SUB	B	W	5	1			
07	C											
08	C		(5)		Z-SUB	C	W					
09	C											
10	C		(6)	B	MULT	G	X	8	4			
11	C											
12	C		(7)	C	DIV	J	Y	6	2			
13	C											
14	C		(8)		MVR		Z	5	3			
15	C											
16	C											
17	C											
18	C											
19	C											
20	C											
	C											
	C											
	C											
	C											
	C											

Before each operation is executed, the field values are:

A = 001
B = 10.0
C = 32
G = 2.77
J = .6

After the operations are executed, the result fields contain the following values:

1	002	5	-0032.0
2	042.00	6	0027.7000
3	032.00	7	0053.33
4	0022.0	8	00.002

Notice that the length and decimal positions of each result field is specified only once.

COMPARE OPERATIONS

The operations in this group compare the contents of factor 1 with the contents of factor 2 and then set resulting indicators to represent the result of the comparision. Alphabetic comparisons are based upon the system collating sequence. Numeric comparisons are performed algebraically. The compare operations group consists of the following operation codes:

Op Code	Meaning
COMP	Compare
CAB	Compare, and branch unconditionally
CABEQ	Compare, and branch if factor 1 equals factor 2
CABNE	Compare, and branch if factor 1 does not equal factor 2
CABLE	Compare, and branch if factor 1 is less than or equal to factor 2
CABLT	Compare, and branch if factor 1 is less than factor 2
CABGE	Compare, and branch if factor 1 is greater than or equal to factor 2
CABGT	Compare, and branch if factor 1 is greater than factor 2

The compare portion of these operations is the same for all operation codes in this group. In addition to the compare function, all the operations except COMP include branching under conditions detected by the comparison. CAB causes an unconditional branch, or a branch on any condition.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	COMP	Required	Blank	Required

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	CABxx	Required	Required	Optional

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	CAB␣␣	Required	Required	Required

Factor 1 (Columns 18 through 27) and Factor 2 (Columns 33 through 44)

An entry in either of these fields can be a character constant, a numeric constant, or the name of a defined field or of an array element containing the data. Character constants must be enclosed between apostrophes. Numeric constants are not enclosed between apostrophes and can contain only numbers, a decimal point, and a sign (+ or -). If a sign is used with a numeric constant, the sign must precede the number.

The name of an array element must be specified as *name,position*, in which *Name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *Name* and *Position* must be separated by a comma (no blanks) and the total length cannot exceed 10 characters.

All entries in *Factor 1* and *Factor 2* must start in the leftmost column of the the field.

Both factors for a comparison must be either character data or both must be numeric data. Numeric data is decimal point aligned for the comparison. If the values being compared are different lengths or have different numbers of decimal positions, zeros are added to make the lengths and decimal positions equal. Character data is leftmost character aligned. If the character data items for comparison are different lengths, the shorter factor is extended to the right with blanks.

Result Field (Columns 43 through 52)

In statements for compare and branch operations, the *Result* field contains the destination label for the branch. The destination label must be either the name in *Factor 1* of a TAG statement or the name in *Factor 1* of an ENDSR statement.

During program execution a branch to the named label occurs after the compare operation if the condition of the branch is met. Branches can be specified only within a subroutine.

Resulting Indicators (Columns 54 through 59)

Entries in the *Resulting Indicators* field specify the indicators that are set on or off as the result of the compare operation. The indicators can be used to condition subsequent operations. One resulting indicator *must* be specified for a COMP operation or an unconditional CAB operation.

The three conditions represented are:

- Factor 1 is greater than factor 2 (columns 54 and 55)
- Factor 1 is less than factor 2 (columns 56 and 57)
- Factor 1 equals factor 2 (columns 58 and 59)

All the specified indicators are set off, then the indicator that corresponds to the comparison result is set on.

Valid entries in the *Resulting Indicators* field are the numbers 01 through 99 for indicators or blank if no indicator is specified.

MOVE OPERATIONS

The operations in this group move all or part of the data from a source into a destination.

The source is not changed by a move operation; all or part of the destination can be replaced by a move. The number of character positions moved corresponds to the length of the shorter member (source or destination) in the move operation. The operation codes of the move group are:

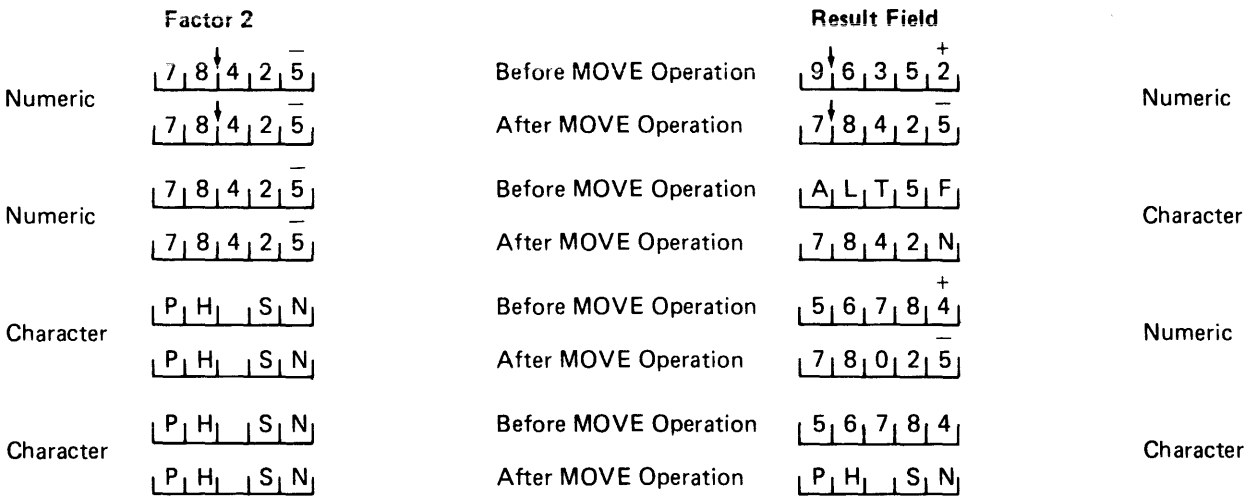
Op Code	Meaning
MOVE	Move from the right of the source to the right end of the destination
MOVEL	Move from the left of the source to the left end of the destination
MOVEA	Move an array

MOVE and MOVEL

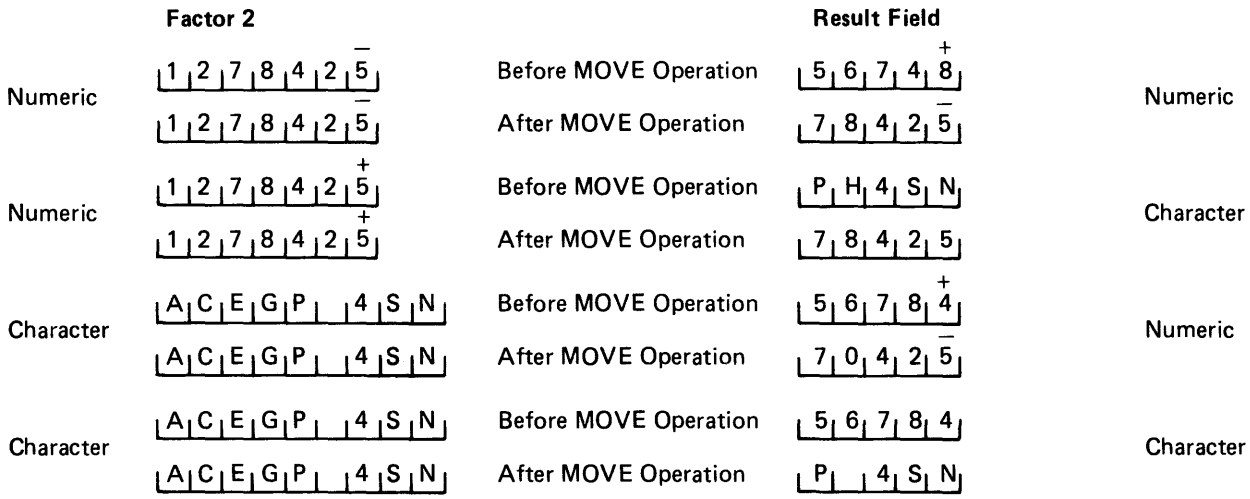
During move operations from a character source to a numeric destination, the moved data undergoes a numeric conversion, which retains the digit portion and replaces the zone portion of each character. (The digit portion of a character in EBCDIC is bits 4 through 7, and the zone is bits 0 through 3.) In the destination only, the zones of the moved characters are normally set to hexadecimal F. (The EBCDIC values for numbers and characters are shown in *Appendix A*.)

The exception to the use of hexadecimal F for the zones of characters that are moved from a character source to a numeric destination involves the sign of numbers. (In a numeric field, the sign of the number is represented by the zone of the rightmost character position. A negative number is identified by a hexadecimal D in the rightmost zone; a positive number contains a hexadecimal F in the rightmost zone in the field.) The zone in the rightmost character position of the character type source is translated: a hexadecimal D remains a hexadecimal D, and all other zones are translated into hexadecimal F. In all cases except one (MOVEL operation with the source shorter than the destination), the translated zone from the rightmost position in the source becomes the sign of the numeric destination. In a MOVEL operation in which the source is shorter than the destination, the sign of the data in the numeric destination before the move is retained. See Figures 4 and 5 for examples of data moves.

Factor 2 and the Result fields are the same length.



Factor 2 is longer than the Result field.



Factor 2 is shorter than the Result field.

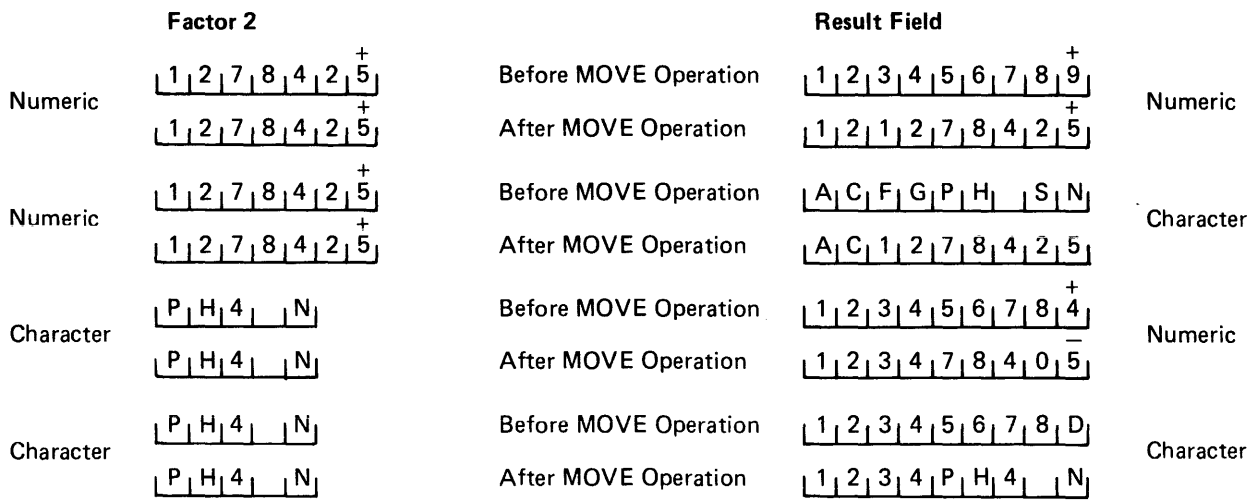
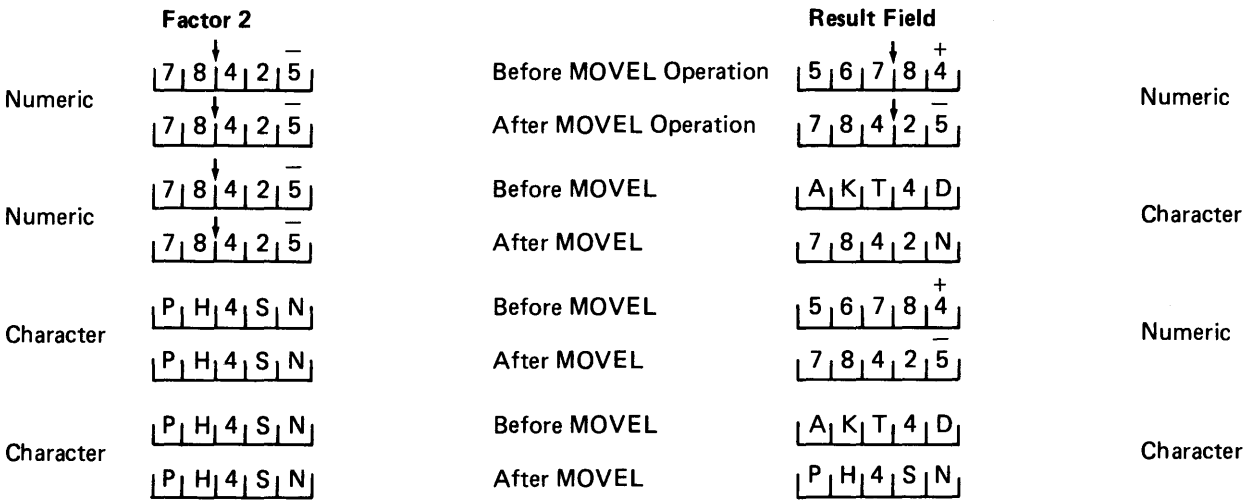
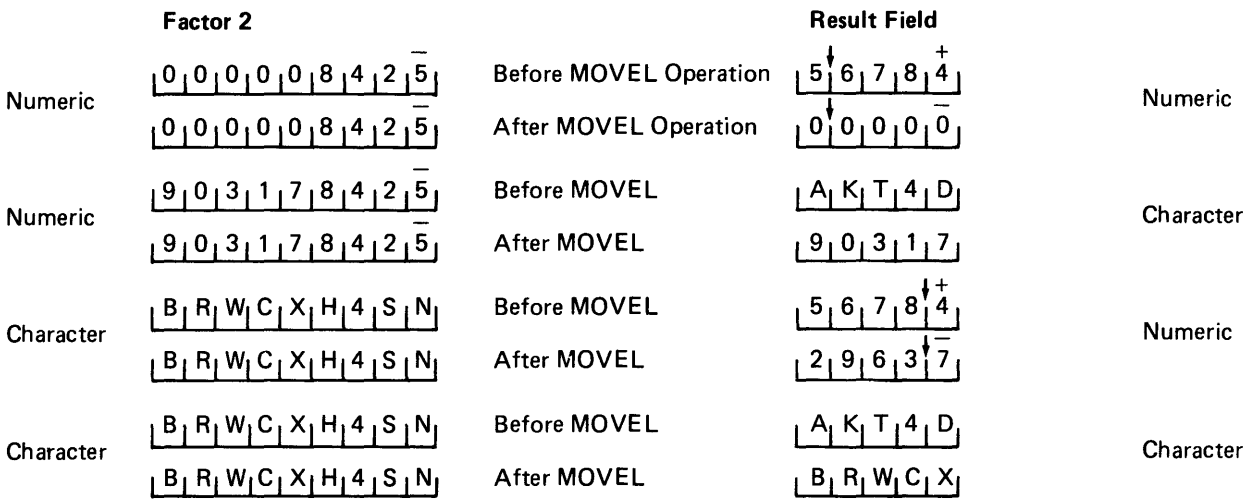


Figure 4. MOVE Operation Examples

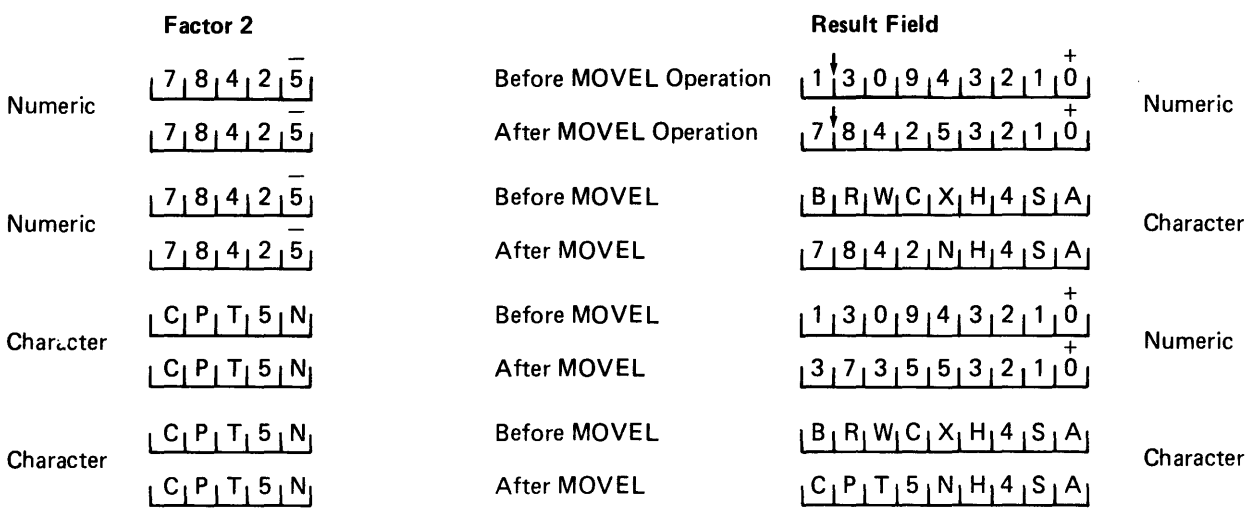
Factor 2 and the Result fields are the same length.



Factor 2 is longer than the Result field.



Factor 2 is shorter than the Result field.



The arrow ↓ between numbers indicates a decimal point.

Figure 5. MOVEL Operation Examples

The basic differences between the move operations are the character position with which the moves start and the direction of progression. MOVE operations start by moving the rightmost character in the source into the rightmost position of the destination. Progression of the MOVE is to the left. MOVEL operations start with the leftmost character position of the source and move into the leftmost position of the destination. Progression of the MOVEL is to the right.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	MOVE	Required	Required	Blank

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	MOVEL	Required	Required	Blank

Factor 2 Field (Columns 33 through 42)

The source from which data is moved must be specified in the *Factor 2* field. The entry in this field can be a character constant, a numeric constant, the name of a defined field, or an array element. Character constants must be enclosed between apostrophes. Numeric constants are coded without apostrophes and can contain only the numbers 0 through 9 and a sign (+ or -) which is optional.

A decimal point can be specified in a numeric constant, but the decimal point location in a numeric source is ignored during these operations. The sign, if used, must be the first character in a numeric constant.

The name of an array element must be specified as *name,position*, in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 10 characters.

An entry in the *Factor 2* field must start in column 33.

Result Field (Columns 43 through 52)

The destination for a move operation must be named in the *Result* field. The entry in this field can be the name of a defined field, an array element, or the entry can define a new field.

The name of an array element must be specified as *name,position*, in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 6 characters.

When the name of a defined field is used, only the name is required. The name must start in column 43. The *Length* field and the *Decimal Positions* field can be left blank. If entries are made, they must match the length and decimal positions entries for the original definition for the field.

When the *Result* field is used to define a new field, entries are required in *Length* and *Decimal Positions* fields.

The length of a numeric field cannot exceed 15 positions, and the length of a character field cannot exceed 256 positions. The length entry must be right adjusted (end in column 51).

To specify a character field, leave the *Decimal Positions* field blank. To specify a numeric field enter a number from 0 through 9 in the *Decimal Positions* field.

MOVEA

The MOVEA operation transfers characters from the *Factor 2* field (starting with the leftmost position) to the *Result* field (starting with the leftmost position). The movement of data starts with the first element of an array if the array is not indexed, or with the element referenced if the array is indexed, or with the leftmost position of the field or the literal.

The length of the MOVEA operation is determined by the shorter of the lengths of *Factor 2* and the *Result* field. If *Factor 2* is longer than the *Result* field, the excess rightmost characters of *Factor 2* are not moved. If the *Result* field is longer than *Factor 2*, the rightmost characters in the *Result* field are unchanged. Data movement may end in the middle of an array element or field in either of these cases.

Note: The MOVEA opcode is used here for consistency with RPG. However, the term *array* in this description refers to DE/RPG table functions.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	MOVEA	Required	Required	Blank

Factor 2 Field (Columns 33 through 42)

The source from which data is moved must be specified in the *Factor 2* field. The entry in this field can be a character constant, the name of a defined field, or an array element. Character constants must be enclosed within apostrophes. Either *Factor 2* or the *Result* field must contain an array. However, *Factor 2* and the *Result* field cannot reference the same array even if the array is indexed. All arrays and fields referenced by a MOVEA operation must be alphameric.

The name of an array element must be specified as *name,position*, in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 10 characters.

An entry in the *Factor 2* field must start in column 33.

Result Field (Columns 43 through 52)

The destination for a MOVEA operation must be named in the *Result* field. The entry in this field can be the name of a defined field, an array element, or the entry can define a new field.

Factor 2 or the *Result* field must contain an array. However, *Factor 2* and the *Result* field cannot reference the same array even if the array is indexed. Resulting indicators are not allowed. All arrays and fields referenced by a MOVEA operation must be alphameric.

The name of an array element must be specified as *name,position,* in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 6 characters.

When the name of a defined field is used, only the name is required. The name must start in column 43. The *Length* field can be left blank.

When the *Result* field is used to define a new field, an entry is required in the *Length* field.

The length of a character field cannot exceed 256 positions. The length entry must be right adjusted (end in column 51).

To specify a character field, leave the *Decimal Positions* field blank.

See Figure 6 for examples of data moves.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/R/K/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
01	A					F		DATA		1							NUMENT(10)
02	A					T		AR1		1							
03	A																
04	A																
05	A																
06	A																

Checks=CHECK (code . . .)

Auto Dup	=AD	Mandatory Entry	=ME
Auto Skip	=AS	Mandatory Fill	=MF
Blank Check	=BC	Rt Adj–Blank Fill	=RB
Bypass	=BY	Right to Left	=RL
Bypass on Verify	=BV	Rt Adj–Zero Fill	=RZ
Data Required	=DR	Self-Check	=mxx
Dup Disable	=DD	D=M/G (Check/Gen)	
Field Exit Required	=FE	xx=Modulus	
Lower Case	=LC		

Functions

ADD (name)
AUXDUP (name)
AUXST (name)
COMP ('test fld1'@ fldn 'literal' (indicator))
DSPATR ('attr)
EDTCDE (code 'float')
ERROR (code ['message'])
EXSR (subroutine)
INSERT (fld1'@ fldn 'literal')
LOOK (table (index))
PMT (prompt)
RANGE (low high)
RANGET (table (index))
RESET ([*TOTn] (name))
SEQ ('test)
SETOF (ind)
SETON (ind)
SHIFT ('shift)
SUB (name)
SUBST (table1 table2 (index))
TADD ([*TOTn] (name))
TSUB ([*TOTn] (name))
XCHK (table index1 index2)
'literal'

'test=EQ,GE,GT,LE,LT,NE
'attr=BL,CA,CS,HI,ND,RI,UL
'@=- + /
'shift=A,D,H,N,V,W,X,Y

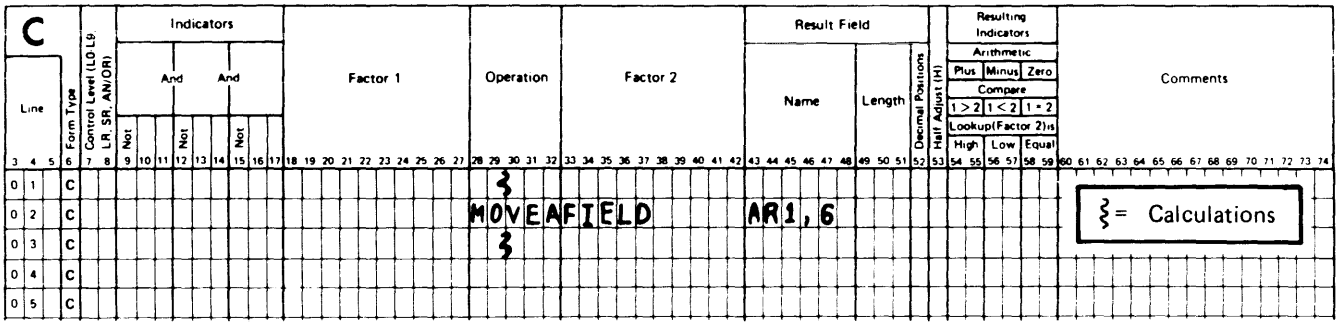
C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And Not	And Not	And Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
01	C						ξ															
02	C						MOVEA	AR1,5	FIELD	10												ξ = Calculations
03	C						ξ															
04	C																					
05	C																					

Factor 2 is shorter than the Result field.

Factor 2 (AR1, 5)		Result Field (FIELD)
A B C D E F G H I J	Before MOVEA operation	Q R S T U V W X Y Z
	After MOVEA operation	E F G H I J W X Y Z

Figure 6 (Part 1 of 2). An Example Showing How to Use the MOVEA Operation



The Result Field is shorter than Factor 2.

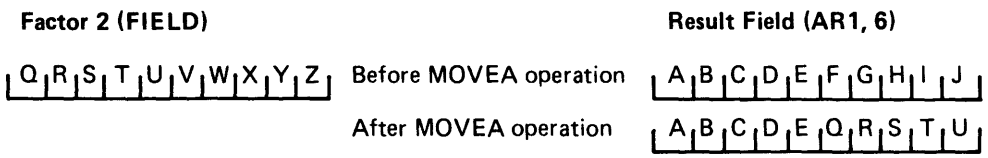


Figure 6 (Part 2 of 2). An Example Showing How to Use the MOVEA Operation

BIT OPERATIONS

The operations in this group set or test individual bits and combinations of bits in a one-character-position field or table entry. The bit positions to be set or tested must be specified in a mask. A mask is a list of numbers (0 through 7) that identifies the numbers of the bits to be set or tested.

The bit operations group consists of three operation codes:

Op Code	Meaning
BITON	The BITON operation causes bits identified in factor 2 to be set on (to be set to 1) in the field named as the result field.
BITOF	The BITOF operation causes bits identified in factor 2 to be set off (to be set to 0) in the field named as the result field.
TESTB	The TESTB operation compares the bits identified in factor 2 with the corresponding bits in the field named as the result field. After the operation, the resulting indicators reflect the status of the result field bits.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	BITON	Required	Required	Blank

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	BITOF	Required	Required	Blank

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	TESTB	Required	Required	Required

Factor 2 (Columns 33 through 42)

The *Factor 2* field represents the mask, which identifies the bit positions to be set on, set off, or tested. The entry in *Factor 2* can be either the bit numbers (enclosed in apostrophes) or the name of a one-character field or array element that contains the mask.

The name of an array element must be specified as *name,position*, in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 10 characters.

When the bit numbers are used to represent the mask, the bits to be set or tested are identified by the numbers of 0 through 7 (0 is the left-most bit). Only the bit positions specified are set or tested. The numbers of the bit positions must be enclosed in apostrophes. For example, to set on bits 0, 2, and 5, factor 2 must specify '025'.

When a field or array element is named in *Factor 2*, the field or array element must be a character field or element with a length of one position. The bits that are on in the field are the bits to be set on, set off, or tested.

Entries in *Factor 2* must start in column 33.

Result Field (Columns 43 through 52)

The *Result* field identifies the field in which the bit positions corresponding to the mask are set on, set off, or tested. The bit positions that do not have corresponding positions specified in the mask are not affected or tested.

The entry in *Result* field must be the name of a character type field or array element with a length of one character position. If the field name in *Result* field is a defined field, only the name is required. If the field is not defined, enter the name starting in column 43 and enter the number 1 in column 51.

The name of an array element must be specified as *name,position*, in which *name* is the name of the array, and *position* is the relative position of the element in the array. The *position* portion of an array-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 6 characters.

Resulting Indicators (Columns 54 through 59)

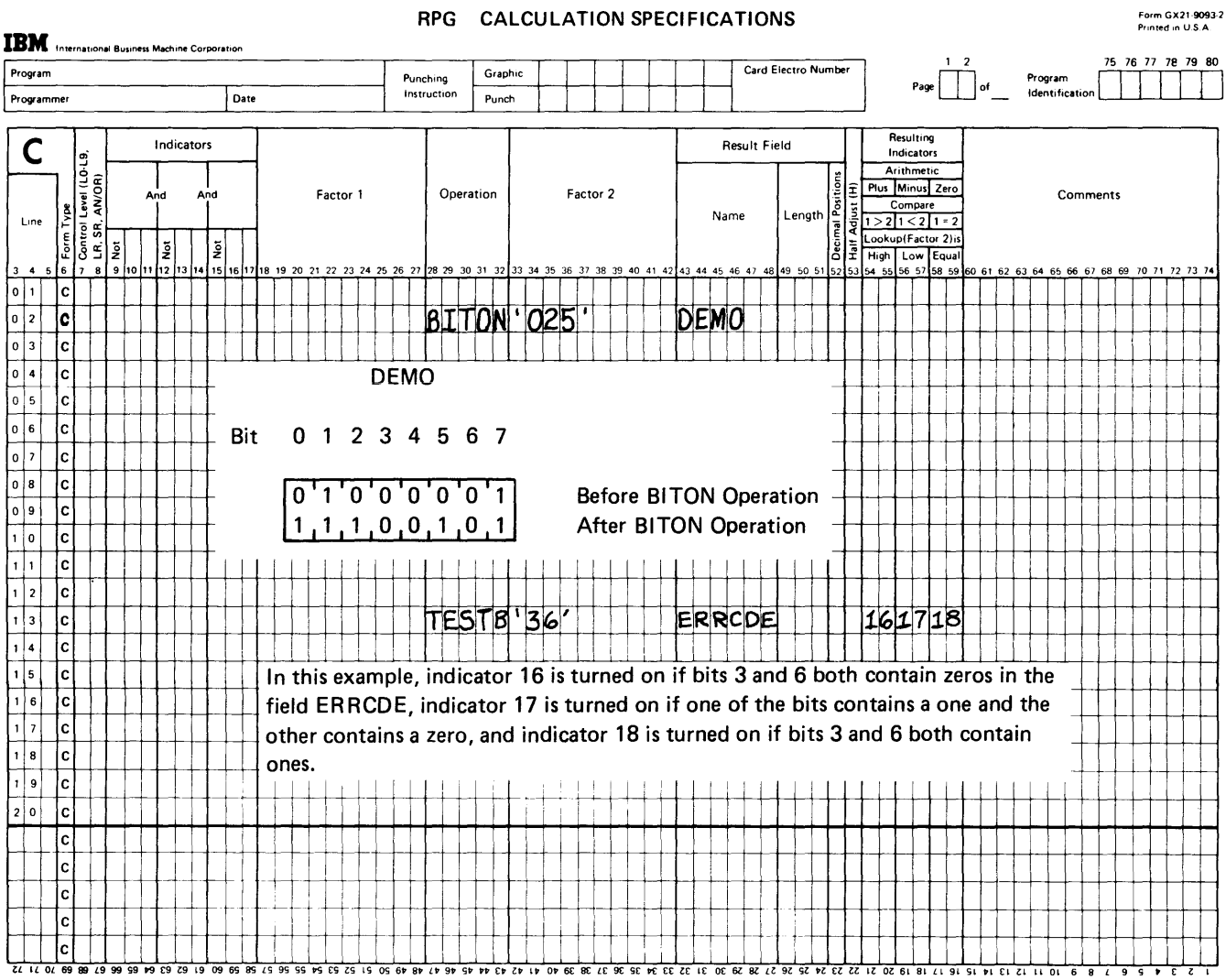
The *Resulting Indicators* field can be coded for TESTB operations to represent the condition of the tested bits. The specified indicator is set on if the corresponding condition exists. Each indicator is specified by its number (01 through 99).

The indicator specified in columns 54 and 55 is set on if all bits tested are off.

The indicator specified in columns 56 and 57 is set on if some of the bits tested are on and some are off.

The indicator specified in columns 58 and 59 is set on if all bits tested are on.

Example



LOKUP Operation

The LOKUP operation searches a named table for a specific element in that table. When the search is satisfactorily completed, a specified indicator is set on.

The LOKUP operation uses a search argument, which is specified in the *Factor 1* field, and successively compares the search argument to the elements in the table until the desired match (as specified in the *Resulting Indicators* field) is found.

The organization of the searched table determines the nature of the searches that can be conducted. Tables in which the elements are in EBCDIC collating sequence (shown in *Appendix A*) can be searched for an element that equals the search argument, for the nearest element greater than the search argument, or for the nearest element less than the search argument. Searches can be conducted for combinations of greater than or equal and less than or equal.

Tables in which the elements are not in collating sequence can be searched for an equal match only.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	LOKUP	Required	Blank	Required

Factor 1 (Columns 18 through 27)

The entry in *Factor 1* specifies the search argument, which is the element for which the table is searched. Search arguments must have the same characteristics as the elements in the searched table. A LOKUP operation on a table containing character type elements must use a character search argument, and an operation on a table containing numeric elements requires numeric search arguments. Search arguments must have the same length as the elements in the searched table.

The search argument can be specified in *Factor 1* as a character constant, a numeric constant, or the name of a field or table element that contains the search argument.

To specify a character constant, enclose the characters in apostrophes. Include any blanks required to match the desired table entry.

Numeric constants are specified without apostrophes and can contain only numbers, a decimal point, and a sign (+ or -). The sign, if used, must precede the first digit.

The name of a table element must be specified as *name,position*, in which *name* is the name of the table, and *position* is the relative position of the element in the table. The *position* portion of a table-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks) and the total length cannot exceed 10 characters.

The entry in *Factor 1* must start in column 18.

Factor 2 (Columns 33 through 42)

The entry in *Factor 2* names the table that is to be searched and can be used to specify the first table element of the search.

The table named in *Factor 2* refers to a table description statement on an A-specification. The name is required and must start in column 33.

A second and optional entry specifies the relative number of the table element at which the search begins. If the second entry is missing, each search starts at the beginning of the searched table and progresses toward the end. The second entry must be separated from the table name by a comma. Imbedded blanks are not allowed.

The second entry can be specified as a whole number to represent the relative number of the first table element of the search or can be specified as the name of a field that contains the integer value. When a field name is used, the relative number of the table element that satisfies the search replaces the initial contents of the field. If the search is not satisfactorily completed, the contents of the field is set to 1.

Resulting Indicators Field (Columns 54 through 59)

The entry in the *Resulting Indicators* field serves two purposes; it specifies the nature of the search (high, low, or equal) and specifies the indicators that can be set at the completion of the search.

Columns 54 and 55 specify a search for the table element immediately higher than the search argument. Again, a search of this nature can be specified only for tables in which the elements are in ascending sequence. When the LOKUP operation is satisfactorily completed, the indicator specified in these columns is turned on.

Columns 56 and 57 specify a search for a table element that is immediately lower than the search argument. A search of this nature can be specified only for tables with elements in ascending sequence. When the search is satisfactorily completed, the corresponding indicator is turned on.

Columns 58 and 59 specify a search for a table element that equals the search argument. The first element that satisfies the search stops the LOKUP operation and turns on the indicator specified in these columns.

Combinations of indicators can be specified in the *Resulting Indicators* field. A search for low and equal or for high and equal can be specified. In such cases, the equal search is performed first. If an equal match between the search argument and a table element is found, the indicator specified in columns 58 and 59 is turned on and the LOKUP operation stops. If an equal match is not found, the high or low (whichever is specified) portion of the search is conducted and the corresponding indicator is set on when the element is found.

At least one and not more than two indicators must be entered in the *Resulting Indicators* field for each LOKUP operation statement.

I/O OPERATIONS

The operations in this group provide access to and control of data transfers between the program and the I/O devices.

The operation codes for I/O access are:

Op Code	Meaning
READ	Read the next record (forward)
READP	Read a prior record (backward)
CHAIN	Read the specified record
WRITE	Create a new record
UPDAT	Rewrite the record in the location where read
DELET	Delete a record
SETLL	Position the file for sequential reading
EXFMT	Execute a format
OPEN	Open a file
CLOSE	Close a file
FEOD	Force end of data

Note: Only the OPEN, READ, UPDAT and CLOSE operation codes can be used for files using an ADDROUT index data set. The FEOD operation code can be used only for communications files (DEVICE(COMM) specified on the file description statement).

The significance of the fields on the calculation form depends upon the operation performed. Therefore, separate coding information is provided for each operation code.

Note: Before the first operation is issued to a communications file, a message should be displayed that instructs the operator to establish the connection, if necessary.

READ Operation

The READ operation provides sequential access to records in a file and can be thought of as an operation to read the next sequential record. READ operations can be used to collect data from a magnetic stripe reader, communications, or to retrieve records from data sets. Nonindexed data sets are accessed in ascending relative record number sequence. Key index data sets are accessed in ascending key sequence and the contents of the records in that data set determine the order in which the data records are processed. A file that has an ADDROUT index data set is accessed in the sequence indicated by the index.

The data record read from a magnetic stripe reader contains character data. The only characters that can be in the record are the numbers 0 through 9, a colon (:), a percent sign (%), an at sign (@), a logical not symbol (¬), and an equal sign (=).

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	READ	Required	Blank	Required

Factor 2 Field (Columns 33 through 42)

The entry in *Factor 2* identifies the item to be read. The item can be specified as either a record name or a file name.

When the *Factor 2* entry is the name of a file, which must be described in a file description statement, the READ operation accesses the next sequential record in the specified file. The record is analyzed using the RECID keyword to determine which record was read.

When the *Factor 2* entry is the name of a record, which must be described in a record description statement, the READ operation accesses the next record and uses the RECID keyword in the record description statement for the named record to determine whether the record accessed is the record type requested. If the record read is not the record type requested, the next sequential record is accessed and the new record is tested. When the requested record type is found, the format for the requested record is applied to the record. If RECID is not specified in the record description statement for the named record, the first record accessed is accepted and the format for the requested record is applied.

The entry in *Factor 2* must start in column 33.

Resulting Indicators Field (Columns 54 through 59)

The *Resulting Indicators* field specifies the indicators to be used to signify that a device error occurred during the execution of the READ operation or that the last record was read. An indicator should be specified if a communications file is being read.

The EOF (end of file) indicator must be entered in columns 58 and 59. The device error indicator (columns 56 and 57) is optional. Columns 54 and 55 must be blank. The only valid entries are the numbers of indicators (01 through 99). The specified indicator is set on to signify that a condition exists.

If no device error indicator is specified in columns 56 and 57 and a device error occurs on the READ operation, the system error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

READP Operation

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	READP	Required	Blank	Required

The READP (read prior record) operation and the READ operation are identical except that the READP operation code causes records to be read backwards through the file. For a keyed file, a READP operation reads the next record in descending key sequence. (READP cannot be used for files using an ADDROUT index data set.)

CHAIN Operation

This operation provides for random retrieval of records from a data set. When a CHAIN operation is executed, a specific record is requested.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	CHAIN	Required	Blank	Required

Factor 1 Field (Columns 18 through 27)

The entry in *Factor 1* is the search argument that identifies the requested record. The requested record can be identified either by relative record number or by key, depending upon the data set organization. If the data set contains records without keyfields, the requested record must be identified by relative record number. If the data set contains records with keyfields, the requested record must be identified by key.

A relative record number can be specified in *Factor 1* as either a numeric constant or the name of a numeric field or table element containing the relative record number. Numeric constants are specified without apostrophes and can contain only numbers and a sign (+). The constant must be a positive number with zero decimal places. The sign, if used, must precede the first number.

Factor 1 specifies the search argument to be used to select the record. The search argument for a keyed file can be specified as a character constant or the name of a field or table element containing the key of the requested record. The search argument specified must have the same characteristics as the key field in the requested record. The specified search argument and the key field in the record must have the same data type, length, and number of decimal positions.

Character constants must be enclosed in apostrophes.

The name of a table element must be specified as *name,position,* in which *name* is the name of the table, and *position* is the relative position of the element in the table. The *position* portion of a table-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *Name* and *Position* must be separated by a comma (no blanks).

Entries in *Factor 1* must start in column 18.

Factor 2 Field (Columns 33 through 42)

The entry in *Factor 2* identifies the requested item. The item can be specified as either a record name or a file name. The entry must start in column 33.

When the *Factor 2* entry is the name of a record, which must be described in a record description statement, the CHAIN operation accesses the specified record and uses the RECID keyword in the record description statement for the named record to determine whether the record accessed is the record type requested. If the record accessed is not the record type requested, the record-not-found indicator specified in the Resulting Indicators field is turned on.

Resulting Indicators Field (Columns 54 through 59)

The resulting indicators field specifies the indicators to be used to signify that a device error occurred during the execution of the chain operation or that the record was not found.

The record-not-found indicator must be entered in columns 54 and 55. The device error indicator (columns 56 and 57) is optional. Columns 58 and 59 must be blank. The only valid entries are the numbers of indicators (01 through 99). The specified indicator is set on to signify that a condition exists. The record-not-found indicator is set on if the record requested by the chain operation is a deleted record.

If no device error indicator is specified in columns 56 and 57 and a device error occurs on the chain operation, the system error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

WRITE Operation

WRITE operations transfer a record to an output device. WRITE operations can be directed to the display, to diskette, to a printer, or to the communications feature. The device is determined by the DEVICE keyword in the file description statement associated with the record description statement for the record named. EDTCDE is the only keyword that can be specified for fields contained in records that are written by the WRITE operation.

Records written to the display cannot request input from the keyboard and can contain only output fields.

The placement of records in diskette data sets during WRITE operations depends upon the presence or absence of key field description statements and the use of the INDEX keyword.

When the records in a data set do not contain a key field, a new record is added at the end of the existing records.

WRITE operations directed to the communications feature transmit records by using the record length specified on the file description statement or the record description statement.

When the records in a data set contain a key field, but no index file is named as a parameter for the INDEX keyword (which can be specified only in file description statements), a new record is inserted or merged into the data set in key sequence. All displaced records are rewritten.

When an index file name is specified in the parameter for the INDEX keyword, a new data record is added at the end of the existing data records, and an index record is inserted into the index file in key sequence.

Note: If the WRITE operation is the only operation used for a diskette file, the file will be erased at Open time. In addition, the file record length will be set to the record length specified in the file definition.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	WRITE	Required	Blank	Optional

Factor 2 (Columns 33 through 42)

The entry in *Factor 2* names the record to be written. The record name must be used in a record description statement.

Resulting Indicators (Columns 54 through 59)

The *Resulting Indicator* field specifies the indicators to be used to signify that a device error occurred during the execution of the WRITE operation or that the last available record space in the file was used. An indicator should be specified if a record is being written to a communications file.

Use of the *Resulting Indicators* field is optional. The only valid entries are the numbers of indicators (01 through 99). Columns 56 and 57 are used for the device error indicator; columns 58 and 59 for the last-record-space-used (end-of-file) indicator. Columns 54 and 55 must be blank.

If no device error indicator is specified in columns 56 and 57 and a device error occurs on the WRITE operation, the system error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

UPDAT Operation

An UPDAT operation rewrites a diskette record in the same location from which it was read. A record can be retrieved from a data set by a READ operation, a READP operation, or a CHAIN operation. The record data can be modified and then an UPDAT operation performed to rewrite (or update) the record. The modified data replaces the original data in the same location. On a keyed file, the key field cannot be updated.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	UPDAT	Required	Blank	Optional

Factor 2 (Columns 33 through 42)

The entry in *Factor 2* must be the name of the record to be written. The record must be described on the A-specification.

Resulting Indicators (Columns 54 through 59)

The *Resulting Indicator* field specifies the indicator to be used to signify that a device error occurred during the execution of the UPDAT operation. Use of this field is optional. The only valid entries are the numbers of indicators (01 through 99). If used, the device error indicator must be specified in columns 56 and 57. Columns 54 and 55 and columns 58 and 59 must be blank.

If no device error indicator is specified and a device error occurs on the UPDAT operation, the system error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

DELET Operation

A DELET operation deletes a record from a diskette data set. The record cannot subsequently be retrieved. In data sets without key fields, a DELET operation can be performed only on the last record retrieved from the data set. In data sets in which the records contain key fields, the record to be deleted can be specified by key.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Optional	DELET	Required	Blank	Optional

Factor 1 (Columns 18 through 27)

An entry in *Factor 1* identifies the search argument for the DELET operation. The search argument specifies the contents of the key field of the record to be deleted. The absence of a search argument in *Factor 1* infers that the last record retrieved from the diskette data set is to be deleted.

The search argument can be specified as a numeric constant, a character constant, or the name of a field or table element that contains the key. Character constants must be enclosed within apostrophes. Numeric constants can contain only the numbers 0 through 9, a decimal point, and a sign (either + or -). When a sign is used, it must precede the first number.

The name of a table element must be specified as *name,position,* in which *name* is the name of the table, and *position* is the relative position of the element in the table. The *position* portion of a table-element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks).

Entries in *Factor 1* must start in column 18, and the search argument must have the same format length, and number of decimal positions as the key field in the record.

Factor 2 (Columns 33 through 42)

Factor 2 identifies the data set from which the record is to be deleted. The entry in *Factor 2* must be the name of a file described on the A-specification.

Resulting Indicators (Columns 54 through 59)

The *Resulting Indicator* field specifies the indicator to be used to signify that a device error occurred during the execution of the DELET operation. Use of the field is optional. The only valid entries are the numbers of indicators (01 through 99). If used, the device error indicator must be specified in columns 56 and 57 and the record-not-found indicator in columns 54 and 55. If factor 1 is specified, the record-not-found indicator must be specified. Columns 58 and 59 must be blank.

If no device error indicator is specified and a device error occurs on the DELET operation, the system error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

SETLL Operation

A SETLL operation allows you to position a diskette file to a specific starting point for a series of sequential read operations. The file position (the search argument) is specified as a relative record number or as the value of a key field. The first record retrieved by a read operation following a SETLL operation is the record with the matching relative record number or key. If no match for a specified key exists, the next record in key sequence is read.

The search argument must be specified as a relative record number if the data set contains records that do not contain key fields. The search argument must be specified as a key value if the data set contains records that contain a key field.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Required	SETLL	Required	Blank	Optional

Factor 1 (Columns 18 through 27)

The entry in *Factor 1* specifies the relative record number or the value of the key field to be used as the search argument. This value can be represented by a character constant, a numeric constant, or by the name of a field or table element that contains the value. The search argument must have the same format, length, and number of decimal positions as the key field of the file. The entry in *Factor 1* must start in column 18.

Character constants must be enclosed with apostrophes. A numeric constant can contain only the numbers 0 through 9, a decimal point, and a sign (+ or -). The sign, if used, must precede the first number, and the sign and the decimal point, if used, do not contribute to the length of the numeric constant.

The name of a table element must be specified as *name,position,* in which *name* is the name of the table, and *position* is the relative position of the element in the table. The *position* portion of a table element name can be specified as either a whole number or the name of a numeric field containing a whole number. The entries for *name* and *position* must be separated by a comma (no blanks).

Factor 2 (Columns 33 through 42)

The entry in *Factor 2* must be the name of the file as described on the A-specification.

Resulting Indicators (Columns 54 through 59)

The *Resulting Indicator* field can be used to specify the indicators to be used for the SETLL operation. The indicator specified in columns 54 and 55 is set on if there are no records in the file greater than or equal to the search argument. The indicator in columns 56 and 57 is used to indicate an I/O device error. The indicator in columns 58 and 59 indicates that a record was found that matched the search argument.

If no device error indicator is specified and a device error occurs on the SETLL operation, the system error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

EXFMT Operation

The EXFMT operation transfers control to a format, which causes the format to be written to the keyboard/display and then read. The format is always written to the display starting on physical line two. A calculation subroutine can execute a format using this operation, but it is invalid to execute to a format while any format is suspended. A suspended format is one that transferred control to a calculation subroutine and that has not been resumed and completed.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	EXFMT	Required	Blank	Optional

The *Factor 2* field must contain the name of a record assigned to the CRT file and described on the A-specification. (This record name does not have to be used as the name of an entry format statement on the Z-specification.) The name must start in column 33.

The resulting indicator field specifies the indicator to be used to signify that the operator pressed a function key, command key, or record backspace key. Use of this field is optional. If no indicator is specified, and the operator presses one of these keys, the keystroke is ignored. The only valid entries in the resulting indicator field are the numbers of indicators (01 through 99). If used, the indicator must be specified in columns 56 and 57. Columns 54 and 55 and columns 58 and 59 must be blank.

When an indicator is specified in columns 56 and 57 and the operator presses a function key, command key, or the record backspace key, the indicator is turned on and the appropriate I/O device status code is returned in the status variable specified by the STATUS keyword in the job statement. The codes that can be returned for the function keys are:

Code	Key Pressed
9501	Home
9502	Select format
9503	Cancel
9504	Page forward
9505	Next format
9506	Print
9507	Erase input
9508	Record correct
9509	Help

The codes that are returned for the command keys are shown in the following chart directly above or below the command function. If no code is shown for a command function, that command operates in the normal manner. The blocks that are blank are ignored. If a key related to one of these blocks is pressed, no operation is performed and no code is returned to the program.

				9519	9520	9521	9522	9523			9524		9525
	Comm Attention	End Input	Comm Status	Search End of Data	Load Format	Change Format	Display Format	Display Data		Dump/ Trace File Open	Auto Mark		Clear Screen
Hex	Review Second Data Set	Transfer Record	Return to Trans-action Data Set	Search Content	Search Seq Content	Search Record Number	End of Job	Edit Release	Mark Field	Delete Record	Insert Record	Display Verify Record	
	9510	9511	9512	9513	9514	9515				9516	9517	9518	

If a device error occurs during the EXFMT operation, the system error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

OPEN Operation

The OPEN operation logically opens a named file for processing. In DE/RPG programs, all files except communications files are opened during program start-up unless JOBOPT(*NOOPEN) is specified on the job specification statement (Z-specification). A file must be opened before it can be used in I/O operations.

Note: If the WRITE operation is the only operation used for a diskette file, the file will be erased at Open time. In addition, the file record length will be set to the record length specified in the file definition.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	OPEN	Required	Blank	Optional

Factor 2 (Columns 33 through 42)

The entry in *Factor 2* identifies the file to be opened. The entry must be the name of a file described in a file description statement on the A-specification.

Resulting Indicators (Columns 54 through 59)

An indicator (01 through 99) can be specified in columns 56 and 57. If an attempt to open the specified file is unsuccessful, the indicator is set on. If the OPEN operation is completed, the indicator is set off.

All other columns in the *Resulting Indicators* field must be blank.

If no indicator is specified in columns 56 and 57 and the OPEN operation fails, an error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

CLOSE Operation

The CLOSE operation logically closes a named file. In DE/RPG programs, all files are automatically closed at end of job if they are not explicitly closed in the program.

If the file being closed is a communications file, the CLOSE operation does *not* drop, or terminate, the communications line. The line is dropped at end of job. It is not necessary to close a communications file between a READ operation and a WRITE operation or between WRITE and READ operations.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	CLOSE	Required	Blank	Optional

Factor 2 (Columns 33 through 42)

The entry in *Factor 2* identifies the file to be closed. The entry must be the name of a file described in a file description statement on the A-specification.

Resulting Indicators (Columns 54 through 59)

An indicator (01 through 99) can be specified in columns 56 and 57. If an error is detected during the CLOSE operation, the specified indicator is set on. If the CLOSE operation is completed without errors, the indicator is set off.

All other columns in the *Resulting Indicators* field must be blank.

If no indicator is specified in columns 56 and 57 and the CLOSE operation fails, an error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

FEOD Operation

The FEOD operation signals the logical end of data indication for a communications file. This operation is valid only for a communications file and must be specified only between WRITE operations to that file. This operation is invalid for COMM3270 files.

The FEOD operation does not disconnect the program from the file. The file can be used for subsequent I/O operations without an OPEN operation being performed. The next operation specified for the file must be a WRITE operation.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	FEOD	Required	Blank	Optional

Factor 2 (Columns 33 through 44)

The entry in *Factor 2* identifies the communications file to which the end of data indication is to be sent. The entry must be the name of a communications file described in a file description statement on the A-specification.

Resulting Indicators (Columns 54 through 59)

An indicator (01 through 99) can be specified in columns 56 and 57. If an error is detected during the FEOD operation, the specified indicator is set on. If the FEOD operation is completed without errors, the indicator is set off.

All other columns in the *Resulting Indicators* field must be blank.

If no indicator is specified in columns 56 and 57 and the FEOD operation fails, an error code is displayed to the operator. Depending on the error condition, the program is terminated or the operation is retried after the operator's response.

INDICATOR SETTING OPERATIONS

The operation codes in this group allow you to set on or off any of the indicators (01 through 99). The two operation codes are:

Op Code	Meaning
SETON	Set on indicators
SETOF	Set off indicators

The entry in the *Resulting Indicators* field represents the indicators (01 through 99) that are to be set to the condition specified by the operation code. The numbers of up to three indicators can be entered.

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	SETON	Blank	Blank	Required

Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
Optional	Blank	SETOF	Blank	Blank	Required

Chapter 7. Contents

Chapter 7. Compile-Time Tables and Self-Check

Compile-time tables must be included in the source program data set. During program compilation, these tables become part of the object program data set.

Compile-time tables are usually used for small tables or tables containing static information.

Compile-time tables can be categorized as follows:

- Data tables
- File translation tables
- Alternate collating sequence tables

All file translation tables and alternate collating sequence tables must be compile-time tables. Data tables can be either compile-time tables or tables in a data set separate from the program.

Note: The term *data table* in this chapter refers to the function provided by arrays in RPG subroutines.

DATA TABLES

All data tables used during the execution of a program must be in storage throughout the program. The organization of the data in data tables is the same for both compile-time data tables and tables in a data set.

Data tables have the following characteristics and requirements:

- All entries in a table are the same length.
- All the entries in a table are character data, or all are numeric data.
- Each data table is named in a table description statement (on the A-specification).
- All tables described subordinate to one file description statement have the same number of entries.
- Every entry in a table is in a separate record.

Defining Compile-Time Data Tables

Compile-time data tables can be coded on any specification. However, the length of the records (or the number of columns used on the specification) cannot exceed 80 positions. The compile-time data tables must follow all Z-, A-, and C-specifications.

The definitions of compile-time data tables must be consistent with both the file description statement for the file with which the tables are associated and the table description statements on the A-specification. Each file must be named in a title record. The title record must be followed by a quantity of records equal to the number of entries specified as the parameter for the NUMENT keyword in the file description statement. If fewer records are supplied than specified by the NUMENT keyword, the extra entries are filled with blanks for character data or filled with zeros for numeric data. If no data records are supplied following the title record, the table is filled with blank entries (for character data) or zeros (for numeric data). There also must be entries in each record of each table description statement subordinate to the file description statement.

The syntax for the title line follows.

Columns 1 and 2 must be **.

Column 3 through 8 must contain CTDATA.

Column 9 must be blank.

Columns 10 through 18 must contain the name of the file with which the tables are associated.

Columns 19 through 80 can contain comments.

In the records following the title, the first columns contain the entries for the first table described subordinate to the file description statement. The entries for the second table immediately follow the end of the first table entries.

The first record following the title record contains the first entries of all tables associated with the file. The next record contains the second entries of all the tables associated with the file, and so forth.

Example

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								Source Document	Page of
Operator	Date		Key									

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/K/R/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Checks=CHECK (code . . .)

Auto Dup	AD	Mandatory Entry	ME
Auto Skip	AS	Mandatory Fill	MF
Blank Check	BC	Rt Adj–Blank Fill	RB
Bypass	BY	Right to Left	RL
Bypass on Verify	BV	Rt Adj–Zero Fill	RZ
Data Required	DR	Self Check	mxx
Dup Disable	DD	n=C/G (Check/Gen)	
Field Exit Required	FE	xx=Modulus	
Lower Case	LC		

Functions

ADD (name)	PMT (prompt)
AUXDUP (name)	RANGE (low high)
AUXST (name)	RANGET (table [index])
COMP ([1]test fld1[3]@ fldn (indicator))	RESET (+TOTn)
DSPATR ([2]attr [2])	SEQ ([1]test)
EDTCDE (code [float])	SETOF (ind)
ERROR (code ['message'])	SETON (ind)
EXSR (subroutine)	SHIFT ([4]shift)
INSERT (fld1[3]@ fldn)	SUB (name)
LOOK (table (index))	SUBST (table1 table2 [index])
[1]test=EQ,GE,GT,LE,LT,NE	TADD (+TOTn)
[2]attr=BL,CA,CS,HI,ND,RI,UL	TSUB (+TOTn)
[3]@=+,–,*,/	XCHK (table index1 index2)
[4]shift=A,D,H,N,V,W,X,Y	'literal'

```
     1 A
     2 A            F INSIDE                  13                NUMENT(8)
     3 A            T NAMET                   8
     4 A            T NUMT                    5  2
     5 A
**CTDATA INSIDE  COMMENTS CAN START IN COLUMN 18 OF THE TITLE LINE.
AL     7 A         00723
FRED   8 A         01462
FREDA  A           02730
JOE    10 A        03129
SAM    11 A        00239
SUE    12 A        01000
TAMMY  A           13775
WILLIAM 00023
       15 A
          A
          A
          A
          A
          A
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

Note: Data tables can be coded on any specification form. When the specifications are entered, the preprinted entries on the form (such as the A in column 6) are ignored. The compile-time data tables must follow all Z-, A-, and C-specifications.

FILE TRANSLATION TABLES

File translation tables provide a means for translating data characters to (or from) EBCDIC from (or to) another code. (The EBCDIC codes are shown in *Appendix A*.) An individual character can be changed, or any and all characters can be changed or exchanged. File translation tables are always specified as compile-time tables.

File translation tables can be defined on any specification. However, no more than 80 columns can be used for each line.

Each file translation table must be identified by a title line. The syntax for a title line for a file translation table follows.

Columns 1 and 2 must be **.

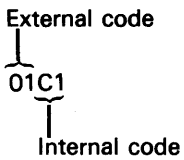
Columns 3 through 8 must contain FTRANS.

Columns 9 through 18 must be blank.

Columns 19 through 80 can contain comments.

The lines (records) that follow the title must contain the name of the file to be translated in columns 1 through 8. The name, which must be the name used in the name field of the file description statement, is required on each line used to define the table. The translation values must start in column 9.

Each translation value is specified as a pair of hexadecimal values representing the external and the internal codes for the translation. The internal code is the value that is used by the program. The external code is the value recorded in the data set. For example, a translation from hexadecimal 01 in a data set to an uppercase A for processing requires the following entry:



Example

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A.

Job		Keying Instruction	Graphic	Description	Page of
Operator	Date		Key		

Sequence (1–5)	Form Type (6)	Name Type (7)	Format ID (1-9, A0-Z9) (8–9)	Job/Format/ Subroutine Name (10–17)	Reserved (18–19)	Repeat (1-9, N) (20)	Mode (E/R) (21)	AND(A) (22)	Test Conditions: Position to be Tested (*POSnnnn) (23–30)	Reserved (31–32)	Condition (33–34)	Character to Test for ('C') (35–37)	Reserved (38–44)	Next Format ID (0-9, A0-Z9) (45–46)	Reserved (47–54)	Options (55–80)

Options — Job Line: CFILE (data set), DATE (•DMY/•YMD), EDITC ([cuptd]), ENTRATR (attr...), EXITATR (attr...), JOBOPT ([•NOPMT][•NOOPEN]), PRTFILE (data set), SHARE (names), SHARER (names), STATUS (name), TFILE (data set [delfreq]), attr BL,CS,HI,ND,RI,UL

Options — Entry Lines: CLRL (number), EOJ [(job dev [•PASS])], SLNO (line), WRITE (name)

```
**FTRANS          EXCHANGES DIGITS 'TO SCRAMBLE A FILE OF NUMERIC DATA
SCRAMBLEF8F0F3F1F6F2F4F3F7F4F0F5F9F6F1F7F5F8F2F9
```

(Preprinted on the remaining lines: sequence 3–15, Z in column 6, *POS in columns 23–26, EQ in columns 33–34, ' in columns 35 and 37.)

This translate table can only be used with data to and from a file named SCRAMBLE. The translation scrambles the hexadecimal representations of decimal digits (0 through 9).

*Number of sheets per pad may vary slightly.

Note: Translation tables can be coded on any specification form. When the specifications are entered, the preprinted entries on the form (such as the Z in column 6) are ignored.

ALTERNATE COLLATING SEQUENCE TABLES

An alternate collating sequence table can be used to change the collating sequence for compare operations on character data. An alternate collating sequence does not affect compare operations on numeric data and does not affect any operations except those resulting from the use of the COMP keyword in a field description statement and the COMP or CABxx operation codes in a calculation statement. Only one alternate collating sequence table can be coded in a program.

Like file translation tables, an alternate collating sequence table can be defined on any form, with no more than 80 columns used for each line.

An alternate collating sequence table is identified by a title line. The syntax for the title line follows.

Columns 1 and 2 must be **.

Columns 3 through 8 must contain ALTSEQ.

Columns 9 through 18 must be blank.

Columns 19 through 80 can contain comments.

Lines following the title line specify changes in the collating sequence. Each of these lines must contain ALTSEQ in columns 1 through 6. Columns 7 and 8 must be blank. The specifications of change values must start in column 9. If ASCII is coded in columns 9 through 13, the program uses the predefined ASCII table. (The ASCII table must have been specified when the system was configured.)

Each change value is specified as a pair of hexadecimal values representing the *normal* and *change to* character codes for the collating sequence. For example, and uppercase A is equal to an uppercase C in compare operations if only the following is entered:

```
C1C3
```

A collates as C.

To exchange positions between an A and a C in the collating sequence, two entries are required.

```
C1C3C3C1
```

A collates as C, and C collates as A.

Example

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A.

Job		Keying Instruction	Graphic								Description	Page of
Operator	Date		Key									

Z

Sequence	Form Type	Name Type	Format ID (1-9, A0-Z9)	Job/Format/ Subroutine Name	Reserved	Repeat (1-9, N)	Mode (E/R)	AND(A)	Test Conditions: Position to be Tested (*POSnnnn)	Reserved	Condition	Character to Test for ('C')	Reserved	Next Format ID (0-9, A0-Z9)	Reserved	Options: Job Line / Entry Lines
1 2 3 4 5	6	7	8 9	10 11 12 13 14 15 16 17	18	19	20	21	22 23 24 25 26 27 28 29 30	31 32	33 34	35 36 37	38 39 40 41 42 43 44	45 46	47 48 49 50 51 52 53 54	55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1	Z								*POS		EQ	' '				
	Z			**ALTSEQ COMMENTS CAN START IN COLUMN 19 OF THE TITLE LINE												
	Z			ALTSEQ ASCII												
4	Z								*POS		EQ	' '				
5	Z								*POS		EQ	' '				
6	Z								*POS		EQ	' '				
7	Z								*POS		EQ	' '				
	Z			**ALTSEQ INVERTS THE SEQUENCE OF Ø THROUGH 9 FOR CHARACTER COMPARES.												
	Z			ALTSEQ FØF9F9F0F1F8F8F1F2F7F7F2F3F6F6F3F4F5F5F4												
1 0	Z								*POS		EQ	' '				
1 1	Z								*POS		EQ	' '				
1 2	Z								*POS		EQ	' '				
1 3	Z								*POS		EQ	' '				
1 4	Z								*POS		EQ	' '				
1 5	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				

Options — Job Line: CFILE (data set) | DATE (+DMY +YMD) | EDITC ([cuptd]) | ENTRATR (attr...) | EXITATR (attr...) | JOBOPT ([+NOPMT][+NOOPEN]) | PRTFILE (data set) | SHARE (names) | SHARER (names) | STATUS (name) | TFILE (data set [,delfreq]) | attr: BL,CS,HI,ND,RI,UL

Options — Entry Lines: CLRL (number) | EOJ [(job dev [+PASS])] | SLNO (line) | WRITE (name)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

Note: Alternate collating sequence tables can be coded on any specification form. When the specifications are entered, the preprinted entries (such as the Z in column 6) are ignored.

SELF-CHECK PROCESSES

The use of self-check fields provides protection against clerical and keying errors by performing a degree of character sequence validation during the entry of data. The self-check process involves the application of an algorithm to the characters entered in the self-check field to calculate check characters. During self-check generate operations, the check characters are inserted into specific positions of the self-check field. During self-check check operations, the calculated check characters are compared with the characters entered in the specified check-character positions. Mismatches between the entered check-characters and the calculated check-characters during check operations cause an error that locks the keyboard and displays an error code, alerting the operator to the invalid entry.

In DE/RPG, you can use either of two standard algorithms (IBM modulus 10 or IBM modulus 11), or you can use customized algorithms for self-check operations by defining them to DE/RPG according to the process described in the following sections. Note that this discussion is *not* intended to help you develop a customized self-check algorithm; this discussion only explains how to define an algorithm to DE/RPG.

Rules and Terms

Defining an algorithm in DE/RPG is simple, but understanding the application of an algorithm to the characters in a self-check field requires familiarity with some fundamental rules for self-check and terms used to describe the self-check process.

- A self-check field can contain a maximum of 32 character positions including either one or two check-characters.
- Check-character positions may be located anywhere within a self-check field, but if two check-characters are used, they must be adjacent.
- The foundation of a self-check field is all the characters in the field except those in the designated check-character positions.
- The term *modulus* refers to the base (or radix) of a number system. Within the conventions for arithmetic results, the largest number in any position is always one less than the base. For example:

  $3_{base4} + 1_{base4} = 10_{base4}$

  When the number in a position reaches the value of the base, a carry into the next position is required.

- The modulus for a self-check algorithm can be any whole number from 2 through 127. The number specified as the modulus is used as the base in arithmetic results.

- Values called *weights* are assigned to positions of the self-check field. The numeric values used for each foundation character are multiplied by the weight assigned to the corresponding position. The products are resolved to a base equal to the specified modulus.

- Any number less than the number specified for the modulus can be assigned as a weight. The weight for check-character positions and any positions to be ignored must be zero.

- The modulus of a value that is expressed with a base equal to the modulus is the number in the units position. For example, the 7 modulus of 23_{base7} is 3.

- The meaning of the term *digit* as used in this manual in describing self-check processes is expanded to mean the number in a single position of an arithmetic result.

For example:

$$\begin{array}{r} 8_{base20} \\ +9_{base20} \\ \hline 17_{base20} \end{array}$$

The sum of 8 plus 9 does not exceed the value of the base and does not cause a carry into the next position. Therefore, the number 17 is considered a single digit.

To avoid confusion when values are expressed to a base other than base 10, this manual uses the following convention to show the positional values of numbers. The columns as shown below each represent a position.

$$|X^3|X^2|X^1|X^0$$

The positions from right to left represent units, base (called *tens*), base squared (called *hundreds*), base cubed (called *thousands*), and so forth. With this convention, values expressed to a base greater than 10 can be more easily understood.

For example, the expression 123_{base25} is ambiguous, yet the decimal value of this same series of numerals is clear when the numbers are placed within columns.

$$\begin{array}{l} |625|25|1| \\ |\ 1\ |\ 2\ |\ 3\ |_{base25} = 678_{base10} \\ \quad |12\ |\ 3\ |_{base25} = 303_{base10} \\ \quad |\ 1\ \ |23\ |_{base25} = \ 48_{base10} \end{array}$$

The Self-Check Process

In DE/RPG, the process for calculating check-characters for self-check fields involves a series of six major steps. Each successive step in the process operates upon the results from the preceding step. The steps are shown in Figure 7. The blocks contained in steps 3 through 6 represent different procedures that can be used. (A procedure from each step is used.)

The algorithm specified for a self-check operation determines:

- The value used as the modulus.
- The weights applied to the value conversions for the foundation characters.
- The location of the check-characters within the self-check field.
- The procedure used in each of the last four steps of the self-check process.

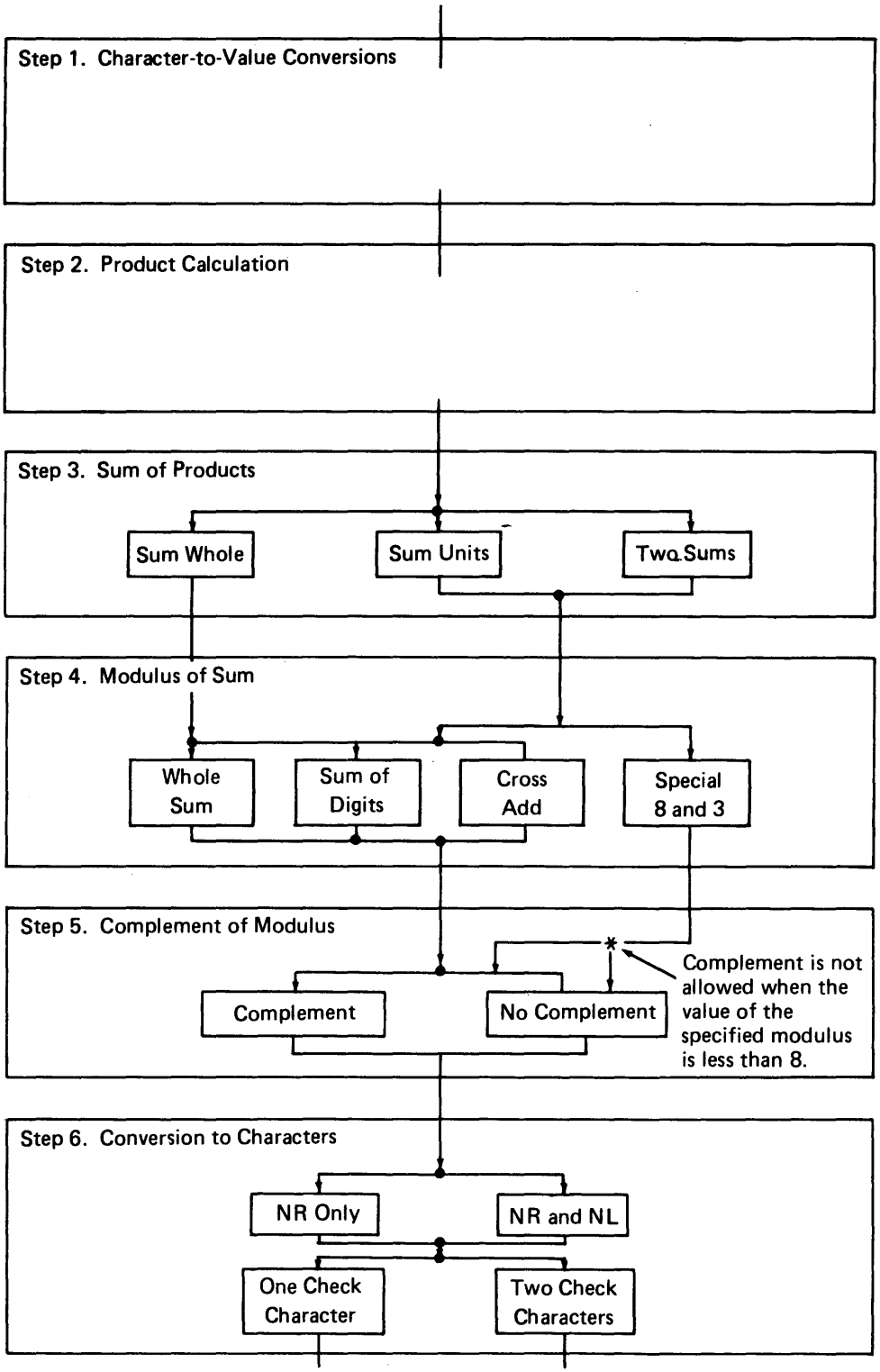


Figure 7. Calculating Check Characters

The following paragraphs describe procedures that you can follow to obtain the results for each step. Remember that the results from the first step are used by the process in the second step, and so forth.

Step 1. Character to Value Conversions

The function of this step is to convert the EBCDIC value of each character in the self-check field into a value for use in step 2 of the self-check process. (Every character is internally represented in the 5280 in EBCDIC.) To derive the same results as the 5280, do the following:

1. List each character in the self-check field in proper sequence.

2. List the EBCDIC value for each character (express in hexadecimal).

3. Discard the zone portion (left-hand character) of each hexadecimal representation.

4. Change any non-numeric character remaining (A through F) to zero.

The values now shown are the result of the character to value conversion in step 1.

Example:

1. Characters	1	C	*	8	3	0
2. EBCDIC Value (in Hex)	F1	C3	5C	F8	F3	F0
3. Discard Zones	1	3	C	8	3	0
4. Change A through F to 0	1	3	0	8	3	0

Step 2. Product Calculation

This step involves the multiplication of each value from step 1 by a weight in corresponding position. Each product is resolved to a base equal to the value of the modulus. The instructions for this step are:

1. Multiply each value from step 1 by the corresponding weight.

2. Resolve the product to the base equal to the modulus.

Example:

For this example, assume the following:

Modulus= 23
Weights= 11 21 6 19 5 0

check-character position

1. Values from step 1	1	3	0	8	3	0
Weights	x11	x21	x6	x19	x5	x0
	11	63	0	152	15	0

2. Resolve products |0|11| |2|17| |0|0| |6|14| |0|15| |0|0|$_{base23}$

Step 3. Sum of Products

This step uses the products from step 2. Any one of three different procedures can be specified for use in this step. The procedures, as shown in Figure 6, are:

- Sum whole
- Sum units
- Two sums

The results of these procedures (and also the results of following steps) are called NR and NL, which mean number right and number left respectively. To avoid confusion, references to NR and NL are qualified to indicate the step that produced them.

Sum Whole: The sum whole procedure accumulates a single sum of the products from step 2. The instructions for this procedure are expressed in these equations.

NR (from step 3)= product + product + product + ...
NL (from step 3) = 0

Example:

Products from step 2=
$|0|11|\ |2|17|\ |0|0|\ |6|14|\ |0|15|\ |0|0|_{base23}$

NR (from step 3)= $|0|11|+|2|17|+|0|0|+|6|14|+|0|15|+|0|0|=|11|_{base23}$
NL (from step 3) = $|0|\ 0|_{base23}$

Sum Units: The sum units procedure accumulates a single sum of the units position digits of all products from step 2. The instructions for this procedure are expressed in these equations:

NR (from step 3)= unit digit + unit digit + unit digit +...
NL (from step 3) = 0

Example:

Products from step 2 $|0|11|\ |2|17|\ |0|0|\ |6|14|\ |0|15|\ |0|0|_{base23}$

NR (from step 3)=
$|11|+|17|+|0|+|14|+|15|+|0|=|2|11|_{base23}$
NL (from step 3) = $|0|0|_{base23}$

Two Sums: The two sums procedure accumulates a sum of the units position digits of all products from step 2 and accumulates another sum of the tens position digits of the products from step 2. The instructions of this procedure are expressed in these equations:

NR (from step 3)= units digit + units digit + units digit + ...
NL (from step 3) = tens digit + tens digit + tens digit + ...

Example:

Products from step 2 $|0|11|\ |2|17|\ |0|0|\ |6|14|\ |0|15|\ |0|0|_{base23}$
NR (from step 3)= $|11|+|17|+|0|+|14|+|15|+|0| = |2|11|_{base23}$

NL (from step 3) = $|0|+|2|+|0|+|6|+|0|+|0| = |0|8|_{base23}$

Step 4. Modulus of Sum

This step uses the NR and NL values from step 3. Within step 4, there are four different procedures as indicated in Figure 7. These procedures are:

- Whole sum
- Sum of digits
- Cross add
- Special 8 and 3

The results of each of these procedures are expressed as NR and NL. The base of the results can be ignored.

Whole Sum: The whole sum procedure takes the modulus of the sum of the NR and NL values from step 3. The instructions for this procedure are expressed in these statements:

NR (from step 4)= modulus of sum of NR (from step 3) plus NL (from step 3)
NL (from step 4) = 0

Example:

This example uses the results from the two sums procedure in step 3.

NR (from step 3)= $|2|11|_{base23}$
NL (from step 3) = $|0|\ 8|_{base23}$
Sum of NR + NL $|2|19|_{base23}$

NR (from step 4)= modulus of $|2|19|_{base23}$ = 19
NL (from step 4) = 0

Sum of Digits: The sum of digits procedure takes the modulus of the sum of the digits of the sum of NR and NL from step 3. The instructions for this procedure are expressed in these statements:

NR (from step 4)= modulus of sum of digits of sum of NR (from step 3) plus NL (from step 3).
NL (from step 4) = 0

Example:

This example uses the results from the two sums procedure example in step 3.

NR (from step 3)= $|2|11|_{base23}$
NL (from step 3) = $|0|\ 8|_{base23}$
Sum of NR + NL $|2|19|_{base23}$
Sum of digits of sum $|21|_{base23}$
NR (from step 4)= modulus of $|21|_{base23}$ = 21
NL (from step 4) = 0

Cross Add: The cross add procedure takes the modulus of two values. The instructions for this procedure are expressed by these statements:

NR (from step 4)= units position of sum of even power position digits of sum of NR (from step 3) plus NL (from step 3)

NL (from step 4) = modulus of sum of the carry from above equation plus the sum of the odd power position digits of the sum of NR (from step 3) plus NL (from step 3)

Even power positions are units and hundreds; odd power positions are tens and thousands.

Example:

For this example base 6 is used to provide extra digits for the illustration. Assume the following values:

Modulus = 6
NR (from step 3)= $|1|4|5|_{base6}$
NL (from step 3) = $|3|5|4|_{base6}$
Sum of NR + NL $|5|4|3|_{base6}$

$5 + 3 = 8_{base10} = |1|2|_{base6}$
$4 + 1 = |5|_{base6}$

NR (from step 4)= 2
NL (from step 4) = 5

Special 8 and 3: The special 8 and 3 procedure takes the 8 modulus of the value of NR (from step 3) and takes the 3 modulus of the value of NL (from step 3). The 8 modulus becomes NR, and the 3 modulus becomes NL. (The values must be expressed to base 8 or base 3.)

Example:

For this example, assume the following values:

Modulus= 23
NR (from step 3)= $|2|11|_{base23}$
NL (from step 3) = $|0|8|_{base23}$

NR (from step 4)= 8 modulus of $|2|11|_{base23}$

$|2|11|_{base23}$ = $|5|7|_{base10}$ = $|7|1|_{base8}$

NR (from step 4)= 8 modulus of $|7|1|_{base8}$=1

NL (from step 4) = 3 modulus of $|0|8|_{base23}$

$|0|8|_{base23}$ = $|8|_{base10}$= $|2|2|_{base3}$

NL (from step 4) = 3 modulus of $|2|2|_{base3}$=2

The special 8 and 3 procedure cannot follow the sum whole procedure in step 3.

Step 5. Complement of Modulus

Within this step, there are only two procedures available. The procedures are:

- Complement
- No complement

Again, these procedures use the results of the preceding step.

Complement: The complement procedure takes the modulus of a modulus complement of the values of NR (from step 4) and NL (from step 4). The instructions for this procedure are expressed by the following equations:

NR (from step 5)= the modulus of [modulus value – NR (from step 4)]

NL (from step 5) = the modulus of [modulus value – NL (from step 4)]

Example:

In this example, the results from the example for the sum-of-digits procedure in step 4 are used.

Modulus= 23
NR (from step 4)= 21
NL (from step 4) = 0

NR (from step 5)= the modulus of (23 – 21)=the modulus of
2 = 2

NL (from step 5) = the modulus of (23 – 0)=the modulus of
23 = 0

The complement procedure cannot follow the special 8 and 3 procedure in step 4 unless the specified modulus is greater than or equal to 8.

No complement: The no complement procedure performs no arithmetic operation. The equations for this procedure are:

NR (from step 5) = NR (from step 4)

NL (from step 5) = NL (from step 4)

Step 6. Conversion to Characters

The conversion of the results from step 5 into check-characters is the last step in check-character calculation. Two different procedures are available. Both procedures can yield two check characters. These procedures are:

- NR only
- NR and NL

Also contained within this last step is the selection of either a single check-character or two check-characters.

NR Only: The NR-only procedure ignores NL (from step 5) and uses only NR (from step 5) for the conversion. The instructions for the conversion are expressed in the following statements:

Right check-character	=	Hexadecimal F zone with rightmost numeral in NR (from step 5)
Left check-character	=	Hexadecimal F zone with the hexadecimal representation of the remaining numerals in NR (from step 5)

Example:

For this example, assume that NR (from step 5) is 127, which happens to be the highest possible numerical value from step 5.

Right check-character	=	Hexadecimal F7, which displays as the graphic character 7
Left check-character	=	Hexadecimal FC, which has no graphic translation. It should be noted that when a value of 99 or less is assigned as the modulus, both check-characters converted via the NR-only procedure are always numbers

When only one check-character is specified, only the right check-character is used.

NR and NL: The NR and NL procedure uses both NR (from step 5) and NL (from step 5). The equations for this procedure are:

Right check-character =	Hexadecimal F zone with the hexadecimal representation of the 16 modulus of NR (from step 5)
Left check-character =	Hexadecimal F zone with the hexadecimal representation of the 16 modulus of NL (from step 5)

Example:

For this example, assume these values:

NR (from step 5)= 95
NL (from step 5) = 49

16 modulus of 95= 15. The hexadecimal representation of 15 = F

16 modulus of 49= 1. The hexadecimal representation of 1 = 1

Right check-character =	Hexadecimal FF, which displays as ■
Left check-character =	Hexadecimal F1, which displays as the number 1

When only one check-character is specified, only the right check-character is used.

Defining a Self-Check Algorithm

You can use any specification to code a self-check algorithm, but no more than 80 positions in each line can be used.

Columns 1 through 11 of the first line must contain the algorithm identification in the form **SLFCHK *nn*, in which *nn* is any number from 01 through 09 and 12 through 99. These numbers are those used to reference the self-check operations in field description statements on the A-specifications. (The numbers 10 and 11 are reserved and correspond to IBM modulus 10 and IBM modulus 11 respectively.)

The remainder of the first line (columns 12 through 80) can be used for comments. The following lines are available for the free-form entry of self-check keywords and their parameters.

The keywords used to define a self-check algorithm in DE/RPG are:

Keyword	Purpose
MOD	For assigning the modulus used in check-character calculations
DISP	For specifying the location of the check-characters within the self-check field
WEIGHTS	For specifying the values for use as weights during step 3 (product calculations)
OPT	For specifying the options to be used in steps 3 through 6 of the application of the algorithm

The standard rules for keywords and parameters apply, including the requirement for continuation characters when a keyword and its parameter are not started and completed on the same line.

MOD(*number*)

The MOD keyword assigns the value for use as the modulus during the calculation of check characters. The number parameter is required and can be any whole number from 2 through 127. There need be no numerical relationship between the value specified for the modulus and the number used as the algorithm identifier (**SLFCHK *nn*).

If you define a self-check algorithm, use of the MOD keyword is required.

DISP(*number*)

The DISP keyword allows you to specify the relative location of the rightmost check character in the self-check field. The parameter *number* represents the displacement from the rightmost character in the self-check field to the rightmost check character. When the rightmost check character is located in the rightmost position of the self-check field, the correct specification for *number* is 1.

If a zero is specified as the *number* parameter, the DISP keyword specifies that there is no check character in the field. In this case, the check digit that is generated by the self-check algorithm must be zero.

If the DISP keyword is not used, the default value of 1 is assumed for the displacement.

Note: The field length specified on the A-specification includes the self-check character location. The operator must supply data for all the positions in the field, including this location. Any character entered in the self-check location is replaced by the program when CHECK(Gxx) is specified.

WEIGHTS(*n n n n . . .*)

The WEIGHTS keyword identifies the weights that are to be used in deriving the products in step 3. (Refer to Figure 8.) Up to 32 weights can be specified as multiple parameters. The parameter *n* can be any whole number that is less than the value assigned to the modulus. The weights specified in positions corresponding to the check-character positions must be 0. Zero may be specified corresponding to any other position that should be ignored in check-character calculations.

The right end of the multiple parameter for the WEIGHTS keyword is aligned with the right end of the self-check field during the application of the algorithm. When there are more weights specified than there are characters in the self-check field, the weights without corresponding characters are ignored.

The WEIGHTS keyword is required for all self-check algorithm definitions.

OPT('*string*')

The OPT keyword allows you to specify the procedures to be used in steps 3 through 6 during the calculation of check characters. (Refer to Figure 8.) The parameter '*string*' is a string of up to 5 character positions enclosed by apostrophes.

Each position is significant as follows:

- The first position represents the procedure for step 3.
- The second position represents the procedure for step 4.
- The third position represents the procedure for step 5.
- The fourth and fifth positions represent the procedures for step 6.

The characters in '*string*' have the following meanings.

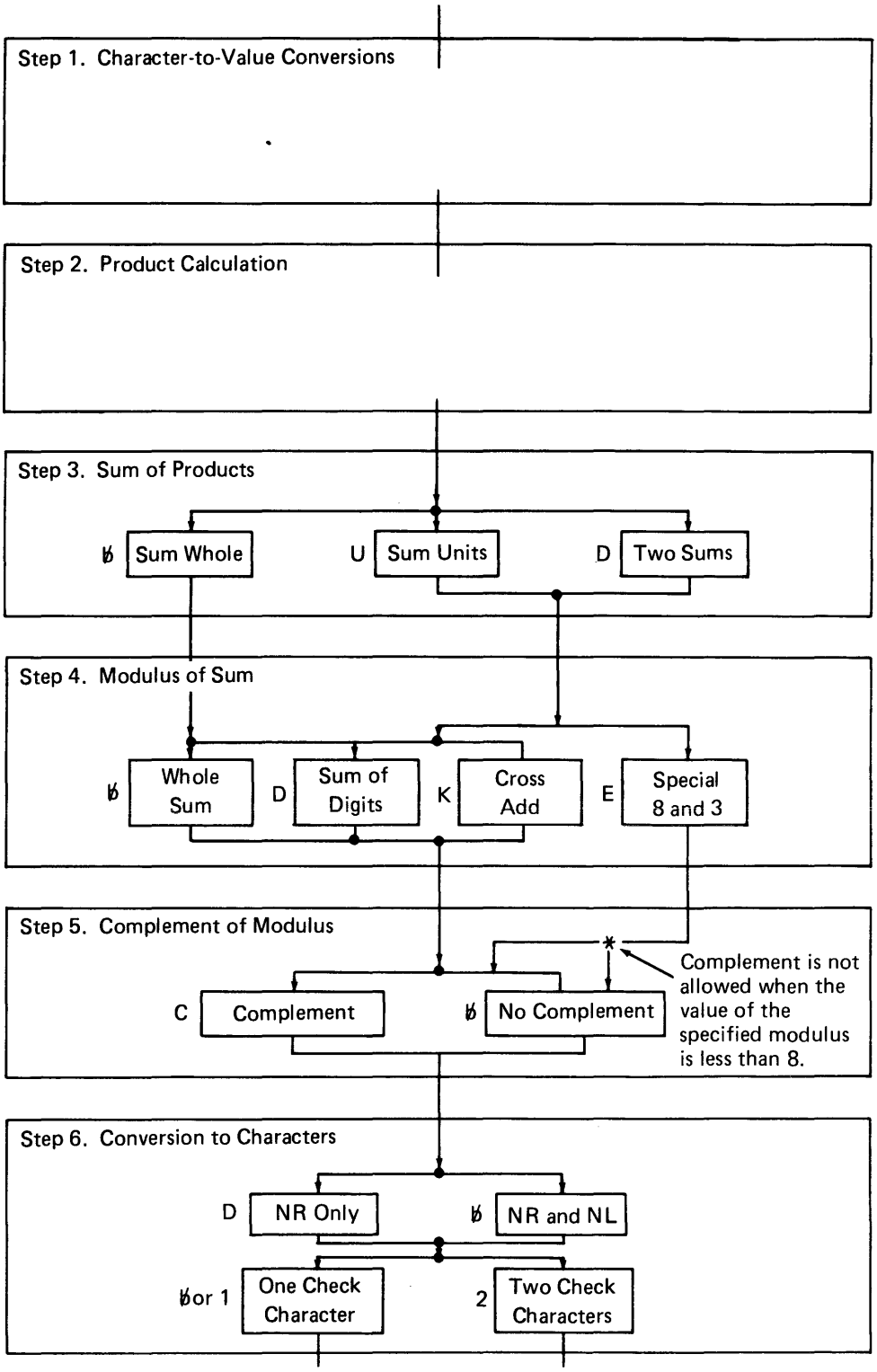


Figure 8. Procedures for OPT Keyword

Position 1 (step 3)	Blank	=	Sum Whole
	U	=	Sum Units
	D	=	Two Sums
Position 2 (step 4)	Blank	=	Whole Sum
	D	=	Sum of Digits
	K	=	Cross Add
	E	=	Special 8 and 3
Position 3 (step 5)	Blank	=	No complement
	C	=	Complement
Position 4 (step 6)	Blank	=	One check-character
	2	=	Two check-characters
Position 5 (step 6)	Blank	=	NR and NL
	D	=	NR Only

A position in '*string*' can be omitted if and only if that position and all positions to the right of that position contain blanks. If all the procedures to be used are represented by blanks, use of the OPT keyword in the algorithm definition is not required.

Examples

The following examples of self-check algorithm definitions need not be entered in a source program data set. The examples show the codes that define IBM modulus 10 and IBM modulus 11, which are both already available for use in DE/RPG programs and should not be redefined.

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

Printed in U. S. A.

Job		Keying Instruction	Graphic								Description	Page of
Operator	Date		Key									

Z

Sequence	Form Type	Job/Format/Subroutine: Name Type	Format ID (1-9, A0-Z9)	Name	Reserved	Repeat (1-9, N)	Mode (E/R)	AND(A)	Test Conditions: Position to be Tested (*POSnnnn)	Reserved	Condition	Character to Test for ('C')	Reserved	Next Format ID (0-9, A0-Z9)	Reserved	Options: Job Line / Entry Lines
1	Z								*POS		EQ	' '				
		**SLFCHK 10														
		MOD(10)DISP(1)WEIGHTS(2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 +														
4	Z		2 0 2 0 2 0)	OPT('D C')												
5	Z								*POS		EQ	' '				
6	Z								*POS		EQ	' '				
		**SLFCHK 11							*POS		EQ	' '				
		MOD(11) DISP (1) OPT (' C D')														
		WEIGHTS(2 7 6 5 4 3 2 7 6 5 4 3 2 7 6 5 4 3 2 7 6 5 4 3 2 7 6 5 4 3 2 0)														
10	Z								*POS		EQ	' '				
11	Z								*POS		EQ	' '				
12	Z								*POS		EQ	' '				
13	Z								*POS		EQ	' '				
14	Z								*POS		EQ	' '				
15	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				
	Z								*POS		EQ	' '				

Job Line: CFILE (data set) · DATE (*DMY · *YMD) · EDITC ([cupid]) · ENTRATR (attr ...) · EXITATR (attr ...) · JOBOPT ([*NOPMT][*NOOPEN]) · PRTFILE (data set) · SHARE (names) · SHARER (names) · STATUS (name) · TFILE (data set [deltreq]) · attr: BL CS HI ND RI UL

Entry Lines: CLRL (number) · EOJ [(job dev [*PASS])] · SLNO (line) · WRITE (name)

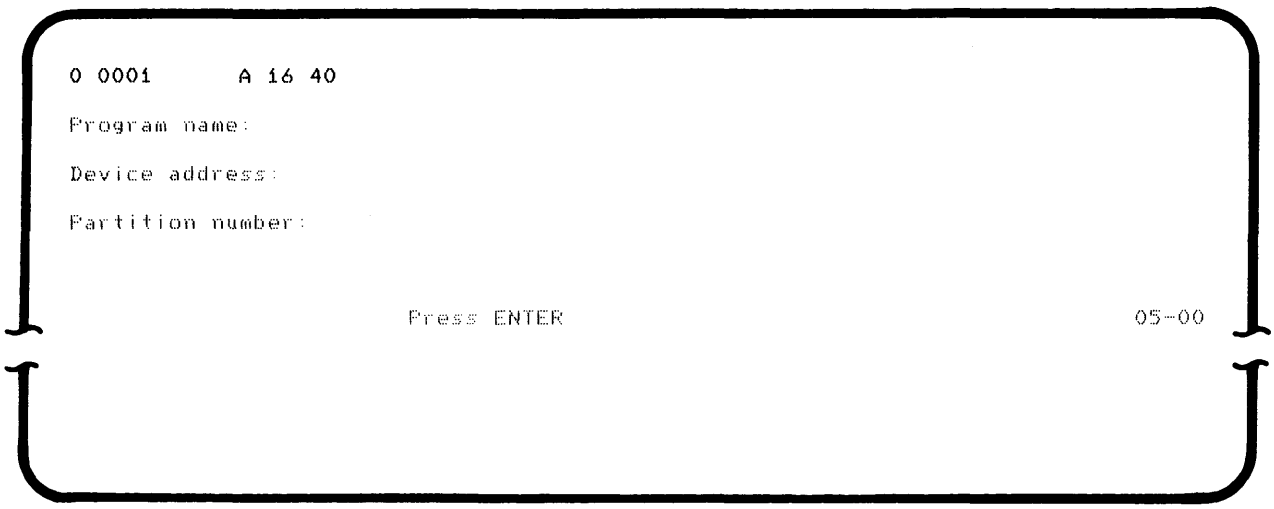
*Number of sheets per pad may vary slightly.

Note: Self-check algorithm definitions can be coded on any specification form. When the specifications are entered, the preprinted entries on the form (such as the Z in column 6) are ignored.

Chapter 8. Contents

The IBM DE/RPG program product compiler is supplied on diskette by IBM. The compiler diskette must be inserted into a diskette drive, and the program can then be loaded into a partition in the normal manner in response to a load prompt.

Note: You should make a backup copy of the compiler diskette. The diskette on which you copy the compiler cannot be a format 1 diskette. Any other format can be used. A backup is important when data on the primary diskette becomes inaccessible due to loss, diskette wear, or diskette damage. The compiler (and other programs) can be copied to a user-created *system diskette*. Use of system diskette lessens the number of diskette insertions and removals performed during the course of a day.



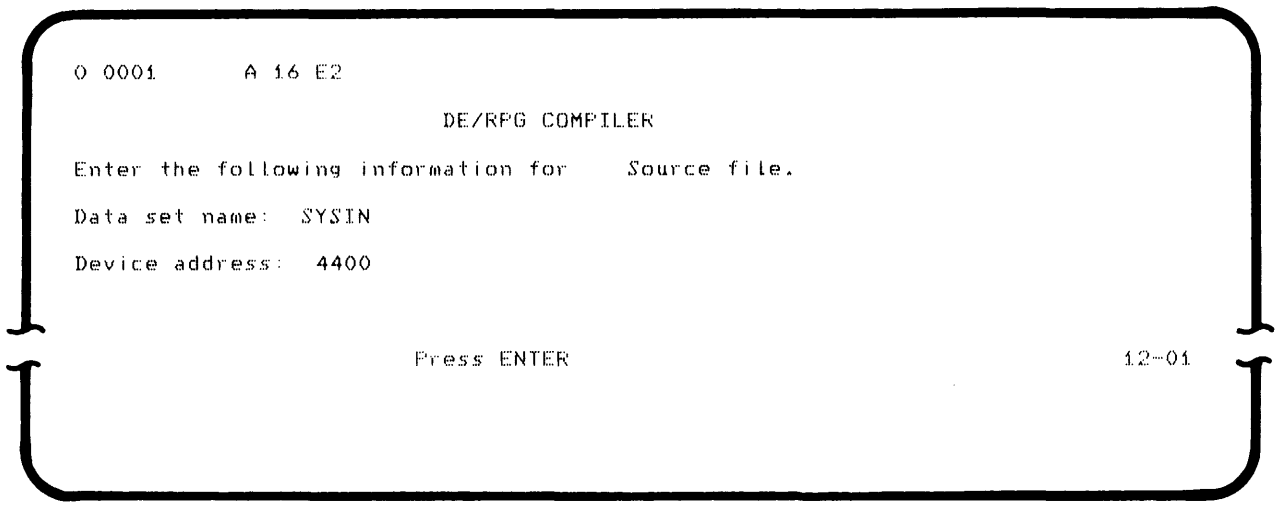
SYSDERPG

The name of the compiler program is SYSDERPG. The compiler requires a partition of at least 9 K bytes. The diskette that contains the compiler must *not* be removed until the compilation is completed.

As the compiler begins its operation, it prompts the user to specify the files and options to be used. These prompts are shown on the following pages. In general, up to 16 characters may be entered in a data set name field and either a 4-character physical device address or a 2-character logical device address may be entered in a device address field. Entries that are shorter than the field length should be entered left-adjusted and followed by blanks.

COMPILER DATA SET

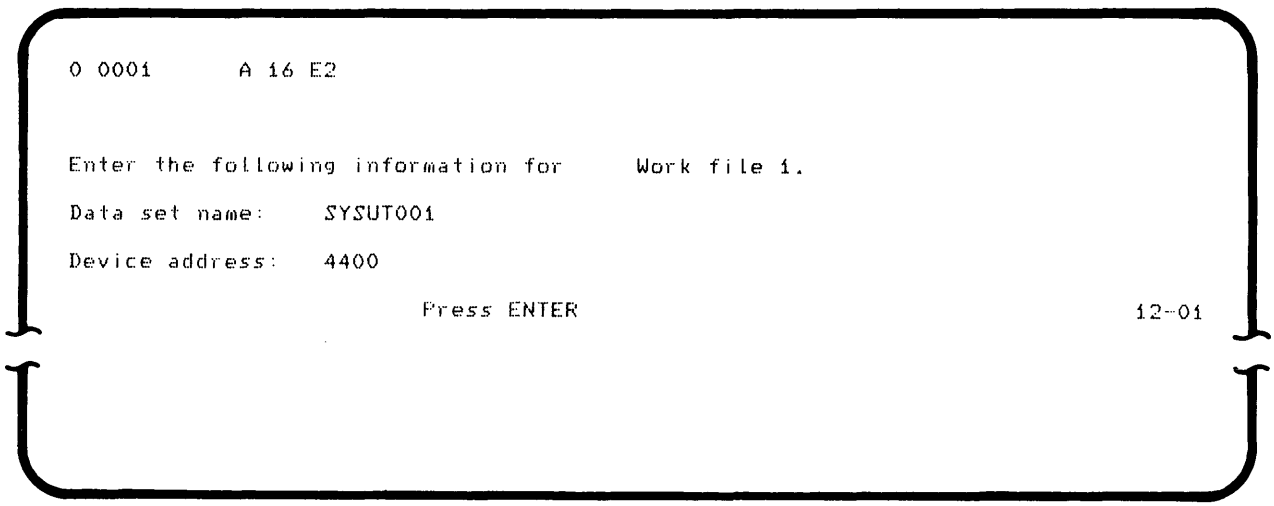
The fundamental input or data source for the compiler is a source program data set, which contains the coded statements that describe a program. When the compiler is loaded, the system prompts for information about the source program data set.



Insert the source program data set into a diskette drive and respond to the prompting messages. The data set name and device address displayed in the prompting messages can be changed. If the default value for device address is used and it cannot be resolved using the resource allocation table from the system configuration, the device address defaults to the address of the device from which the compiler was loaded. In that case, the compiler and the source program data set must be on the same diskette because both must remain available to the system throughout the compilation.

To reduce the amount of read/write storage required for compiling a program, the compiler uses temporary storage on diskette. This temporary storage, or work file, receives intermediate output from the compiler. The intermediate output is later used as input during subsequent phases of the compilation. Upon completion of the compilation, the work space data sets are cleared of all intermediate data.

The system prompts for information about the required work spaces. The first of these prompting messages is for information about work file 1.



As before, the displayed name and device address can be changed. If the defaults are used, the work space is assigned on the diskette with the compiler.

If the specified file does not exist on the specified device, you have the option of reentering the data set name and device address or of allocating the data set. Work file data sets allocated by the compiler are deleted when the compilation ends.

Next, the system prompts for information about work file 2.

```
0 0001      A 16 E2
Enter the following information for    Work file 2.
Data set name: SYSUT002
Device address: 4400

                          Press ENTER                          12-01
```

The appropriate diskette should be mounted before the response to the prompting messages are completed.

The final output from a program compilation is an object program data set. The system prompts for information about this data set next.

```
0 0001      A 16 E2

Enter the following information for     Object file.
Data set name:     SYSOUT
Device address:    4400
                          Press ENTER                          12-01
```

Again, the information displayed can be changed. The name of this data set will be the name that must be used to load the object program for future uses. If the default and device address is used and the device address cannot be resolved using the resource allocation table, the object program is written to the diskette containing the compiler.

During the compilation process, each source program statement is read. All statements except comments are analyzed and checked for proper syntax and keyword compatibilities. Names that are used are checked for proper reference and definition. Any errors detected are associated with compilation error codes and descriptive text, which are incorporated into a source program listing to aid in correcting the source program data set.

After the response to the prompt for information about the object data set, the following is displayed.

```
0 0001      D 01 F1
Select listing option
  1. List to printer       4. List to printer (halt if error)
  2. List to diskette
  3. No list               1-3 Halt only if severe errors.
Select option:   1  Press ENTER                                    12-03
```

If option 1, option 2, or option 4 is selected, the system prompts for information about the list file.

If an error is detected and option 1, 2, or 3 is selected, the compiler does not halt. It halts, however, if the error is severe.

If an error is detected and option 4 is selected, the compiler pauses and you choose whether to continue the compile or terminate the compile.

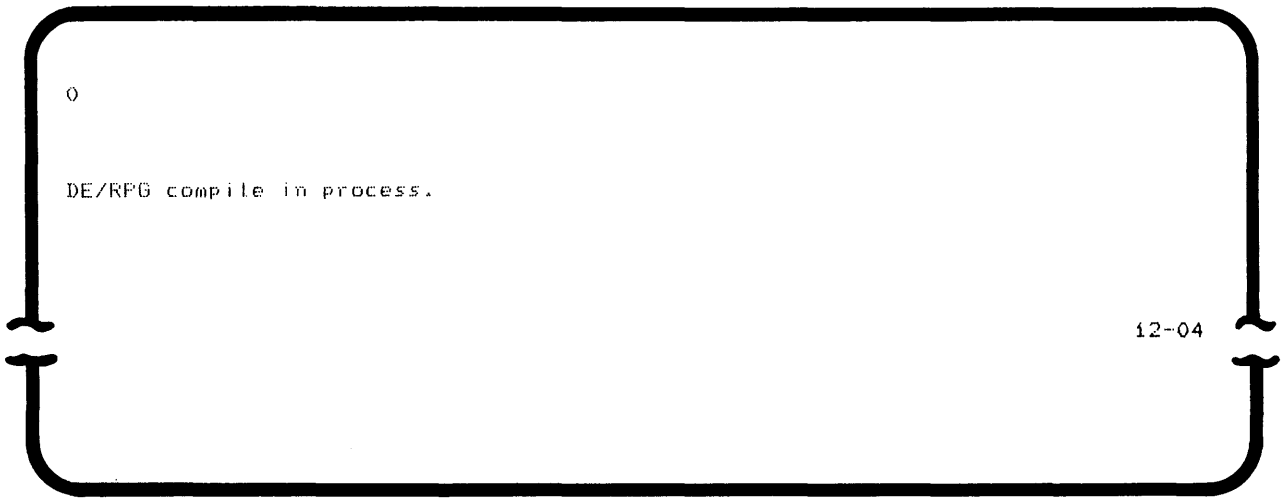
```
0 0001      A 16 E2

Enter the following information for     Listing file.
Data set name:     SYSPRNT
Device address:    8000
Form length:       066     Press ENTER                             12-01
```

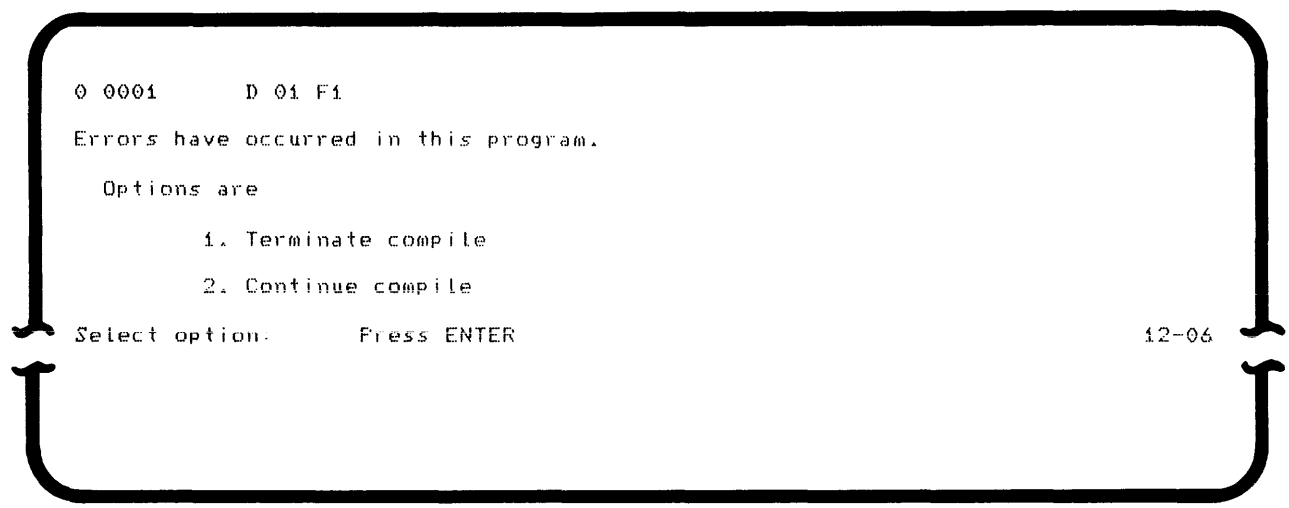
If the default device address is used and *P1* cannot be resolved using the resource allocation table, the four-digit address code 8000 is used.

If the listing is to a diskette, the data set must have previously been allocated with a record length of 80.

At the completion of these set up prompts, the actual compilation begins and the following display appears.



If listing option 1, 2, or 3 was selected, the source program is listed. The listing includes the error codes and text for any detected errors. If a severe error was detected, the compile terminates after the listing is completed. If no severe errors were detected and option 1, 2, or 3 was selected, the compilation continues. If any errors were detected and option 4 was selected on prompt 12-03, the operator is given an opportunity to either continue or end the compilation.



If an error occurs on the object data set at the end of compilation, the operator may press the Reset key and then either select the Retry option or allocate a new object data set.

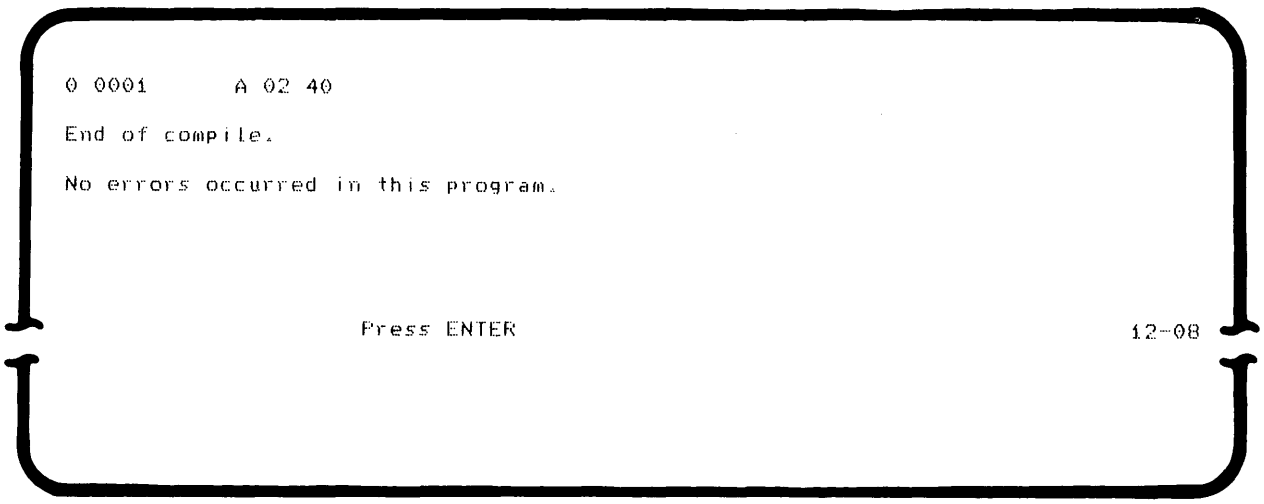
SOURCE PROGRAM ERRORS

Source program errors are identified as one of the following:

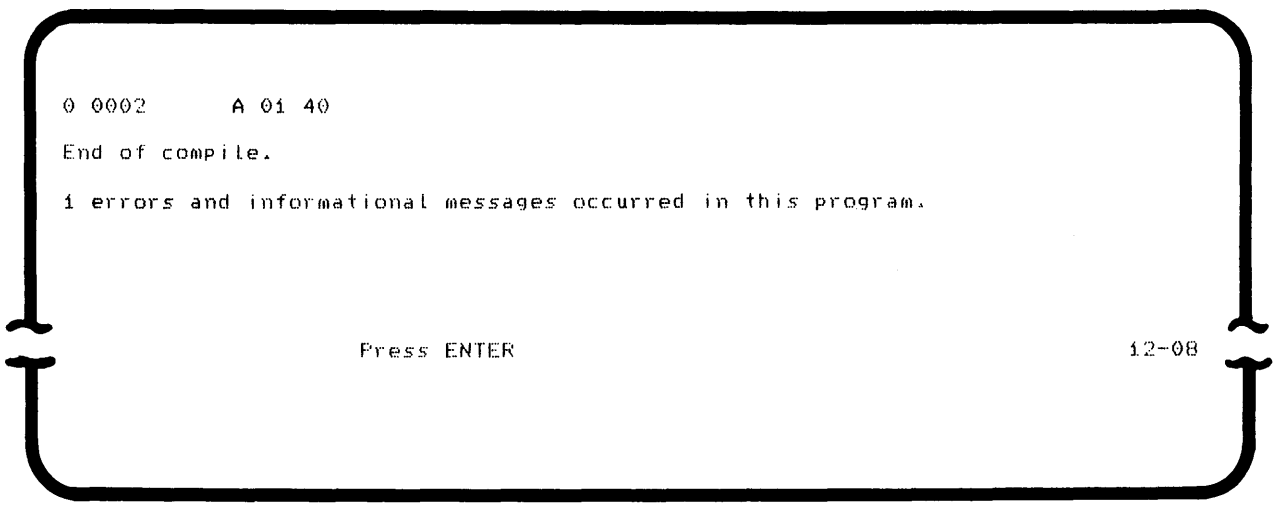
- Severe
- Error
- Informational

Severe errors are those that prevent the compilation of the source program. Errors are those that result from less major syntax violations and keyword incompatibilities. An informational message indicates a condition that could cause a problem. The message is not the result of an error condition.

Upon the successful completion of the compilation, the following is displayed.



If an error did occur in the program, the following is displayed.



When the Enter key is pressed, the load prompt is displayed, and another program can be loaded.

LISTING FORMATS

When the source program is listed to diskette, the 80-character record format of the source program is retained so that the list data set is usable as a source program data set. Error codes and the accompanying text and diagnostic information are written in separate records, which are identified so that they are ignored by the compiler if the data set is used as a source program data set in subsequent compilations. Internally generated sequence numbers replace the contents of the sequence field in the list data set records.

When the listing is printed, the source program, the error codes and text, and the diagnostic information is printed in lines containing 86 positions. The format and content of the printed listing is the same as that of the list data set except that the internally generated sequence number precedes the sequence number used in the source program data set. See Figure 9 for an example of a listing obtained during the compilation.

A complete list of the error messages that can be shown on the compiler listing is contained in *Appendix F. DE/RPG Compiler Error Messages*.

```
00001      Z*********************************************************************
00002      Z* PROGRAM 53 (WITH ERROR).   DE/RPG REFERENCE MANUAL, FIGURE 9 *
00003      Z*********************************************************************
00004      ZJ  EXPLIND
00005      Z A1BEGIN     1E                                    EOJ
00006      A           F DISP            7           DEVICE(CRT) DSPSIZ(6 80)
00007      A           R LOOK
00008      A             NUMBER          6   I       PMT(ENTER THE NUMBER ID)
00009      A           F ITMASTF        27           DEVICE(DISK D1)
00010      A           R ITEM
00011      A           K CUSTN           6          1
00012      A             ITEM#           6
00013      A             QUANT           4
00014      A             PERPRI          5
00015      A             COST
00016      A           F OUTGO          15           DEVICE(DISK D1)
00017      A           R LAST
00018      A             INTER          15  2
00019      C             BEGIN     BEGSR
00020      C             AGAIN     TAG
00021      C                       EXFMTLOOK
00022      C             NUMBER    CHAINITEM                 02
00023      C   02                  GOTO AGAIN
00024      C                       Z-ADDCOST       INTER  152     03
00025      C             BRANCH    TAG
00026      C                       READ ITEM                      05
00027      C     N05N03COST        ADD  INTER      INTER
00028      C  N05                  GOTO BRANCH
00029      C                       WRITELAST
00030      C             END       ENDSR

      * ADDR  CONSTANT
      * 02F0  'ENTER THE NUMBER ID'
      *
      * ADDR  NAME
      * 0303  NUMBER
      * 0309  CUSTN
      * 030F  ITEM#
      * 0315  QUANT
      * 0319  PERPRI
      * 031E  INTER
      *
      *LINE   ERROR
      *00015  0959
      *00024  0963
      *00027  0961
      *
      *003 errors appeared in this program
      *
      *ERROR MESSAGE TEXT
  DE S*0959 A spec name is undefined or an invalid type; name ignored.
  DE E*0961 Factor 1 contains an undefined name or invalid type; name ignored.
  DE E*0963 Factor 2 contains an undefined name or invalid type; name ignored.
      *Severe errors have occurred in this program.
```

Figure 9. Compiler Listing

Chapter 9. Contents

Chapter 9. Operating Characteristics

The characteristics of each DE/RPG program depend upon the source statements used to describe the job. However, certain common characteristics exist during the start-up and execution of these programs. Refer to the *Operator's Guide* for a complete description of the displays and procedures used for program execution.

PROGRAM START-UP

After a DE/RPG program is loaded into a partition, the system attempts to open all files except communications files unless JOBOPT(*NOOPEN) is specified on the job statement. The system displays messages prompting for the name and the device ID for each file. (At this time, the device associated with the file being opened should be readied by the operator.) The name and device ID included in the source statements are displayed, and the operator has the opportunity to change the names and assignments or to accept them as displayed. The device ID displayed is the four-digit hexadecimal ID, which is a translation of the logical ID through the resource allocation table as established during system configuration. (These prompts are bypassed if the job specification statement included the keyword/parameter JOBOPT(*NOPMT).)

If the attempt to open a file is unsuccessful, the system displays an error indication, which can be reset with the Reset key. The system then displays a menu to allow the operator to request a retry of the open or, if appropriate, to request a data set allocation.

If the job includes the use of the transaction file, the operator is prompted to select an initial mode of operation for the program before the transaction file is opened. The modes that can be selected are:

Mode	Purpose
Enter (add)	To add records to an existing data set (extends EOD)
Enter (new)	For creating a new data set (replaces any existing data and existing data is no longer accessible)
Update	For changing selected records in the data set
Verify	For verifying the accuracy of the entry process for records in the data set
Rerun	For automatic reprocessing of an existing data set

The data set involved in these modes of operation is the data set assigned as the transaction file.

If the job does not include the use of the transaction file, the prompt for initial mode selection is bypassed, and the entire program is executed in execute mode.

Refer to the *IBM 5280 Distributed Data System Operator's Guide*, GA21-9364 for operator instructions for executing a DE/RPG program.

PROGRAM EXECUTION

The execution of a program is the sequential processing of the instructions that result from the program compilation. DE/RPG compiled programs that require an operator are controlled by both the source statements and the operator's use of the keyboard.

Although the operator selects an initial mode of operation for a data-entry job at program start-up, changes from one mode to another can be caused by the use of key-initiated functions during program execution.

There are 15 modes of operation available for data entry jobs. The currently active mode of operation is indicated in positions 35 through 37 of the status line. The modes of operation and the key-initiated functions that are available in each mode are described in this section.

Modes of Operation

The modes of operation are listed in the following table in the order in which they are described. The table shows the mode identification displayed in positions 35 through 37 on the status line for each mode of operation and also names the keywords that must be used in the job specification statement to make the modes of operation available during execution.

Modes	Identifiers	Keywords
Enter	E	TFILE
Copy	C	TFILE and CFILE
Copy search	C-S	TFILE and CFILE
Copy transfer	C-T	TFILE and CFILE
Print	P	TFILE and PRTFILE
Rerun display	R-D	TFILE
Update	U	TFILE
Update insert	U-I	TFILE
Update search	U-S	TFILE
Verify	V	TFILE
Verify correct	V-C	TFILE
Verify display	V-D	TFILE
Verify insert	V-I	TFILE
Verify search	V-S	TFILE
Execute	X	

Notice in the preceding chart that execute mode is the only mode of operation for programs that do not use the transaction file.

Enter Mode

The purpose of enter mode is to create or to extend the diskette data set assigned to the transaction file. During enter mode, data is entered under the control of the record description statements and field description statements associated with the file to which the keyboard/display is assigned. As each record is completed, it is transferred to the transaction file.

The order of use of record formats depends both upon the coded entry format statements in the program source and upon selection of formats by the operator. The first entry format statement in the source is always the first format available for use. Thereafter, format use is a function of programmed sequence or the operator's use of the *next format* or *select format* key-initiated functions.

When a format is available for data entry, the organization of the display, the conditioning of the keyboard, and data manipulations and edits are controlled by the coded entries in field description statements that are associated with the record named by the controlling entry format statement. A cursor is positioned beneath the display location for the next data character from the keyboard.

As each input field is entered (or processed), the data is temporarily stored until the record is completed. The stored data is positioned such that when the record is completed, the stored record is in the format specified for the record when written to diskette. Any reformatting from the default record is accomplished as the record is assembled.

As each record is completed, the record is transferred to the transaction file, which writes the data to the data set assigned to the transaction file. Each record is retained in storage in the format written to diskette until the next record is completed. This record retention allows duplication of data from corresponding positions of the previous record into a current record, and also accomodates the next format key-initiated function with conditions specified in the entry format statement.

Copy Mode

Copy mode allows the operator to display records in an existing diskette data set assigned as the copy file. The operator can select records to copy to the data set assigned as the transaction file. The data set from which the records are copied is unchanged, but data in a displayed record can be altered before the record is added to the data set assigned as the transaction file.

Copy mode is entered only from enter mode. Records are initially displayed in format 0. (Records displayed in format 0 are in the image of the record on diskette, and each position in the record is considered a field.) The operator can use the format select function to change the format with which the records are displayed. When the records are displayed using an entry format, the keyboard conditioning and the edits described for the fields apply.

Upon entry to copy mode, the system displays a prompting message instructing the operator to enter the name of the data set from which the records are to be copied. The message also asks for the address of the diskette drive. After the information is keyed, the record advance function signifies to the system that the response to the prompting message is complete.

Copy Search Mode

Copy search mode allows the operator to locate and retrieve a record in a data set assigned as the copy file by position in the data set or by record content.

A change to copy search mode results from the initiation of any of four key-initiated search functions while the program is in copy mode. The key-initiated search functions are search end-of-data, search relative record number, search content, and search sequential content (binary search).

Copy search mode is identified in positions 35 through 37 of the status line by the characters C-S. Upon the completion or cancellation of the specified search, program operation returns to copy mode.

For each type of search except a search end-of-data, the system displays a prompting message instructing the operator to supply search parameters. The message depends upon the type of search requested.

For a search relative record number, the operator is asked to enter the relative record number of the desired record.

For a search content, the operator is allowed to specify up to three character strings and the starting position of each string. The character strings are combined and used collectively as a mask for the search, which permits searching for a record on the contents of three fields. Each character string must be enclosed in apostrophes.

For a search sequential content, the system prompts the operator to enter a character string and a starting position for the string. The character string serves as a mask for the search. This type of search is valid only for ordered data sets – that is, the contents of the searched positions must be in ascending sequence from record to record. The character string must be entered enclosed in apostrophes.

Copy Transfer Mode

During copy transfer mode, records retrieved from the data set assigned to the copy file are copied to the transaction file, which adds the records to the data set assigned to the transaction file. Each record copied extends the end of data.

Print Mode

Unformatted printing of a current record occurs in print mode. The record is printed as a character string in the image of the record on diskette.

From the enter or record insert modes, only a single record can be printed each time the print function is initiated. After the record is printed, the mode of operation returns to the original mode.

When the print function is initiated in other modes of operation, either a single record or a sequence of records can be printed. If the automatic record advance function is active, consecutive records can be printed. The print function is discontinued when the last record in the data set is printed or when the function is canceled with the cancel function or when auto record advance is turned off.

Rerun Display Mode

In rerun display mode, the records in the data set assigned as the transaction file are automatically processed sequentially from the beginning of the data set through the last record. After the last record is processed, the mode changes to enter mode.

If multiple record types are contained in the data set, the format applied to each record must be determined by tests specified in review format statements in the program source.

During rerun display mode, the processing of records is similar to the processing in update mode when the record advance function is initiated in the first manual position of each record. The functions represented by keywords are executed. (Functions that control the keyboard and the display are ignored.) Records in which any data is changed are rewritten in the data set in the location from which they were read.

The data and prompts are displayed. No operator intervention is required unless an error is detected. Then, an error indication is posted and operator intervention is required to resume rerun display mode.

Update Mode

Update mode allows data to be changed or updated in the data set assigned as the transaction file. During update mode, the records read from the data set are displayed, and the operator has the opportunity to replace the existing data by entering the new data into the manual fields. Updated records are then rewritten in the data set assigned as the transaction file. The rewritten data occupies the same physical location in the data set as the data replaced.

The data entered to update a record can be subjected to the same format controls and edits that were used when the record was initially entered. The appropriate entry format can be automatically used if adequate review format statements are included in the program source. The operator also has the option of using key-initiated functions to select a format.

Update Insert Mode

Update insert mode allows the operator to enter new records between two existing records in the data set assigned as the transaction file. The initiation of the record insert function, while the program is in update mode, changes the mode of operation to update insert mode with format 0 active.

The entry of data records is the same as during enter mode. However, if the auto dup/skip function was active when update insert mode was initiated and more than one record is to be inserted, the auto dup/skip function is disabled until the record insert function is completed.

After the new records have been inserted, or when the mode is discontinued, the mode of operation returns to update mode.

The quantity of records to be inserted is determined by the operator's response to a prompting message. The records are inserted in a location starting with the relative record number displayed in the status line when the record insert function was initiated. When the last insert record is entered, the screen is cleared and the mode indicator blinks until the insert function has been completed.

For example, assume that the following two conditions are true:

1. Relative record number 50 is indicated in positions 21 through 26 on the status line when the record insert function is initiated.

2. The operator responds to the prompting message and specifies three records are to be inserted.

After the operator enters the first new record in insert mode, the original record in relative record number 50 and the records following it are rewritten starting at the location for relative record number 53, and the new record is written as relative record number 50. Space is available in the data set for the second and third new records after the first new record is entered.

Update Search Mode

Update search mode allows the operator to locate and retrieve a specific record from a data set assigned as the transaction file. Update search mode is entered from update mode by any one of four key-initiated search functions, which are explained in the topic *Copy Search Mode* in this chapter. Update search is indicated on the status line by U-S in positions 35 through 37. Upon completion of the search, the program returns to update mode.

Verify Mode

Verify mode allows the operator to reenter data for manual fields to verify the accuracy of the initial entry process. As the operator keys the data from the source document, the program compares each character to the corresponding character in the original record, which is read from the data set assigned as the transaction file.

During verification, the same format controls and edits that were used when the program was initially entered can be used. The correct entry format is automatically applied to the record if adequate review format statements are included in the source program. The operator also has the option of using key-initiated functions to select a format for use.

In the verification of a record, the cursor is initially displayed at the first position of the first manual field. All data positions from the cursor through the end of the record are blank on the display. As each data character is keyed, it is compared to the corresponding character from the record in the data set. If the characters match, the position is verified. The verified character is displayed and the cursor is moved to the next position.

If a mismatch is detected, an error is indicated, and the data keys are ignored until the error is reset. While the error is indicated, the entire field in which the error occurred is displayed. When the error is reset (via the Reset key), the operator can again enter a character in the error position.

The second entry for a character position is accepted as correct in either of two conditions:

- The character matches the character from the data set.
- The character matches the character in the previous verify attempt.

When the character is accepted, it is displayed on the screen above the cursor. The cursor is moved to the next position and all unverified character positions are again blank.

Verification of fields with insert or substitution data (INSERT or SUBST keywords) is automatic. The contents of the record are compared to the value determined by the program. If the data from the record matches the data supplied by the program, the verified data is displayed and the cursor moves to the next field.

If a mismatch is detected, the error is indicated on the status line and the data from the record is displayed.

The operator must reset the error with the Reset key and then can exercise either of two options – retain the data from the record, or replace the data in the field with the program-supplied data. If the operator initiates the field advance function, the original data in the record is retained. If the operator instead initiates the field correct function, the program-supplied data replaces the data from the record. In either case, the cursor advances to the next field.

Verification of duplicated fields (resulting from the AUXDUP keyword or the auto dups function) is automatic. The contents of the record is compared with the data in storage (AUXDUP keyword) or from the previous record (Auto Dup). If the data from the record matches the data to be duplicated, the verified data is displayed and the cursor moves to the next field.

If a mismatch is detected, the error is indicated on the status line and the data from the record is displayed with the cursor positioned at the character that caused the error. The operator must reset the error with the Reset key and then press the Dup key. This causes the erroneous character to be replaced (from the storage location or the previous record). Automatic verification then continues to the end of the field.

Verification of skip fields is automatic. If a nonblank character is detected in the field, the error is indicated on the status line and the remaining positions of the field are displayed. The operator must reset the error with the Reset key and then press the Skip key. This causes a blank character to replace the nonblank character. Automatic verification then continues to the end of the field.

In fields for which right-adjust is specified (including the signed numeric data type), data entered during verification can match the keystrokes for the original entry. The first character keyed for verification can be the same as the first character keyed for the field in enter mode. The fill characters are automatically verified. However, if one fill character (leading zero or leading blank) is entered during verification, then all fill characters in the field must be entered.

Fields containing negative quantities contain a hexadecimal D zone in the units position. If field-exiting-required is specified for the field, the zone and digit portions of the character are verified separately. First the digit is keyed to verify the digit portion, and then the field-exit minus function, which verifies the hexadecimal D zone, can be initiated. If the digit keyed causes a mismatch, both the digit and the sign from the original record are displayed and the error is indicated on the status line. The operator must reset the error and can then key again. If the same digit is keyed the second time, that digit replaces the original digit in the display. The sign no longer is displayed although the hexadecimal D zone is retained. The zone must then be verified via the field-exit minus function.

Zone and digit portions of a character can also be verified with the hexadecimal key function in which the zone and the digit are keyed in sequence.

Whenever a record is changed by verify operations the changed record is updated in the data set assigned as the transaction file.

Verify Correct Mode

Verify correct mode allows the operator to change the contents of the current record without the normal comparisons between the data in the record and the data keyed as in verify mode. The data entry processes in verify correct mode resemble the processes in update mode. Upon the completion of the record format, the program returns to verify mode, and the corrected record is available for verification.

Verify Display Mode

Verify display mode allows the operator to view an entire record on the display. No verification or correction can be accomplished in this mode.

Verify display mode can be entered only from verify mode via the record display function.

Verify Insert Mode

Verify insert mode allows the operator to enter new records between two existing records in the data set assigned as the transaction file. This mode is identical to update insert mode except that verify insert mode is entered from and returns to verify mode. See *Update Insert Mode* in this chapter for a description of the insert mode of operation.

Verify Search Mode

Verify search mode allows the operator to locate and retrieve a specific record from a data set assigned as the transaction file. Verify search mode is entered from verify mode by any one of four key-initiated search functions, which are explained in the topic *Copy Search Mode* in this chapter. Verify search mode is identified in the status line by V-S in positions 35 through 37. Upon the completion of the search, the program returns to verify mode.

Execute Mode

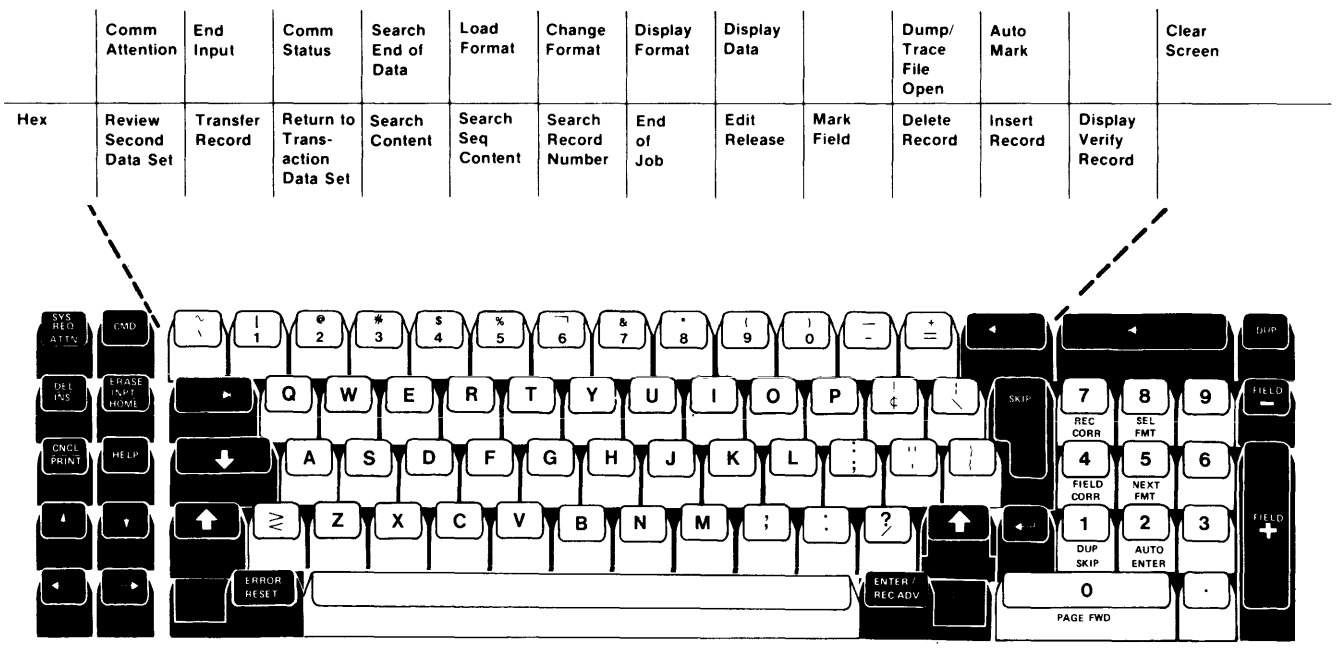
During the execution of subroutines, which are built with calculation statements in the source program, the program operates in execute mode. If the subroutine performs an extended edit or an explicit I/O operation and then returns to a data entry job that includes the use of the transaction file, the change to execute mode is temporary, and the mode of operation returns to the previous mode (enter, update, or verify). When the transaction file is not used in the job, the program remains in execute mode permanently.

When the keyboard/display is available for use as the result of an EXFMT operation in calculation subroutines, the program is always in execute mode. Data is assembled under the control of the record description statements and the field description statements associated with the file to which the keyboard/display is assigned. As each record is completed, the program returns to the subroutine with the operation that follows the EXFMT operation.

Certain functions that can be specified in the edit field of file description statements for normal data entry jobs cannot be used with execute mode. These functions are automatic duplication, automatic skip, auxiliary store, and auxiliary duplication.

Key-Initiated Functions

In the following illustration of a typewriter keyboard, the function keys are dark and the data keys are light. Additional functions can be initiated by pressing the Cmd (Command) key as a prefix to 14 of the keys at the top of the keyboard or the seven function keys on the numeric pad on the right (0, 1, 2, 4, 5, 7, and 8).



For some functions, several conditions that are indicated on the display must be understood. These conditions are *awaiting-field-exit* and *awaiting-record-advance*.

In fields that are described to require field exiting (either CHECK(FE) or signed numeric data type), an awaiting-field-exit condition is indicated after the last character position for the field is keyed. To indicate awaiting-field-exit, the cursor remains in the last data position in the field and blinks, and positions 15 and 16 in the status line (positions remaining in the field) contain 01.

An awaiting-record-advance condition exists after the last manual position in a record is keyed if the automatic enter function is disabled. Awaiting-record-advance is indicated on the display by the blinking cursor beneath the last position of the record. The status line contains 00 in positions 15 and 16 (positions remaining in the field).

The following table lists the key-initiated functions and the modes of operation. The Xs indicate that the function can be used in the mode.

Function	Modes														
	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
Attention	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Auto Dup/Skip	X	X			X		X			X					X
Auto Enter	X	X			X		X			X	X				X
Auto Mark	X	X			X		X			X	X				X
Cancel			X		X			X	X				X	X	
Character Advance	X	X					X	X		X	X		X		X
Character Backspace	X	X					X	X		X	X		X		X
Character Delete	X	X					X	X			X		X		X
Character Insert	X	X					X	X			X		X		X
Clear Screen	X	X					X	X							
Cursor Down	X	X					X	X			X		X		
Cursor Left	X	X					X	X		X	X		X		X
Cursor Up	X	X					X	X		X	X		X		
Cursor Right	X	X					X	X	X			X	X		X
Duplicate	X	X					X	X		X	X		X		X
Edit Release	X	X				X	X	X		X	X		X		X
End of Job	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Erase Input	X	X	X		X		X	X	X	X	X		X	X	
Field Advance	X	X					X	X		X	X		X		X
Field Backspace	X	X					X	X		X	X		X		X
Field Correct										X					
Field Exit	X	X					X	X		X	X		X		X
Field Exit Minus	X	X					X	X		X	X		X	X	
Help	X	X					X	X			X		X		X
Hexadecimal	X	X					X	X		X	X		X		X
Home (Record Backspace)	X	X	X				X	X	X	X	X		X	X	X
Mark Field	X	X				X	X	X		X	X		X		X
New Line	X	X					X	X			X				
Next Format	X	X					X	X		X	X		X		
Page Forward		X					X			X					
Print	X	X					X			X					
Record Advance	X	X	X		X		X	X	X	X	X		X	X	X
Record Backspace (Home)	X	X	X				X	X	X	X	X		X	X	X
Record Correct										X					
Record Delete	X						X			X					
Record Display										X		X			
Record Insert							X			X					
Record Transfer		X	X												
Reset	X	X				X	X	X		X	X		X		
Review File	X														
Search Content		X					X			X					
Search End-of-Data		X					X			X					
Search Relative Record		X					X			X					
Search Sequential Content		X					X			X					
Select Format	X	X					X	X		X	X		X		
Skip	X	X					X	X		X	X		X		X
System Request	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X

A modified form of the preceding chart is included with the description of each key-initiated function. The symbols identify the portion of the description that applies to each mode of operation. These descriptions are relevant to data operations rather than responses to system prompts. While system prompts are displayed, the functions available during execute mode can be used.

Attention

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

(A) The attention function initiates a procedure that allows a program operating in a background partition to attach to the keyboard/display if that program requires operator intervention. If the background program requires operator action, the entire display is replaced by that program. Upon completion of any required response by the operator, the original program resumes control of the keyboard/display, and execution of that program continues.

Auto Dup/Skip

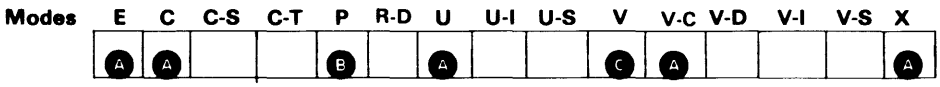
Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	A	A			B		A			C					A

(A) Pressing the Dup/Skip key enables the automatic duplication function, which is indicated by a reverse image D in position 28 of the status line. When the automatic duplication function is enabled, programmed auto skip, auto duplication, auxiliary duplication, and auxiliary store operations are performed. While the automatic duplication function is enabled, pressing the Dup/Skip key disables the function.

(B) The automatic duplicate function can be enabled or disabled while the program is in print mode. This function has no direct effect on the operations in print mode.

(C) Pressing the Dup/Skip key enables the automatic duplication function, which is indicated by a reverse image D in position 28 of the status line. When the automatic duplication function is enabled, programmed auto skip, auto duplication, and auxiliary duplications are automatically verified and auxiliary store operations are performed. While the automatic duplication function is enabled, pressing the Dup/Skip key disables the function.

Auto Enter



(A) Pressing the Auto Enter key enables the automatic enter function, which is indicated by a reverse image R in the status line. While the automatic enter function is enabled, the completion of the last manual field in each record automatically initiates the record advance function.

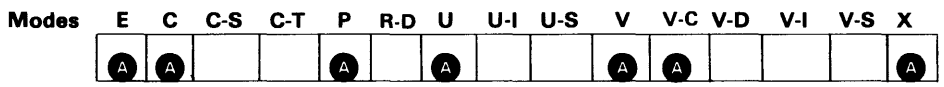
When the automatic enter function is enabled, pressing the Auto Enter key disables the automatic enter function, which is indicated by a blank in position 30 of the status line.

(B) The Auto Enter key is used to discontinue the printing of consecutive records.

(C) Pressing the Auto Enter key enables the automatic enter function, which is indicated by a reverse image R in position 30 in the status line. While the automatic enter function is enabled, the verification of the last manual field in each record automatically initiates the record advance function.

When the automatic enter function is enabled, pressing the Auto Enter key disables the automatic enter function, which is indicated by a blank in position 30 in the status line.

Auto Mark



(A) Pressing the Auto Mark key enables the automatic mark function. When the automatic mark function is enabled, pressing the Auto Mark key disables the function.

While the automatic mark function is enabled, the occurrence of a field edit error causes a mark (X'FF') to be placed in the first position of the field and automatically initiates the field advance function.

Cancel

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
			(A)		(B)			(C)	(A)				(C)	(A)	

(A) When a message prompting for search parameters is displayed, the cancel function discontinues the search mode and returns the program to update mode, verify mode, or copy mode.

(B) The Cancel key can be used to discontinue printing consecutive records or to cancel the print function if the open operation on a printer file failed.

(C) When the message prompting for the number of records to insert is displayed or when the Cancel key is pressed before the Enter/Rec Adv key, the cancel function discontinues the insert mode and returns the program to update mode or verify mode. The data set assigned as the transaction file is unchanged – no space is made for new records.

Character Advance

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	(A)	(A)					(A)	(A)		(B)	(A)		(A)		(C)

(A) The character advance function moves the cursor ahead one position in a field. The data in the position is not changed. When the cursor is moved out of one field into the next, a field advance function is performed. An attempt to advance at the end of a record (advance from the last position of the last manual field) causes an error unless the automatic record advance function is enabled, in which case the record advance function is performed.

When awaiting-field-exit is indicated, the character advance function serves as the field exit function.

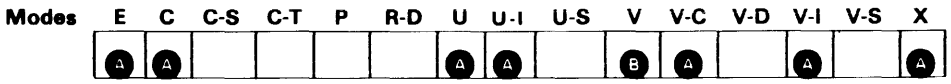
(B) The character advance function can be used when awaiting-field-exit is indicated to advance to the next field. When awaiting-record-advance is indicated, the character advance function causes an error.

In fields described as right-to-left with the keyword/parameter CHECK(RL), the character advance function blanks any character passed, and the blanked character must be reverified.

(C) The character advance function moves the cursor ahead one position in a field. The data in the position is not changed. When the cursor is moved out of one field into the next, a field advance function is performed. An attempt to advance at the end of a record (advance from the last position of the last manual field) causes an error unless the automatic record advance function is enabled, in which case the program control returns to the operation following the EXFMT operation.

When field-exit-pending is indicated, the character advance function serves as the awaiting-field-exit function.

Character Backspace



(A) The character backspace function moves the cursor back one position. The data is not changed. The function can be used to move the cursor from the first position of one manual field to the last position of the preceding manual field but cannot be used to backspace into a previous record. In the first manual position in a record format, the character backspace function causes reexecution of any automatic functions for fields preceding the first manual field.

When awaiting-field-exit is indicated, the character backspace function resets the condition. The cursor remains positioned in the last position of the field and the position can be rekeyed.

When awaiting-record-advance is indicated, the character backspace function resets the awaiting-record-advance condition and positions the cursor in the last manual position of the record format.

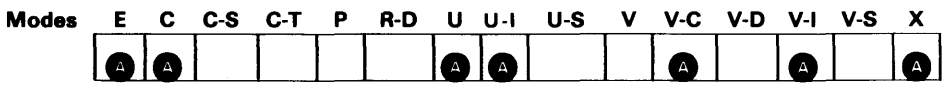
(B) The character backspace function moves the cursor back one position. The data in the position is blanked on the display. The function can be used to move the cursor from the first position of one manual field to the last position of the preceding manual field but cannot be used to backspace into a previous record. In the first manual position in a record format, the character backspace function causes reverification of any automatic fields preceding the first manual field.

Except for right-to-left fields, when field-exit-pending is indicated, the character backspace function resets the condition. The cursor remains in the last position of the field and the position must be reverified.

Except for right-to-left fields, when record-advance-pending is indicated, the character backspace function resets the record-advance-pending condition and positions the cursor in the last manual position of the record format.

The character backspace function is invalid in fields described with CHECK(RL) except when either awaiting-field-exit or awaiting-record-advance is indicated. Then the entire field is blanked on the display and must be reverified.

Character Delete

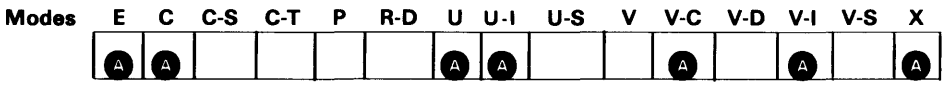


A The character delete function deletes the character at the cursor position. The characters within the field and to the right of the cursor shift to the left one position, and a blank is inserted in the rightmost position of the field. The cursor position does not change.

In a character check type field (C-specified in the data type field in the field description statement), the character delete function acts within subfields only. Subfields are adjacent character positions for which the same character type is specified in the parameter for the keyword SHIFT.

The character delete function cannot be used in a blank check field or in a mandatory fill field.

Character Insert



A The character insert function, which is indicated by > in position 14 in the status line, inserts a character at the current position of the cursor. Each data key operation first shifts the data in the cursor position and all data between the cursor and the right end of the field to the right one position, and then inserts the new character into the cursor position. The cursor moves one position to the right, remaining under the character it was under when the operation started. During the character insert function, the cursor can be moved within the field by using the character advance and character backspace keys.

The character insert function must be discontinued by the reset function before the field can be left.

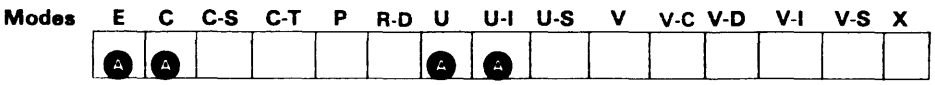
When a nonblank data character occupies the rightmost position of the field, an attempt to insert a character causes an error.

In a field specified for character check type data (C in the data type field in the field description statement), the character insert function shifts data only within a range of positions for which the same character type is specified in the parameter for the SHIFT keyword. Data is not shifted into a position for which a different data type is specified.

Character insert can be used in a field specified for hexadecimal type (H in the data type field), but if the function is initiated after only the first half of a character position is entered, that half of a character is lost and must be reentered.

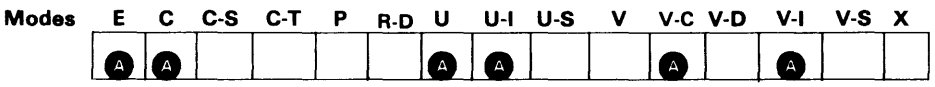
Character insert cannot be used in fields for which mandatory fill is specified.

Clear Screen



(A) The clear screen function discards any data displayed for the current record format and positions the cursor in the first position of format 0 for data entry.

Cursor Down



(A) The cursor down function is valid only while format 0 is being used. This function moves the cursor *down* one line on the display. (This function does *not* move the cursor from the bottom line to the top line.) The horizontal position of the cursor is unchanged. Cursor movement is limited to the number of positions specified for the length of a record in the transaction file.

Cursor Left

The cursor left function is the same as the character backspace function.

Cursor Right

The cursor right function is the same as the character advance function.

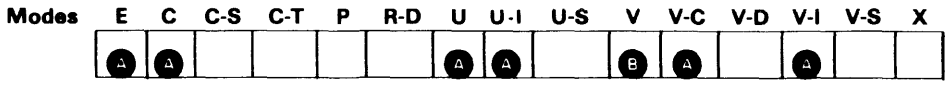
Cursor Up

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	(A)	(A)					(A)	(A)		(B)	(A)		(A)		

(A) The cursor up function is valid only while format 0 is being used. This function moves the cursor *up* one line on the display. (This function does *not* move the cursor from the top line to the bottom line.) The horizontal position of the cursor is unchanged. Cursor movement is limited to the number of positions specified for the length of a record in the transaction file.

(B) The cursor up function is valid only while format 0 is being used. This function moves the cursor *up* one line on the display. (This function does *not* move the cursor from the top line to the bottom line.) The horizontal position of the cursor is unchanged. Any data positions passed are blanked on the display and must be reverified. Cursor movement is limited to the number of positions specified for the length of a record in the transaction file.

Duplicate



A The duplicate function copies data from a source into the remaining positions of the current field. The source of the copied data depends upon the use of the keyword AUXDUP in the corresponding field description statement in the program source.

If AUXDUP is not specified, the source of the data copied is the corresponding positions of the previous record as written to diskette.

If AUXDUP is specified for the field, the source of the data is the corresponding positions of the field named as the parameter for the AUXDUP keyword. To determine the corresponding positions, consider the source and the current field left-end aligned.

If SUBST is specified for the field, the DUP key is disabled. If the operator presses the DUP key, an error message is displayed.

B The duplicate function compares data from a source with the remaining positions of the current field. The source of the data depends upon the use of the keyword AUXDUP in the corresponding field description statement in the program source.

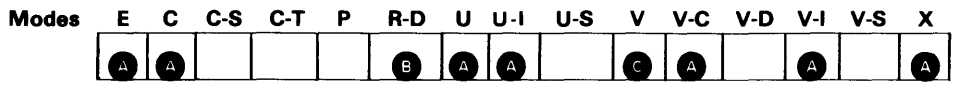
If AUXDUP is not specified, the source of the data is the corresponding positions of the previous record as written to diskette.

If AUXDUP is specified for the field, the source of the data is the corresponding positions of the field named as the parameter for the AUXDUP keyword. To determine the corresponding positions, consider the source and the current field left-end aligned.

When an error is detected, the remainder of the field is displayed with the cursor positioned at the position in error. If the duplicate function is again initiated after the error is reset, the character in error is replaced by the character in the corresponding position of the source, and automatic verification resumes.

If SUBST is specified for the field, the DUP key is disabled. If the operator presses the DUP key, an error message is displayed.

Edit Release



(A) The edit release function can be used in response to an error to suspend programmed keyboard conditioning (data type field) or to bypass the use of a keyword specified in the field description statement for the current field.

When an error is detected while the data for a field is being keyed, the operator must reset the error with the Reset key. Then the operator can initiate the edit release function to allow the entry of any character for the remainder of the field.

When an error is detected after all data for the field is entered, the error is the result of a keyword function such as a range check, self-check, or table operation. The operator can initiate the edit release function to reset the error and to bypass the keyword function that detected the error. The next keyword function is then processed.

(B) The edit release function suspends the keyword function that detected the error. Rerun mode is restored, and processing is resumed with the next edit function for the same field. The edit release function must be preceded by the reset function.

(C) The edit release function can be used in response to an error to suspend programmed keyboard conditioning (data type field) or to eliminate the use of a keyword specified in the field description statement for the current field.

When an error is detected while the data for a field is being key verified, the operator must reset the error with the Reset key. Then the operator can initiate the edit release function to allow the entry of any character for the remainder of the field.

When an error is detected after all data for the field is entered, the error is the result of a keyword function such as a range check, self-check, or table operation. The operator can initiate the edit release function to reset the error indication and to bypass the keyword function that detected the error. The next keyword function is then processed.

End-of-Job

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

(A) The end-of-job function closes all files and ends program execution. The operator is prompted to select end-of-job options.

After selecting the desired option, the operator must press the Record Advance key to execute the selected option.

Note: When the end-of-job key is used, the EOJ keyword(s) is bypassed.

Erase Input

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	A	A	B		C		A	A	B	D	A		A	B	

(A) The erase input function blanks all the input fields for the current record format. The cursor is returned to the first position of the first manual field of the same format.

The erase input function can be initiated at any time within the current format use.

(B) The erase input function clears any response to the search prompts.

(C) The erase input function can be used while the program is in print mode to clear a response to a prompting message. The cursor is returned to the first position of the response field ready for the first character of the operator's response.

(D) The erase input function clears any data displayed and changes the mode of operation from verify mode to verify correct mode. The cursor is positioned at the first manual position of the format.

Field Advance

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	(A)	(A)					(A)	(A)		(B)	(A)		(A)		(A)

(A) The field advance function signifies that the data entry for the field is completed and starts the processing of any specified edit functions. Any data from the cursor position to the end of the field is unchanged. If the edit functions are completed without error, the cursor is then positioned in the first position of the next manual field.

When awaiting-field-exit is indicated, the field advance function satisfies the field exiting requirement.

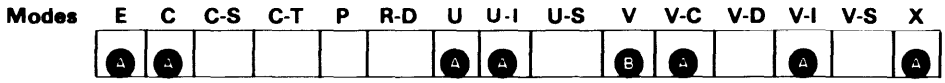
When awaiting-record-advance is indicated, the field advance function causes an error unless auto enter has been enabled.

(B) When awaiting-field-exit is indicated, the field advance function satisfies the field exiting requirement.

When awaiting-record-advance is indicated, the field advance function causes an error.

The field advance function can be used after an insert error is reset to accept the data from the original record as correct.

Field Backspace



(A) When the cursor is positioned in other than the first position of a field, the field backspace function positions the cursor in the first position of the same field. When the cursor is positioned in the first position of a field, the field backspace function positions the cursor in the first position of the previous manual field.

When awaiting-field-exit is indicated, the field backspace function resets the condition and returns the cursor to the first position of the field.

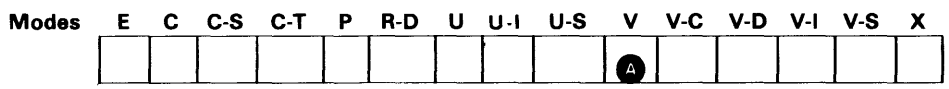
When awaiting-record-advance is indicated, the field backspace function resets the condition and returns the cursor to the first position of the last manual field in the record format.

(B) When the cursor is positioned in other than the first position of a field, the field backspace function positions the cursor in the first position of the same field. The field is blanked on the display and must be reverified. When the cursor is positioned in the first position of a field, the field backspace function positions the cursor in the first position of the previous manual field. Then that field is blanked on the display and must be reverified.

When awaiting-field-exit is indicated, the field backspace function resets the condition and returns the cursor to the first position of the field.

When awaiting-record-advance is indicated, the field backspace function resets the condition and returns the cursor to the first position of the last manual field in the record format.

Field Correct



(A) The field correct function allows the replacement of the contents of a manual field. Comparisons between the data being replaced and the data entered are suspended.

The function can be initiated in any position of the field. The field is blanked on the display and the cursor is positioned in the first position of the field. Data can be entered as in enter mode. Upon completion of data entry, the field is blanked on the display, and the cursor is returned to the first position of the field, which can then be verified.

The field correct function can also be used after an insert verify error is reset. The field correct function replaces the original data from the data set with data processed during the current program execution.

Field Exit

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	(A)	(A)					(A)	(A)		(B)	(A)		(A)		(A)

(A) The result of the field exit function depends upon the characteristics of the field as specified in the field description statement.

In fields described for signed numeric data (the letter S in the data type field in the field description statement), the field exit function signals the completion of data entry. Any data to the right of the cursor is blanked, and the data to the left of the cursor is shifted into the right end of the field (right adjusted) with zero fill. An additional display position immediately to the right of the field is set blank to indicate a positive number. Any edit functions specified are then processed. If no errors are detected, the cursor is positioned for the next manual field.

In fields for which right adjusting is specified (CHECK keyword with RZ or RB as a parameter), the field exit function signals the completion of data entry for the field. Data to the left of the cursor is right adjusted. Positions to the left of the data are filled with the specified fill characters. Then any specified edits are performed. If no errors are detected, the cursor advances to the next manual field.

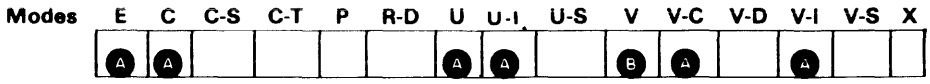
When the field is neither a signed numeric field nor a right-adjust field, the field exit function fills from the cursor position to the end of the field with blanks and then starts the processing of any specified edits. If no errors are detected, the cursor advances to the next manual field.

(B) The result of the field exit function depends upon the characteristics of the field as specified in the field description statement.

In fields described either as a right-adjust field or for signed numeric data (the letter S in the data type field in the field description statement), the field exit function is valid only when awaiting-field-exit is indicated or in the leftmost position of the field when the entire field is fill characters. When awaiting-field-exit is indicated and both the zone portion and the digit portion of the rightmost character have been verified, the field exit function causes an advance to the next field. When field-exit-pending is indicated and the zone portion of the last character position has not been verified, the field exit function verifies a hexadecimal F zone. If the data is negative (hexadecimal D zone), an error is indicated. In signed numeric fields, the minus sign is displayed. In other right-adjust fields the graphic character for the negative number is displayed. If the same function is initiated after the error is reset, the zone is changed and the next field is available for verification.

When the field is neither a signed numeric field nor a right-adjust field, the field exit function verifies blanks from the cursor position to the end of the field. If no errors are detected, the program advances to the next field.

Field Exit Minus



(A) The field exit minus function is valid only for fields for the following data types: signed numeric, digits only, numeric only, and numeric shift. With these data types, the field exit minus function and the field exit function are identical except that the sign of the data in the field indicates a negative quantity.

In signed numeric fields, a minus sign is displayed in the position to the right of the field.

In fields for digits only, numeric only, or numeric shift data types when right-adjust is specified or awaiting-field-exit is indicated, the zone of the rightmost character (which must be a number) is changed to hexadecimal D. The rightmost character in the field is displayed as follows to signify the negative sign.

Numeric Shift	**Digits Only/ Numeric Only**
} for 0	$\bar{0}$ for 0
J for 1	$\bar{1}$ for 1
K for 2	$\bar{2}$ for 2
L for 3	$\bar{3}$ for 3
M for 4	$\bar{4}$ for 4
N for 5	$\bar{5}$ for 5
O for 6	$\bar{6}$ for 6
P for 7	$\bar{7}$ for 7
Q for 8	$\bar{8}$ for 8
R for 9	$\bar{9}$ for 9

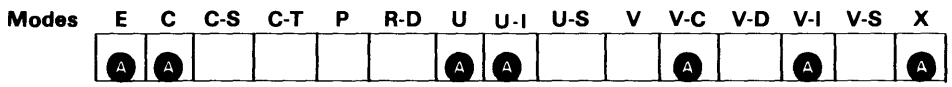
In fields for digits only, numeric only, or numeric shift data types without right-adjust specified and without awaiting-field-exit indicated, all positions from the cursor through the next-to-last position of the field are blanked, and the rightmost position is set to hexadecimal D0, which displays as for numeric shift and 0 for digits only and numeric only data types.

(B) The field exit minus function is valid in fields described for the following data types: signed numeric, digits only, numeric only, and numeric shift. With these data types, the field exit minus function and the field exit function are identical except that the zone of the right-most character is verified for a hexadecimal D to signify a negative quantity.

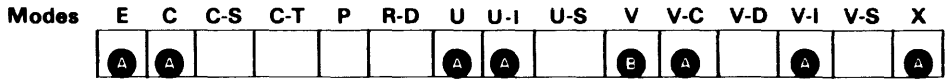
For signed numeric data type fields or fields with right-adjust specified, the field exit minus function is valid only while awaiting-field-exit is indicated or when the cursor is in the first position of the field and the field is all fill characters except for the sign.

In nonright-adjust fields, the field-exit minus function verifies all remaining positions of the field except the last position for blanks. If a non-blank character is encountered, an error is indicated and the unverified positions are displayed with the cursor positioned at the error. If the field exit minus function is again initiated after the error is reset, the nonblank character is replaced by a blank, and verification for blanks continues for the remaining position. The rightmost position of the field is verified for a hexadecimal D0.

Help



(A) The help function can be initiated while a user-defined error (code 98XX) is indicated on the status line. (The 98XX error codes are produced only when the ERROR keyword is specified in a secondary line of a field description statement.) The help function displays the user's help message (the second parameter of the ERROR keyword) in positions 41 through 80 of the status line. (Support for the HELP function must be included during system configuration. See the *IBM 5280 System Control Programming Reference/Operation Manual*.)



A The hexadecimal function allows the operator to enter any hexadecimal values by pressing two data keys in succession for each character position. The numbers 0 through 9 and the letters A through F are the only valid entries.

The character displayed above the cursor is the single character that results from the translation of the hexadecimal value entered. If the translation does not yield a displayable character, the value is displayed as ■. The hexadecimal representation of the character (the value of the two data keys operated) displays in positions 18 and 19 in the status line.

The reset function can be initiated at any time to cancel the hexadecimal function. If the Reset key is pressed before a valid pair of data keys are operated, the hexadecimal function is cancelled and no data is displayed above the cursor.

B The hexadecimal function allows the operator to verify any hexadecimal values by pressing two data keys in succession for each character position. The zone and digit portions of a character are verified separately. The numbers 0 through 9 and the letters A through F are the only valid entries.

The character is displayed at the cursor position after the complete hexadecimal value is entered. If the translation does not yield a displayable character, the value is displayed as a solid rectangular block. The hexadecimal representation of the character (the value of the two data keys operated) displays in positions 18 and 19 in the status line.

The reset function can be initiated at any time to cancel the hexadecimal function. If the Reset key is pressed before a valid pair of data keys are operated, the hexadecimal function is cancelled and no data is displayed above the cursor.

Mark Field

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	(A)	(A)				(B)	(A)	(A)		(C)	(A)		(A)		(A)

(A) The mark field function can be used to flag a field of data as invalid. A hexadecimal FF, which displays as ■, is placed in the leftmost position of the field. All remaining edits specified in the field description statement are ignored, and the cursor advances to the first position of the next manual field when the operator presses the Fld Adv key.

The mark field function can be initiated while the cursor is in any position in the field to be marked.

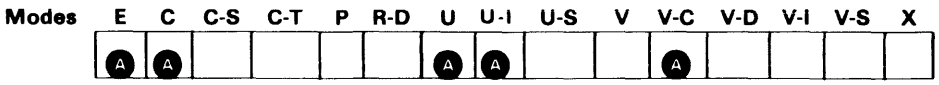
(B) The mark field function places hexadecimal FF in the leftmost position of the field containing the error. Processing of the record resumes with the next field. The mark field function must be preceded by the reset function.

(C) The mark field function can be used to flag a field of data as invalid. A hexadecimal FF, which displays as ■, is placed in the leftmost position of the field. All remaining edits specified in the field description statement are ignored, and the cursor advances to the first position of the next manual field when the operator presses the Fld Adv key.

The mark field function can be initiated while the cursor is in any position in the field to be marked.

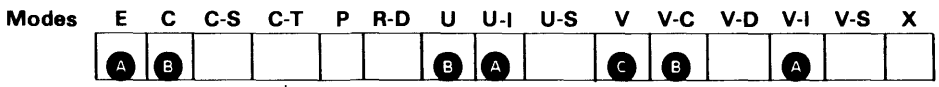
The use of the mark field function in a record suspends the writing of the verify mark, which can be specified by the keyword VMARK in a record description statement in the source program.

New Line



(A) The new line function is valid only while format 0 is being used. The new line function positions the cursor in the first position of the next line on the display.

Next Format

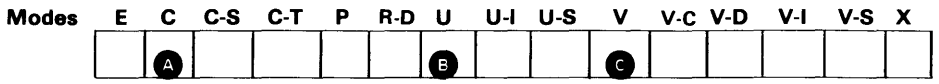


(A) At any time during the use of a format, pressing the Next Format key discontinues the format and advances to the format as specified in the next format ID field in the current entry format statement. Any data assembled for the record at the time the key is pressed is discarded. The cursor is positioned in the first position of the first manual field of the new format.

(B) At any time during the use of a programmed format (a format other than format 0), pressing the Next Format key discontinues the format and advances to the format specified in the *next format ID* field in the current entry format statement. The same record is reread and displayed. The cursor is positioned in the first position of the first manual field of the new record format.

(C) At any time during the verification of a record, pressing the Next Format key discontinues the use of the current format and advances to the next sequential format as specified in the entry format statements. Any data verified at the time the key is pressed is blanked on the display, and the entire record must be reverified using the new format.

Page Forward

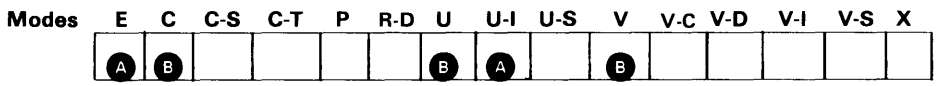


(A) The page forward function bypasses the processing of any remaining fields in the displayed record. The next record in the data set assigned as the copy file is displayed.

(B) The page forward function bypasses the processing of any remaining fields in the displayed record. The record is not updated in the data set. The next record in the data set assigned as the transaction file is displayed for update activity.

(C) The page forward function suspends the verification of any remaining fields in the record. The current record in the data set is not updated. The next sequential record is read and is available for verification.

Print



(A) The print function changes the mode of operation to print mode. The single record containing the record can be printed if the job description statement contains the keyword PRTFILE.

(B) The print function can be used only if the keyword PRTFILE was used in the job specification statement for the current job. Pressing the Print key changes the mode of operation to print mode and the operator is prompted to enter a printer assignment. See *Print Mode*, earlier in this chapter.

Record Advance

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	A	B	C		D		E	F	C	G	H		F	C	I

A The record advance function initiates the processing of any remaining fields in the record format. If the fields are completed without errors, the record is transferred to the transaction file, and the next record format is available for data entry.

B The record advance function initiates the processing of any remaining fields in the record format. The data set assigned to the copy file is not changed, and the record is not added to the data set assigned to the transaction file. The next record is displayed in format 0 with the cursor in the first position.

C The record advance function can be used, while the prompting message is displayed, to signify to the system that all search parameters are entered. The search is in the forward direction for the search content function.

D The record advance function signifies that the operator's response to a prompting message is complete.

E The record advance function initiates the processing of any remaining fields in the record format for the displayed record. If the fields are completed without errors and the record was changed in update mode, the record is transferred to the transaction file, and the next record is read and displayed for update activity.

When the last record in the data set is displayed, the record advance function changes the mode to enter mode.

F The record advance function initiates the processing of any remaining fields in the record format. If the fields are completed without errors, the record is transferred to the transaction file. While the prompting message (prompt for number of records to insert) is displayed, the record advance function signifies the completion of the response.

G The record advance function initiates verification of any remaining fields in the record. Remaining manual fields are verified for blanks. Mismatches between the data from the data set and and the program supplied data are indicated on the status line, and the cursor indicates the position of the mismatch. Verified records are updated in the data set assigned to the transaction file. An advance from the last record in the data set (EOD) initiates the end-of-job function.

H The record advance function initiates the processing of any remaining fields in the record format. If all the fields are completed without detected errors, the mode of operation returns to verify mode. The cursor is positioned at the first position of the first manual field of the corrected record.

(I) The record advance function initiates the processing of any remaining fields in the format. If the fields are completed without errors, program control returns to the routine.

Record Backspace

Modes	E	C	C-S	C-T	P	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S	X
	(A)	(B)	(C)				(D)		(C)	(E)	(F)		(G)	(C)	(H)

(A) While the cursor is in any position other than the first position of the first manual field of the record, the record backspace function returns the cursor to the first position of the first manual field.

While the cursor is at the first position of the first manual field, the record backspace function changes the mode to update mode and displays the last record transferred to the transaction file.

(B) While the cursor is in any position other than the first position of the first manual field of the record, the record backspace function returns the cursor to the first position of the first manual field.

While the cursor is at the first position of the first manual field, the record backspace function displays the physically preceding record in the data set assigned as the copy file.

(C) The record backspace function can be used, while the prompting message is displayed (search content only), to signify to the system that all search parameters are entered. The search is backward, toward the first record in the data set.

(D) While the cursor is in any position other than the first position of the first manual field of the record, the record backspace function returns the cursor to the first position of the first manual field.

While the cursor is at the first position of the first manual field, the record backspace function displays the physically preceding record in the data set assigned as the transaction file.

(E) When the record backspace function is initiated in other than the first position of the first manual field, the cursor is returned to the first position of the first manual field. Any verified fields are blanked from the display and must be reverified.

When the record backspace function is initiated with the cursor positioned in the first manual position of the record, the physically preceding record in the data set is available for verification with the cursor in the first manual position of the record.

(F) If the cursor is positioned at other than the first position of the first manual field, the record backspace function returns the cursor to the first position of the first manual field. The program remains in verify correct mode.

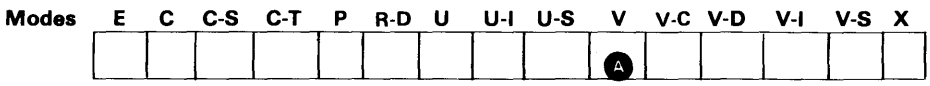
If the cursor is positioned at the first position of the first manual field, the record backspace function changes the mode of operation to verify mode and positions the cursor at the first position of the first manual field of the previous record.

G If the cursor is positioned at other than the first position of the first manual field, the record backspace function returns the cursor to the first position of the first manual field. The program remains in update insert mode.

If the cursor is positioned at the first position of the first manual field, the record backspace function changes the mode of operation to insert mode or verify mode and positions the cursor at the first position of the first manual field of the previous record.

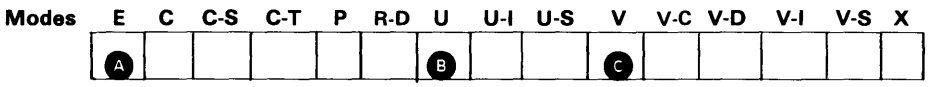
H The record backspace function returns the cursor to the first position of the first manual field of the current format.

Record Correct



A The record correct function positions the cursor in the first manual position of the current format and changes the mode of operation from verify mode to verify correct mode.

Record Delete



A The record delete function discards any data displayed for a record format, transfers a logically deleted record to the transaction file, and positions the cursor to the first position of the first manual field of the next record format.

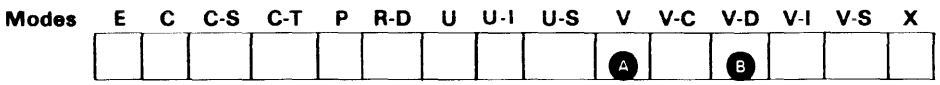
Logically deleted records are written to the diskette data set assigned to the transaction file and are included in relative record numbering. Deleted records cannot be retrieved.

B The record delete function replaces the displayed record with a logically deleted record. The function then reads the next record from the data set and positions the cursor to the first position of the first manual field.

Logically deleted records cannot be retrieved but are included in relative record numbering. During operations that advance or backspace to a new record, deleted records are ignored and the next record is displayed.

C The record delete function deletes the current record in the data set assigned as the transaction file. The next record in the data set is then available for verification.

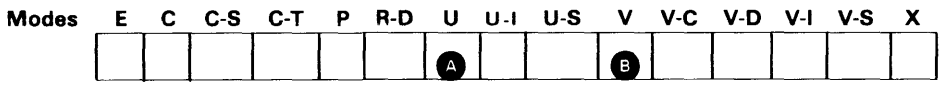
Record Display



A The record display function disables all data keys and changes the mode of operation to verify display mode. The entire record is displayed.

B The record display function returns the mode of operation to verify mode. Unverified positions of the record are blanked on the display, and the cursor is positioned beneath the next position to be verified.

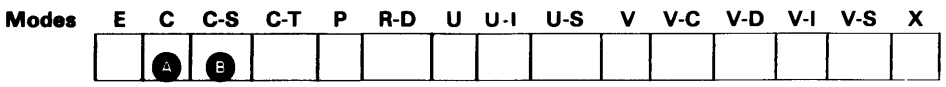
Record Insert



A The record insert function displays a prompting message and changes the mode of operation to update insert mode. The prompting message instructs the operator to enter a value to specify the number of records to be inserted into the data set assigned to the transaction file. See *Update Insert Mode*, earlier in this chapter.

B The record insert function displays a prompting message and changes the mode of operation to verify insert mode. The prompting message instructs the operator to enter a value to specify the number of records to be inserted into the data set assigned to the transaction file. See *Verify Insert Mode*, earlier in this chapter.

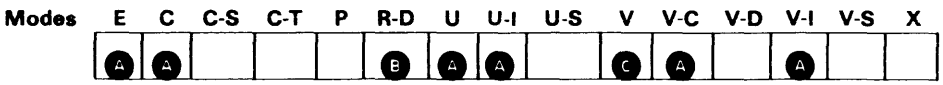
Record Transfer



A The record transfer function changes the mode of operation to copy transfer mode. After the displayed record is added to the data set assigned as the transaction file, the mode of operation returns to copy mode.

B The record transfer function can be used in copy search mode to write a group of consecutive records read from the data set assigned as the copy file to the data set assigned as the transaction file. The record transfer function, in this case, must be used in place of the record advance function to signify that the search parameters are entered. The search must be a forward search. Each record, starting with the record displayed when the search was initiated and continuing to, but not including the record specified by the search parameters, is added to the data set assigned as the transaction file. The record transfer function is invalid in both verify search mode and update search mode.

Reset



(A) The reset function can be used to restore operation when an error is posted on the status line, to cancel a prefix key, or to discontinue a keyboard state.

When an error is posted on the status line, the reset function resets the error message, restores the normal status line, and unlocks the keyboard for data entry.

The reset function can be initiated to cancel the effects of the Command key as a prefix in multi-key sequences.

The character insert keyboard state and the hexadecimal keyboard state can be discontinued by the reset function. See *Character Insert Function* or *Hexidecimal Function*.

(B) The reset function resets the error indication on the status line and enables the keyboard function keys.

(C) The reset function can be used to restore operation when an error is posted on the status line, to cancel a prefix key, or to discontinue a keyboard state.

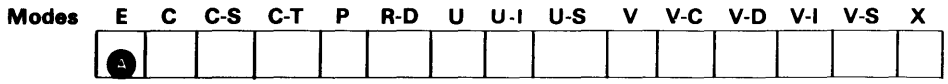
The reset function is initiated by the Reset key.

When an error is posted on the status line, the reset function resets the error message, restores the normal status line, and unlocks the keyboard.

The reset function can be initiated to cancel the effects of the Command key as a prefix in multi-key sequences.

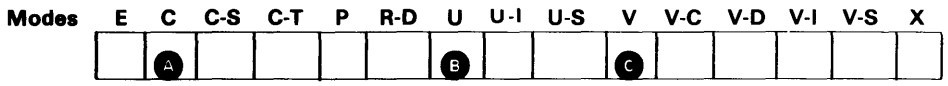
The hexadecimal keyboard state can be discontinued by the reset function.

Review File



A The review file function changes the mode of operation to copy mode. See *Copy Mode*, earlier in this chapter.

Search Content

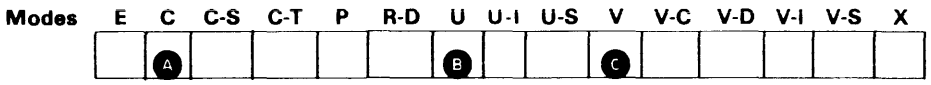


A The search content function displays a prompting message and changes the mode of operation to copy search mode (indicated by C-S in positions 35 through 37 in the status line). The prompting message instructs the operator to enter parameters to control the search. See *Copy Search Mode*, earlier in this chapter.

B The search content function displays a prompting message and changes the mode of operation to update search mode (indicated by U-S in positions 36 through 37 in the status line). The prompting message instructs the operator to enter parameters to control the search. See *Update Search Mode*, earlier in this chapter.

C The search content function displays a prompting message and changes the mode of operation to verify search mode (indicated by V-S in positions 35 through 37 in the status line). The prompting message instructs the operator to enter parameters to control the search. See *Verify Search Mode*, earlier in this chapter.

Search End-of-Data



(A) The search end-of-data function finds and displays the last record in the data set assigned as the copy file. While the data set is being searched, the mode changes to copy search mode (C-S in positions 35 through 37 in the status line).

When the record is found and displayed, the operating mode returns to copy mode. The record is then available for alteration and/or transfer to the transaction file.

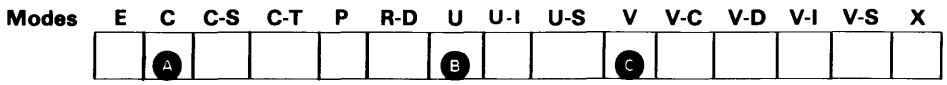
(B) The search end-of-data function finds and displays the last record in the data set assigned to the transaction file. While the data set is being searched, the mode changes to update search mode (U-S in positions 35 through 37 in the status line).

When the record is found and displayed, the operating mode returns to update mode. The record is then available for update activity.

(C) The search end-of-data function finds and displays the last record in the data set assigned to the transaction file. While the data set is being searched, the mode changes to verify search mode (V-S in positions 35 through 37 of the status line).

When the record is found and displayed, the operating mode returns to verify mode. The record is then available for verification.

Search Relative Record

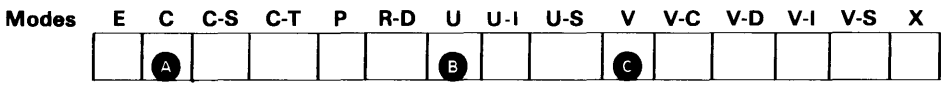


A The search relative record number function displays a prompting message and changes the mode of operation to copy search mode. The prompting message instructs the operator to enter the relative record number for the desired record.

B The search relative record number function displays a prompting message and changes the mode of operation to update search mode. The prompting message instructs the operator to enter the relative record number for the desired record.

C The search relative record number function displays a prompting message and changes the mode of operation to verify search mode. The prompting message instructs the operator to enter the relative record number for the desired record.

Search Sequential Content (Binary Search)

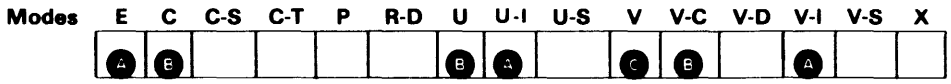


A The search sequential content function displays a prompting message and changes the mode of operation to copy search mode. The prompting message instructs the operator to enter parameters to control the search.

B The search sequential content function displays a prompting message and changes the mode of operation to update search mode. The prompting message instructs the operator to enter parameters to control the search.

C The search sequential content function displays a prompting message and changes the mode of operation to verify search mode. The prompting message instructs the operator to enter parameters to control the search.

Select Format



A The select format function allows the operator to select a format for use by keying the ID of the format as coded in the entry format statements.

At any time during the use of a format, pressing the Select Format key initiates the select format function, which is completed by the entry of the ID of an entry format. (For a two-character ID, the first character must be a letter, A through Z, and the second character must be a number, 0 through 9. A one-character ID is a number only.) When the number is entered, the current format is discontinued, and any data assembled for a record is discarded. The newly selected format is available for data entry with the cursor positioned in the first position of the first manual field.

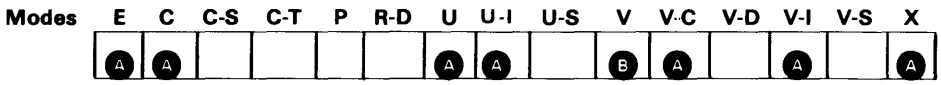
B The select format function allows the operator to select a format for use by keying the ID of the format as coded in the entry format statements.

At any time during the use of a format, pressing the Select Format key initiates the select format function, which is completed by the entry of the ID of an entry format. (For a two-character ID, the first character must be a letter, A through Z, and the second character must be a number, 0 through 9. A one-character ID is a number only.) When the number is entered, the current format is discontinued, and any data assembled is discarded. The same record is reread and is displayed with the cursor positioned in the first position of the first manual field.

C The select format function allows the operator to select a format for use by keying the ID of the format as coded in the entry format statements.

At any time during the verification of a record, pressing the Select Format key initiates the select format function, which is completed by the entry of the ID of an entry format. (For a two-character ID, the first character must be a letter, A through Z, and the second character must be a number, 0 through 9. A one-character ID is a number only.) When the number is entered, the current format is discontinued, and any data verified is blanked on the display. The entire original record can be verified under the control of the selected format.

Skip

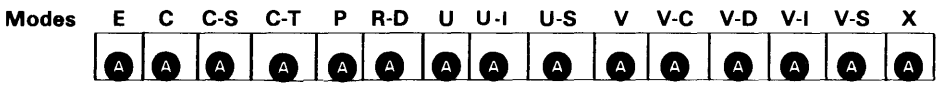


(A) The skip function fills the remainder of the field with blanks, and then starts the processing of any specified edits. If no errors are detected, the cursor advances to the next manual field.

When field-exit-pending is indicated, the skip function satisfies the field exiting requirement.

(B) The skip function verifies the remaining positions of a field for blanks. When a non-blank character is found, an error is indicated and the entire field is displayed. If the skip function is again initiated after the error is reset, the non-blank character is replaced by a blank and verification for blanks continues through the remainder of the field.

System Request



(A) The system request function allows the operator to temporarily discontinue the execution of the current program to load a program into another partition. At the completion of the program load operation, execution of the current program resumes, unless the new program was loaded into the partition in which execution was discontinued.

STATUS LINE

The status line, which appears on the top line of the display during the execution of a DE/RPG program, contains coded information about the status of the system.

Four status line formats are used depending upon the conditions being displayed.

```
                                1    1    2    2    3    3    4
                           1  5  0    5    0    5    0    5    0
                           ---------------------------------------
Normal Operation           X-XXXX      X XX XX XXXXXX X X XX X-X  X
KB Error                   X-XXXX-XXXX-X XX XX XXXXXX X X XX X-X  X
DE/RPG Edit Error          X      XXXX-X XX XX XXXXXX X X XX X-X  X
DE/RPG I/O Error           X XXXX-XXXX-XX XXXXXXXX XXXXXXXXXXXXXXXXX
```

Normal Operation

During program execution, while no errors are indicated, the status line is organized in the following format.

Position	Contents
1	Partition number – The number of the partition that relates to the display is represented by a hexadecimal value. Partition numbers range from 0 through 9 and A through F, with 0 being the first and F being the sixteenth.
2	Always blank.
3-6	Keystroke count – During data entry, the number displayed in these positions is the current keying position of the next keystroke to be entered relative to the beginning position of the record.
7-12	Blank.
13	Keyboard shift – The conditioning of the keyboard is represented by the following characters: A Alphabetic shift N Numeric shift H Hexadecimal X Alphabetic only Y Numeric only D Digits only W Right half shift V Right half only
14	Blank except when the character insert function is active. When character insert is active, position 14 contains the character >.
15-16	Positions remaining in field – The number of character positions remaining in the current field is displayed in these display positions. When the remaining positions to be keyed are greater than 99, two asterisks (**) are displayed. When the number of character positions remaining is 1 through 99, the actual number is displayed.
17	Blank.
18-19	Hexadecimal display – The hexadecimal value of EBCDIC character in the cursor position is displayed in positions 18 and 19.
20	Blank.
21-26	Record number – The relative record number is the count of both data records and deleted records from the start of the data set to the current record.

Position	Contents
27	Blank.
28	Auto dup/skip function status – When the automatic duplication function is enabled, this display position contains the letter D, which is displayed with image reversal. When the automatic duplication function is disabled, this position is blank.
29	Blank.
30	Automatic record advance function status – When the automatic record advance function is enabled, this display position contains the letter R with image reversal. When the automatic record advance function is disabled, this position is blank.
31	Blank.
32-33	Entry format identification – The ID assigned to the entry format being used or ready for use is displayed in these positions. (See the Format ID field on the Z-coding sheet.)
34	Blank.
35-37	Mode indication – The modes of program execution and certain command-key initiated functions are represented by the following characters: E Enter mode U Update mode U-S Update search mode U-I Update insert mode V Verify mode V-C Verify correct mode V-I Verify insert mode V-S Verify search mode V-D Verify display mode C Copy mode C-S Copy search mode C-T Copy transfer mode R-D Rerun display mode P Print mode X Execute mode
38	Blank.
39	Verify mark – While the the system is in verify or update mode, this position contains a V if the current record has been verified and the record contains the verify mark.

Keyboard and Edit Errors

Keyboard errors and errors detected by edit functions are indicated on the status line by a four-digit error code in positions 8 through 11. When an error occurs, the status line flashes or blinks until the Reset key is pressed. When the error is a keying error, the remaining positions of the status line are the same as normal. When the error is an edit error, the keystroke count (positions 3 through 6) is blanked on the status line and the remaining positions remain the same as normal.

The error codes used to indicate keyboard errors and edit errors are described in the *IBM 5280 Operator's Guide*.

I/O Errors

I/O device errors that occur during program execution are also indicated on the status line. An I/O error causes the following information to be displayed on the status line.

Position	Contents
1	Partition number (same as normal operation)
2	Blank
3-6	Physical address — a four-digit code that specifies the device
7	Dash character
8-11	Error code
12	Dash character
13-14	Logical device ID
15	Blank
16-23	Program name – The name assigned in the job specification statement
24	Blank
25-40	Data set label – These columns contain the last 16 positions of the data set label or the file name

PRODUCTION STATISTICS

Production statistics are accumulated by the system during the execution of DE/RPG data-entry programs. These statistics are maintained for each job and for each keyboard/display station. Job statistics are maintained in reserved counters only for the duration of the job. At the end of the job, the job statistics are added into reserved counters for the station statistics, and the job counters are reset.

Job Counters

Job counters are maintained for data-entry operations, update operations, and verify operations. The job counters are updated at the completion of each record and contain the following statistics:

- Number of keystrokes
- Record count
- Elapsed time in minutes
- Number of marked records

During verify operations, a separate count is maintained for the number of keystrokes in verify correct mode.

Station Counters

The counters maintained for each station contain an accumulation of the statistics from the job counters since IPL time. Additional station counters contain the number of jobs for which the statistics were collected.

Access to Production Statistics

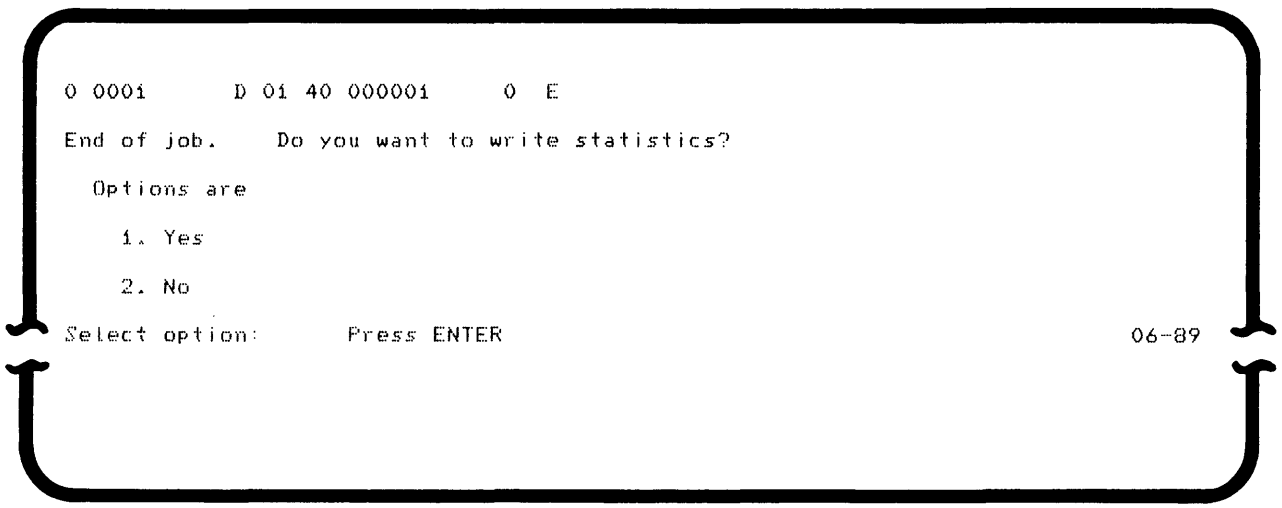
Production statistics can be accessed two ways. They can be retrieved under the control of a user's program or they can be written to a diskette data set at the end of a job. The statistics for a job or a station are accessible only at the station to which they apply.

User Program Access

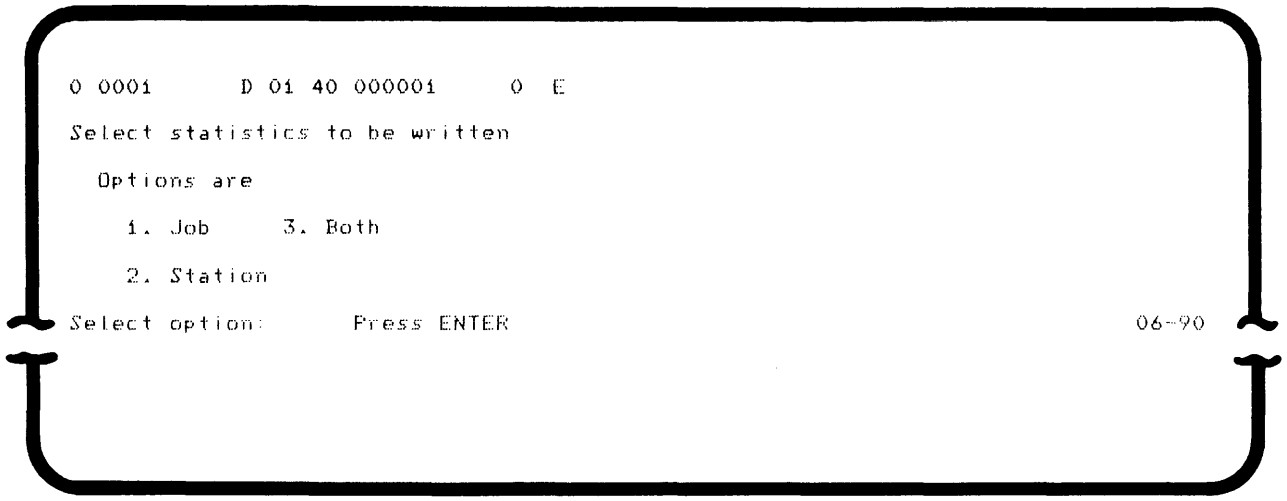
The production statistics counters can be referred to as a source of numeric data by keywords and calculation statements in a user's program, but the counter contents cannot be modified by the user's program. Program references to production statistics counters must be by the reserved name *STAT*n*, in which *n* is a number from 1 through 29. The counters are shown in the table under the heading *Production Statistics Table*, later in this section. The names, lengths, and contents of each counter are shown in the table.

Diskette Data Set Access

The second method for retrieving production statistics is through an end-of-job option. At end-of-job, the following menu is displayed:



When the operator selects the first option, the following display appears:



When one of the three options is selected, and if the statistics data set does exist, statistics are written. If the statistics data set does not exist, the operator is prompted to allocate the data set before the statistics can be written.

The statistics are written as a 128-byte record. If both job and station statistics are required, two records are written. The first record contains the job statistics and the second contains the station statistics.

Production Statistics Table

The following table shows the statistics that are accumulated during the execution of a DE/RPG program. Notice that space is allocated in the record for the verify correction keystrokes count in all three statistics groups, even though a value can only be expected in these positions for verify operations.

Type of Statistic	Positions	Length	Statistic	Station Counter Name	Job Counter Name
Enter	1-6	6	Keystrokes	*STAT01	*STAT17
	7-12	6	Records	*STAT02	*STAT18
	13-18	6	Marked records	*STAT03	*STAT19
	19-22	4	Elapsed time	*STAT04	*STAT20
	23-26	4	Verify correction keystrokes		
Update	27-32	6	Keystrokes	*STAT05	*STAT21
	33-38	6	Records	*STAT06	*STAT22
	39-44	6	Marked records	*STAT07	*STAT23
	45-48	4	Elapsed time	*STAT08	*STAT24
	49-52	4	Verify correction keystrokes		
Verify	53-58	6	Keystrokes	*STAT09	*STAT25
	59-64	6	Records	*STAT10	*STAT26
	65-70	6	Marked records	*STAT11	*STAT27
	71-74	4	Elapsed time	*STAT12	*STAT28
	75-78	4	Verify correction keystrokes	*STAT13	*STAT29
Enter	79-82	4	Jobs *	*STAT14	
Update	83-86	4	Jobs *	*STAT15	
Verify	87-90	4	Jobs *	*STAT16	

*Applies only to station statistics.

The job counts in positions 79 through 90 are maintained for station statistics. They do not apply to job statistics.

Keys which are included in the keystrokes count are as follows:

- Data keys which do not result in a character edit check error or verify mismatch error.
- Dup, Skip
- Character Advance, Character Backspace
- Field Exit, Field-, Field+
- Hex key sequences
- Enter/Record Advance
- Record Backspace
- Field Advance, Field Backspace

Keys not included in the keystroke counts are:

- Reset
- Shift keys, Shift Lock
- Data keys which result in an edit check error or verify mismatch error
- Cursor Right, Cursor Left, Cursor Up, Cursor Down, New Line
- Field Correct
- Keystrokes in field/record correct mode (first pass after correct key)
- Character Insert, Character Delete
- Command key sequence (except Hex key sequence)

The verify correction keystroke counter is used to count changes made to the original record in verify mode.

When job or station statistics are written to a diskette, the diskette records are 128 bytes in length. For a job statistics record, bytes 79 through 128 contain:

Bytes	Contents
79-118	Reserved
119-126	Job name
127	Partition number
128	J (to indicate job statistics)

For a station statistics record, bytes 91 through 128 contain:

Bytes	Contents
91-118	Reserved
119-126	Job name
127	Partition number
128	S (to indicate station statistics)

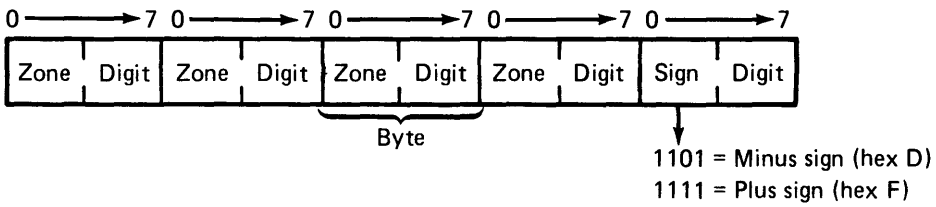
Chapter 10. Contents

Chapter 10. Data Formats

DE/RPG fields can be in alphameric, zoned decimal, packed decimal, or binary format on a diskette file. The compiler converts all numeric input fields to zoned decimal format for internal processing. The program's execution is not affected by whether the external numeric data is in zoned decimal format, packed decimal format, or binary format.

ZONED DECIMAL FORMAT

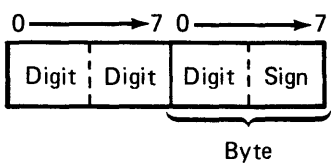
Zoned decimal format means that each byte of storage can contain one digit or one character. Any alphameric or numeric field can be read in zoned decimal format. In the zoned decimal format, each byte of storage is divided into two portions: a 4-bit zone portion and a 4-bit digit portion. The zoned decimal format looks like this:



The zone portion of the low-order byte indicates the sign (+ or –) of the decimal number. The standard signs are used: hex F for positive numbers and hex D for negative numbers. In zoned decimal format, each digit in a decimal number includes a zone portion; however, only the low-order zone portion serves as the sign. Figure 10 shows what the decimal number 8191 looks like in zoned decimal format.

PACKED DECIMAL FORMAT

Packed decimal format means that each byte of storage (except for the low-order byte) can contain two decimal digits. Each byte (except for the low-order byte) is divided into two 4-bit digit portions. The low-order byte contains one digit in the leftmost portion and the sign (+ or –) in the rightmost portion. The standard signs are used: hex F for positive numbers and hex D for negative numbers. The packed decimal format looks like this:



The sign portion of the low-order byte indicates whether the numeric value represented in the digit portions is positive or negative. Figure 10 shows what the decimal number 8191 looks like in packed decimal format.

To specify a packed decimal field for input or output, enter a P in the *Data Type* column (position 35). Use the following formula to calculate the maximum length of the external data field:

External length = (L÷2) + 1
where L = the internal (specified) length

Note: Any remainder in division is ignored.

Packed fields can be up to 8 bytes long. The following chart shows the packed equivalents for zoned decimal fields up to 15 digits long:

Specified Zoned Decimal Length in Digits	Resulting Packed Length in Bytes
15	8
14	8
13	7
12	7
11	6
10	6
9	5
8	5
7	4
6	4
5	3
4	3
3	2
2	2
1	1

BINARY FORMAT

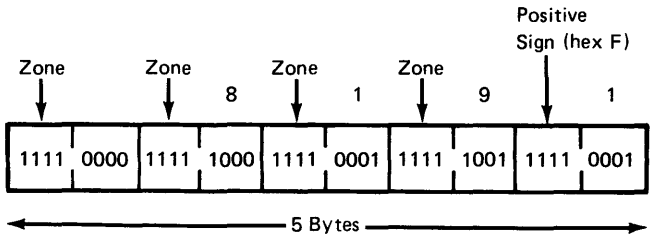
Binary format means that the sign (+ or –) is in the leftmost bit of the field and the integer value is in the remaining bits of the field. Positive numbers have a zero in the sign bit; negative numbers have a one in the sign bit and are in twos complement form. In binary format, each field must be either 2 or 4 bytes long.

To specify a binary field for input or output, enter a B in the *Data Type* column (position 35).

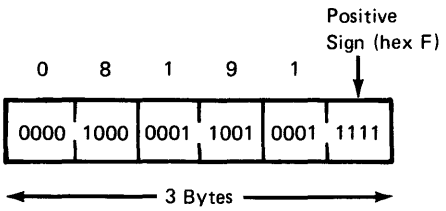
The length of a binary format field cannot exceed nine digits. If the specified length of the field is from 1 to 4 digits, the compiler assumes an external binary field length of 2 bytes. If the length of the field is from 5 to 9 digits, the compiler assumes a binary field length of 4 bytes.

Because the compiler converts a 1 to 4 digit field to 2 bytes of external binary data, it is possible for the external value to be too large to fit into the zoned internal field. If this happens, the leftmost digits of the field are dropped. For example, the compiler converts a binary field with a hex value of 7000 to 28 672 in zoned decimal. If the internal length is four digits, the 2 is dropped, and the result is 8672. If the internal length is three digits, the 2 and the 8 are dropped, and the result is 672.

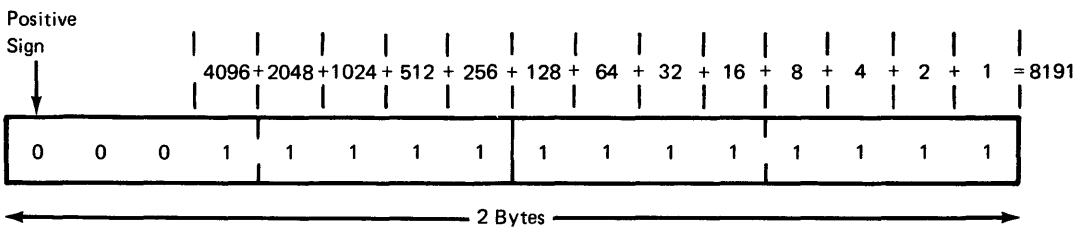
Zoned Decimal Format:[1]



Packed Decimal Format:



Binary Format:[2]



[1] If 8191 is stored in a zoned decimal field of 4 or 5 bytes, and it is converted to packed decimal format, it occupies 3 bytes.

[2] To obtain the numeric value of a positive binary number, add the values of the bits that are on (1); the sign bit is not included. To obtain the numeric value of a negative binary number, add the values of the bits that are off (0) plus one; the sign bit is not included.

Figure 10. Binary, Packed, and Zoned Decimal Representation of 8191

Signs

DE/RPG ensures that a consistent plus or minus sign is present for all numeric fields. The standard signs for all packed and zoned numeric fields are hex F for plus and hex D for minus.

Appendix A. EBCDIC Collating Sequence

Entry	Graphic	Entry	Graphic	Entry	Graphic	Entry	Graphic	Entry	Graphic
00		34		68		9C		D0	}
01		35		69		9D		D1	J
02		36		6A	¦	9E		D2	K
03		37		6B	,	9F		D3	L
04		38		6C	%	A0		D4	M
05		39		6D	_	A1	~	D5	N
06		3A		6E	>	A2	s	D6	O
07		3B		6F	?	A3	t	D7	P
08		3C		70		A4	u	D8	Q
09		3D		71		A5	v	D9	R
0A		3E		72		A6	w	DA	
0B		3F		73		A7	x	DB	
0C		40	blank	74		A8	y	DC	
0D		41		75		A9	z	DD	
0E		42		76		AA		DE	
0F		43		77		AB		DF	
10		44		78		AC		E0	\
11		45		79	`	AD		E1	
12		46		7A	:	AE		E2	S
13		47		7B	#	AF		E3	T
14		48		7C	@	B0		E4	U
15		49		7D	'	B1		E5	V
16		4A	¢	7E	=	B2		E6	W
17		4B	.	7F	"	B3		E7	X
18		4C	<	80		B4		E8	Y
19		4D	(	81	a	B5		E9	Z
1A		4E	+	82	b	B6		EA	
1B		4F	|	83	c	B7		EB	
1C		50	&	84	d	B8		EC	
1D		51		85	e	B9		ED	
1E		52		86	f	BA		EE	
1F		53		87	g	BB		EF	
20		54		88	h	BC		F0	0
21		55		89	i	BD		F1	1
22		56		8A		BE		F2	2
23		57		8B		BF		F3	3
24		58		8C		C0	{	F4	4
25		59		8D		C1	A	F5	5
26		5A	!	8E		C2	B	F6	6
27		5B	$	8F		C3	C	F7	7
28		5C	*	90		C4	D	F8	8
29		5D	)	91	j	C5	E	F9	9
2A		5E	;	92	k	C6	F	FA	
2B		5F	¬	93	l	C7	G	FB	
2C		60	-	94	m	C8	H	FC	
2D		61	/	95	n	C9	I	FD	
2E		62		96	o	CA		FE	
2F		63		97	p	CB		FF	
30		64		98	q	CC			
31		65		99	r	CD			
32		66		9A		CE			
33		67		9B		CF			

Note: The EBCDIC graphics are shown in this chart. See the *System Concepts* manual for the graphics displayed for other national character sets. Graphics are assigned to all blank positions (except hex 40) for maintenance purposes. These graphics are incompatible with other systems and cannot be used for exchange purposes.

Appendix B. ASCII Collating Sequence and Translate Table

The following table represents the ASCII line code to EBCDIC translations. The table shows the ASCII characters, their line codes, their IBM 5280 character representations on an IBM 5280 ASCII keyboard display, and the EBCDIC codes. If a blank appears in the IBM 5280 display column, the EBCDIC character in that row displays as a solid rectangular block.

ASCII Code	Control Character	IBM 5280 Display	EBCDIC Code
00	NUL		00
01	SOH		01
02	STX		02
03	ETX		03
04	EOT		37
05	ENQ		2D
06	ACK		2E
07	BEL		2F
08	BS	$\overline{6}$	16
09	HT		05
0A	LF		25
0B	VT		0B
0C	FF		0C
0D	CR		0D
0E	SO		0E
0F	SI		0F
10	DLE	$\overline{0}$	10
11	DC1	$\overline{1}$	11
12	DC2	$\overline{2}$	12
13	DC3	$\overline{3}$	13
14	DC4		3C
15	NAK		3D
16	SYN	$\overline{6}$	32
17	ETR	$\overline{7}$	26
18	CAN	$\overline{8}$	18
19	EM	$\overline{9}$	19
1A	SUB		3F
1B	ESC	$\overline{1}$	27
1C	FS	$\overline{*}$	1C
1D	GS		1D
1E	RS		1E
1F	US		1F
20	SP	blank	40
21		!	4F
22		"	7F
23		#	7B
24		$	5B
25		%	6C
26		&	50
27		'	7D
28		(	4D
29		)	5D
2A		*	5C
2B		+	4E
2C		,	6B
2D		−	60
2E		.	4B
2F		/	61
30		0	F0
31		1	F1
32		2	F2
33		3	F3
34		4	F4
35		5	F5
36		6	F6
37		7	F7
38		8	F8
39		9	F9
3A		:	7A
3B		;	5E
3C		<	4C
3D		=	7E
3E		>	6E
3F		?	6F

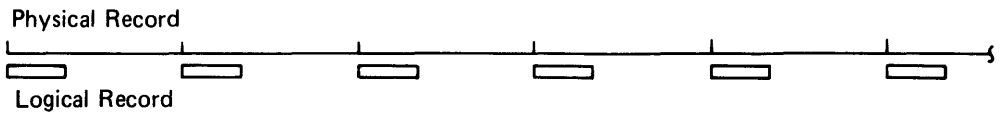
ASCII Code	Control Character	IBM 5280 Display	EBCDIC Code
40		@	7C
41		A	C1
42		B	C2
43		C	C3
44		D	C4
45		E	C5
46		F	C6
47		G	C7
48		H	C8
49		I	C9
$A		J	D1
4B		K	D2
4C		L	D3
4D		M	D4
4E		N	D5
4F		O	D6
50		P	D7
51		Q	D8
52		R	D9
53		S	E2
54		T	E3
55		U	E4
56		V	E5
57		W	E6
58		X	E7
59		Y	E8
5A		Z	E9
5B		[	4A
5C		\	E0
5D		]	5A
5E		‾	5F
5F		_	6D
60		`	79
61		a	81
62		b	82
63		c	83
64		d	84
65		e	85
66		f	86
67		g	87
68		h	88
69		j	89
6A		j	91
6B		k	92
6C		l	93
6D		m	94
6E		n	95
6F		o	96
70		p	97
71		q	98
72		r	99
73		s	A2
74		t	A3
75		u	A4
76		y	A5
77		w	A6
78		x	A7
79		y	A8
7A		z	A9
7B		{	C0
7C		¦	6A
7D		}	D0
7E		~	A1
7F	DEL		07

Appendix C. Diskette Data Set Organization and Access Methods

All diskette data sets that are created or used by DE/RPG programs can contain only fixed length logical records. That is, all records in a data set must be the same length. A logical record in a diskette data set can contain no more than 4096 character positions.

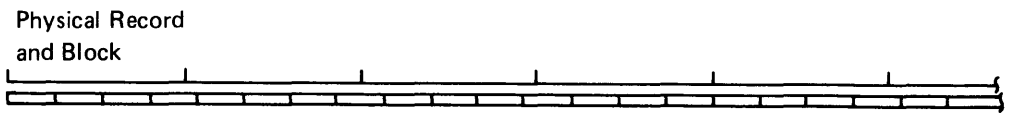
The way DE/RPG programs place the logical records on the diskette (the physical records) for a particular file is controlled by the parameters for the keyword BLKING in the file description statement for that file. Logical records can be placed on the diskette as either unblocked and unspanned or blocked and spanned.

Unblocked and unspanned means that each logical record begins at the start of a physical record (sector) and does not extend beyond the end of the sector, as follows:

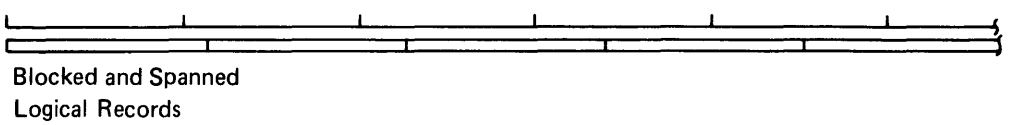


When records are blocked and spanned, the first record starts at the beginning of a physical record (sector). Thereafter, the records are placed contiguously without regard to sector boundaries. In the following drawing, example A shows logical records shorter than the physical records. Example B shows the logical records longer than the physical records.

Example A



Example B



RECORD ARRANGEMENT IN A DATA SET

Within the data set, records are arranged in one of the following ways:

- Sequential – where the records exist in the order in which they were written.
- Keyed sequence – where the records exist in an order that is based on the contents of a particular field (key field) in each record.
- Key index – where two data sets exist, one containing the data records and the other containing an index into the data records.

Sequential

When records are arranged sequentially in a data set, the records are written in the order in which they are entered. The first record entered is the first record in the data set; the last record entered is the last record in the data set. There are, however, key-initiated functions that allow the insertion of records between existing records in the data set assigned as the transaction file.

The sort program can be used to provide an ADDROUT index to the data set. As with a key index data set, two data sets exist to provide access to the data records. The sort program is used to select a group of records from the data set according to some specified criteria. The sort program puts the relative record numbers of the selected records into the ADDROUT index. The DE/RPG program uses the index to know which records in the data set are to be processed.

Keyed Sequence

When records are written in a data set according to a key field, the order of the records within the data set is based upon the contents of the key fields in the records. New records are inserted into the data set in ascending key sequence, and all the records that follow the new record are rewritten. (See *Performance Considerations in Using Keyed and Keyed Index Files* later in this section.)

Key Index

When a key index is used, two data sets exist to provide access to the data records. One data set contains the data records in sequential order; the order of the records is *not* based upon the contents of the key field. The other data set contains records made up of two fields: the key field (which corresponds to a field in a record of the first data set) and the location of the corresponding record in the first data set (the record length of this file is the key length plus 4). The order of the records in the index data set is based on the contents of the key fields. When new records are added, two records are produced, one in each data set. The data record is added at the end of the existing data records. The index record is inserted into the key index in the correct sequence according to the contents of the key field. (See *Performance Considerations in Using Keyed and Keyed Index Files*.)

Performance Considerations in Using Keyed and Keyed Index Files

Adding a new record to keyed or keyed index files can affect processing time. When a new record is added to an application file that is a keyed file or that uses a keyed index file for record access, a new record is inserted in the proper position in the file (keyed or keyed index). All of the following records are rewritten until either a deleted record is found or the last record in the file is reached. Therefore, a keyed or keyed index file containing periodically-spaced deleted records improves performance during a record add function.

The IBM 5280 diskette copy utility (SYSCOPY) can be used to insert deleted records into a file at user-specified intervals.

ACCESS METHODS

During the execution of DE/RPG programs, the records in data sets can be accessed by the following methods:

- Sequential
- Direct by relative record number
- Direct by key

The access methods that can be used for a data set depend both upon the use of the data set in the program and upon the arrangement of the records in the data set.

A data set can be assigned as the transaction file or the copy file, or can be controlled by calculation statements. Calculation statements must be used to control access to a diskette data set assigned as any file other than the transaction file or copy file.

Access Via Transaction File and Copy File

Records in data sets created via the transaction file are arranged sequentially. During update and verify modes of program execution, the records in the data set assigned to the transaction file are normally accessed in relative record number sequence. However, keyboard-initiated search functions can be used to locate specific records. The search functions include search by relative record number, search of specified positions for certain characters, and search of entire records for a character string. During rerun mode, access is sequential. Processing begins with the first logical record in the data set and progresses to the last logical record.

Sequential access is used to retrieve records in the data set assigned to the copy file. However, these records can be located with the same search functions as those used to locate records in the data set assigned to the transaction file. Once located, a single record or a series of consecutive records can be retrieved from the copy file and transferred to the data set assigned to the transaction file.

Access Via Calculation Statement Control

All data sets not accessed via the transaction file or the copy file must be controlled by calculation statements. The ways that records can be accessed depend upon certain characteristics of the data set as specified in the associated file description statement and field description statements (and the organization of the data set if it already exists).

Data sets with records that are arranged sequentially (no key fields) can be accessed as follows:

- Sequential (relative record sequence)
- Direct by ADDROUT index
- Direct by relative record number

Data sets with records that are arranged according to key fields can be accessed as follows:

- Sequential
- Direct by key

When key-indexed data sets are used, the file description statement must contain the INDEX keyword with the name of the index data set specified as the parameter. Key index data sets can be accessed as follows:

- Sequential (key sequence)
- Direct by key

Appendix D. Printer Uses

The printers available as part of the IBM 5280 system can be used during the execution of DE/RPG programs in either of two ways. A key-initiated print function can be used to print records as unformatted character strings, and formatted printing can be accomplished under program control.

KEY-INITIATED PRINTING

The key-initiated print function prints a selected record (or consecutive records) in the format in which they exist in the data set. This function can be used only if the transaction file is used in the program. In addition, the keyword PRTFILE must be used in the job specification source statement. A file description statement is also required in the source program to assign the printer for the function.

FORMATTED PRINTING

Formatted printing requires a file description statement to identify the printer, and record and field description statements to describe the placement or format of the printed data. A record produces a single line of print. The field description statements control the location (horizontal) of the data on the printed line.

A calculation statement (WRITE operation) is required for each line printed.

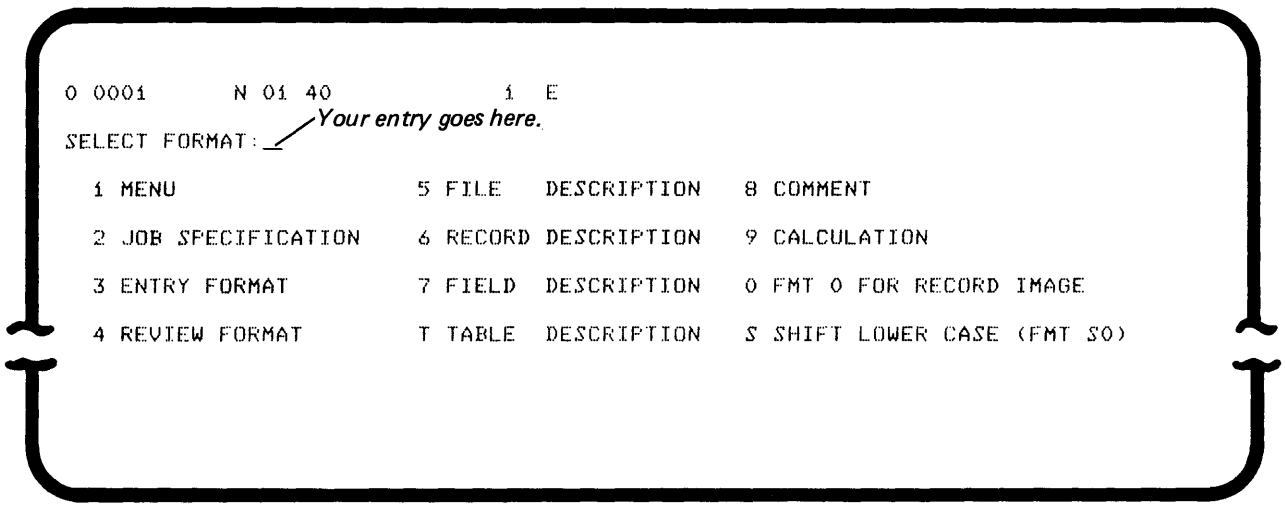
The vertical position of each printed line can be controlled by the use of SPACEB, SPACEA, SKIPB, and SKIPA keywords in the record description statements. These keywords specify the forms movement that is to occur both before printing and after printing the line (record).

The pertinent dimensions of the form, such as length, and the overflow line, can be described as parameters for the FORM keyword in the file description statement.

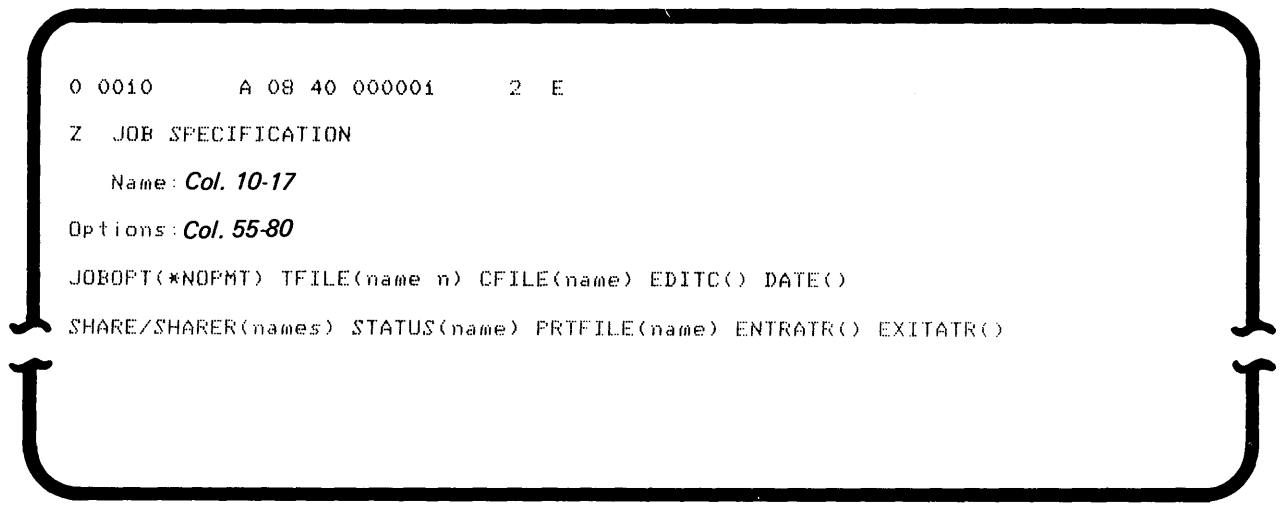
Appendix E. Source Entry Program Formats

This appendix shows the displays that are associated with each format in the Source Entry Program. The displays are shown in order by the format **numbers.**

The displays show the names of each field that are valid for a statement type. **As the displays are shown here, each field name is followed by the column numbers (in italics) where the field is coded on the specifications. This information is useful when you are entering source statements that have been coded on the Z-specifications, A-specifications, and C-specifications.**



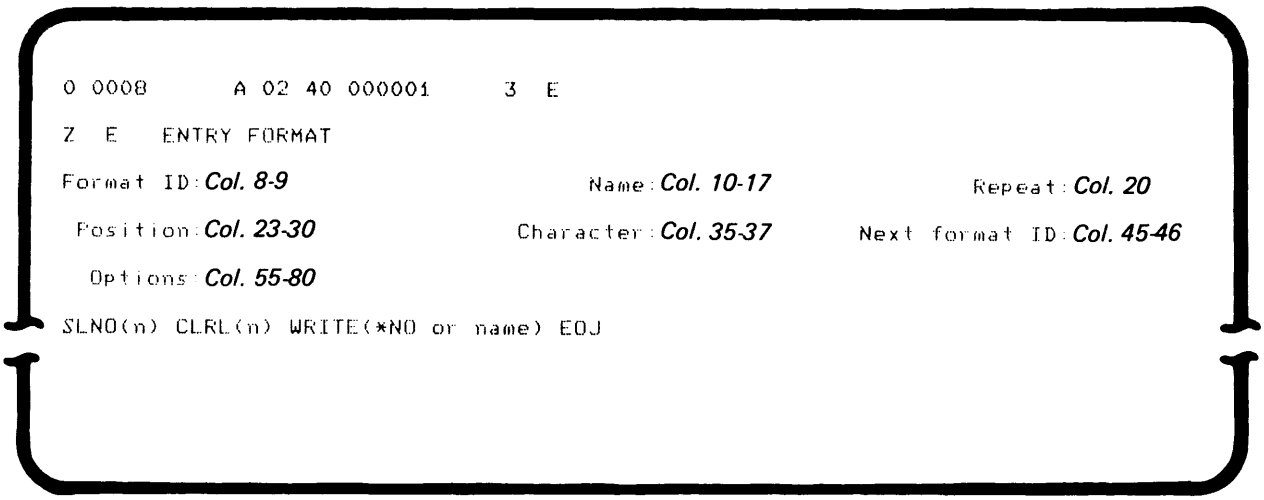
FORMAT ID: 1



FORMAT ID: 2

Format after Enter key is pressed: C3

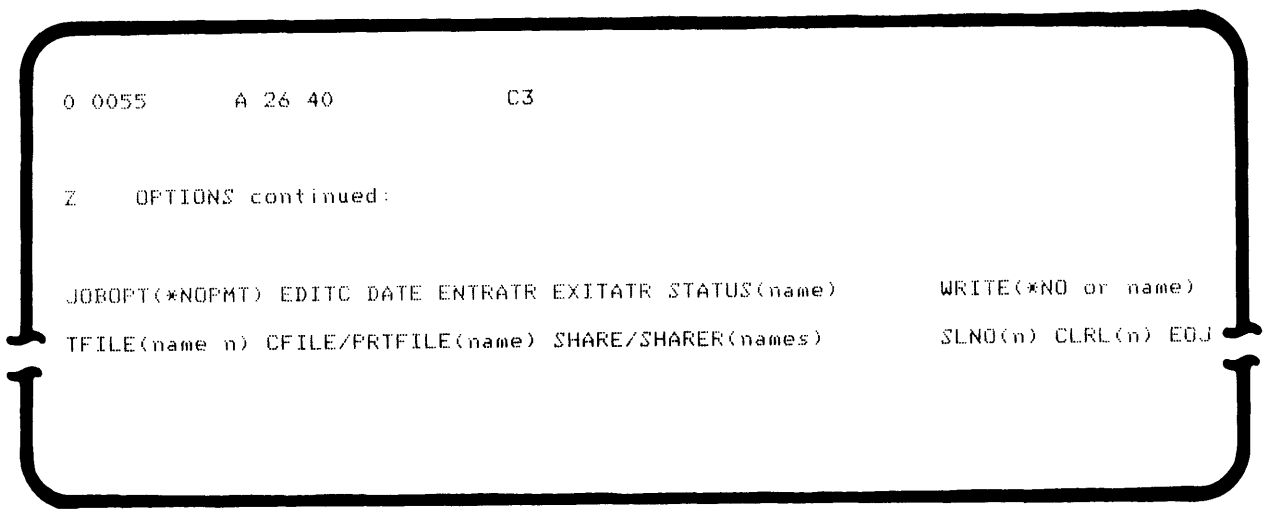
Format after Next Fmt key is pressed: C3



FORMAT ID: 3

Format after Enter key is pressed: 3

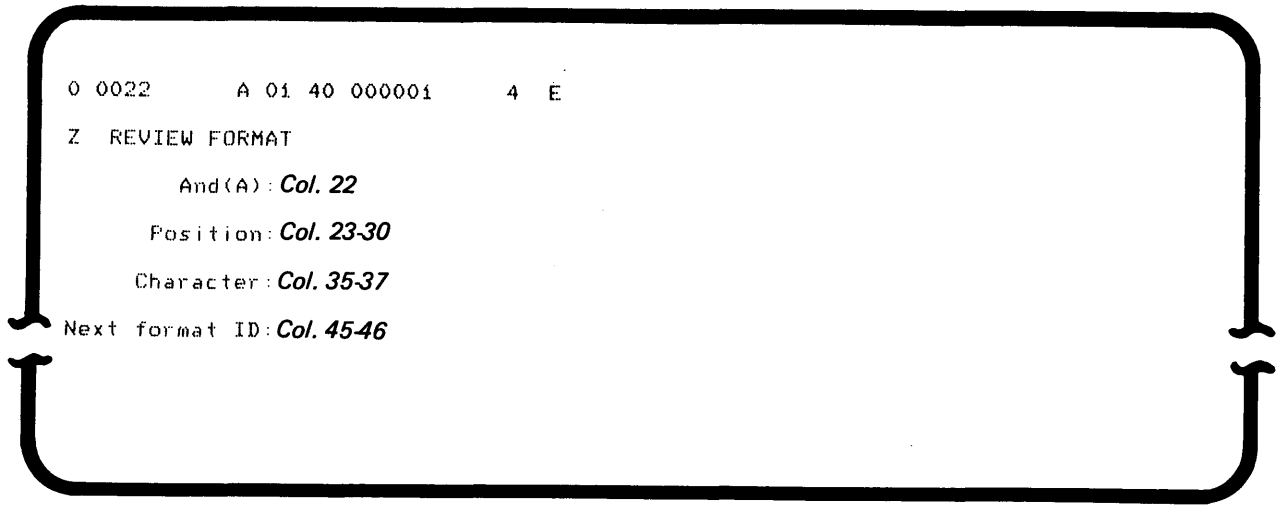
Format after Next Fmt key is pressed: 4



FORMAT ID: C3

Format after Enter key is pressed: C3

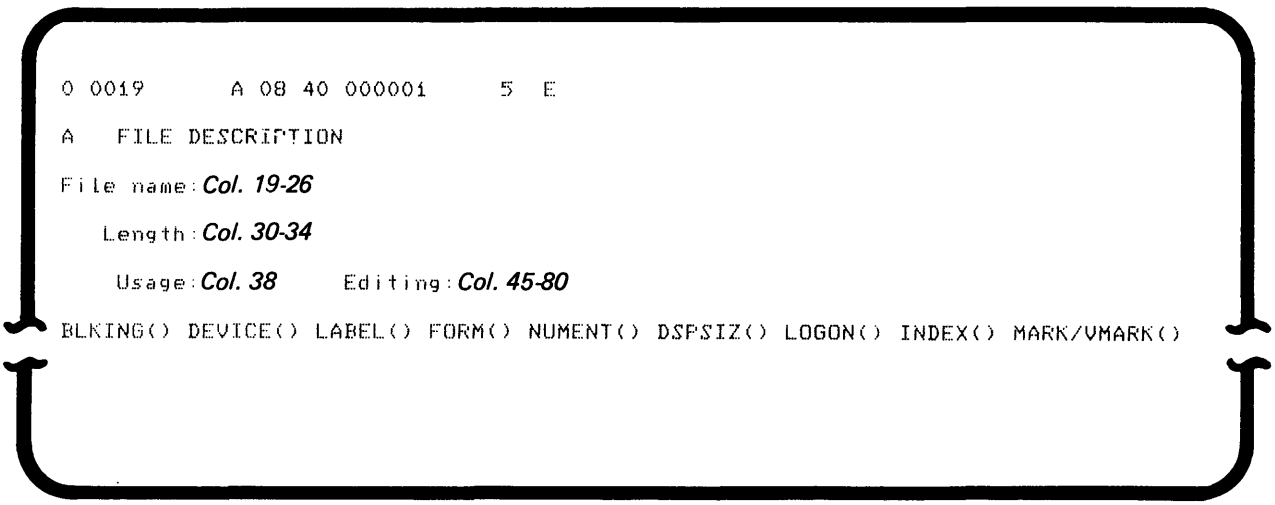
Format after Next Fmt key is pressed: 3



FORMAT ID: 4

Format after Enter key is pressed: 4

Format after Next Fmt key is pressed: 1



FORMAT ID: 5

Format after Enter key is pressed: 5

Format after Next Fmt key is pressed: 6

```
0 0045      A 36 40 000001     C5 E

A   FILE EDITING continued: Col. 45-80


BLKING() DEVICE() LABEL() FORM() NUMENT() DSPSIZ() LOGON() INDEX() MARK/VMARK()
```

FORMAT ID: C5

Format after Enter key is pressed: C5

Format after Next Fmt key is pressed: 5

```
0 0019      A 08 40 000001     6  E
A   RECORD DESCRIPTION
Record name: Col. 19-26
      Usage: Col. 38
    Editing: Col. 45-80
DSPATR() RECID() SPACEA(n) SPACEB(n) SKIPA(nnn) SKIPB(nnn)
```

FORMAT ID: 6

Format after Enter key is pressed: 7

Format after Next Fmt key is pressed: 7

```
0 0045      A 36 40 000001     C6 E

A   RECORD EDITING continued:Col. 45-80


DSPATR() RECID() SPACEA(n) SPACEB(n) SKIPA(nnn) SKIPB(nnn)
```

FORMAT ID: C6

Format after Enter key is pressed: 7

Format after Next Fmt key is pressed: 7

```
0 0009      Y 02 40 000001     7  E
A   FIELD DESCRIPTION
    Indicator:Col. 9-10 Name type(K):Col. 17 Field name:Col. 19-26      Length:Col. 30-34
    Data type:Col. 35 Decimal posns:Col. 37 Usage:Col. 38 Line:Col. 39-41 Posn:Col. 42-44
      Editing:Col. 45-80
ADD AUXDUP CHECK COMP DSPATR ERROR EXSR INSERT LOOK MMT RESET SEQ SHIFT SUBST...
```

FORMAT ID: 7

Format after Enter key is pressed: 7

Format after Next Fmt key is pressed: C7

Note: Format C7 allows you to make entries only in the *Editing* field on a continuation statement. If you want to specify an indicator to control the execution of keywords specified on the continuation line, you must use Format 7 for the line. When you have entered an indicator in the *Indicator* field, the Source Entry Program automatically advances the cursor to the *Editing* field.

```
O 0045      A 36 40 000001     C7 E

A   FIELD EDITING continued: Col. 45-80

ADD AUXDUP AUXST CHECK COMP DSPATR EDTCDE ERROR EXSR INSERT LOOK MMT RANGE
RANGET RESET SEQ SETOF SETON SHIFT SUB SUBST TADD TSUB XCHK
```

FORMAT ID: C7

Format after Enter key is pressed: C7

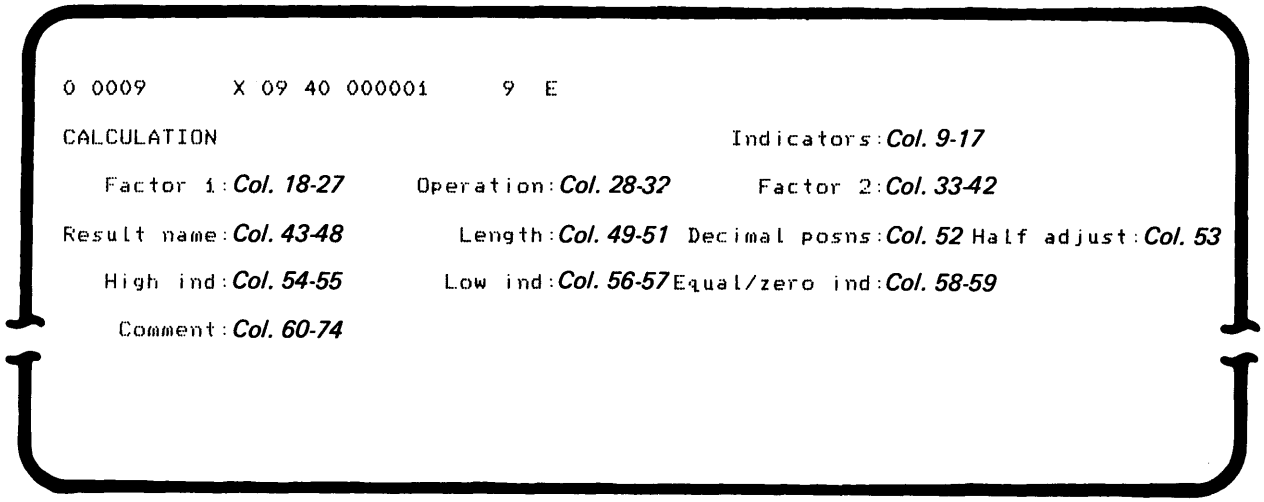
Format after Next Fmt key is pressed: 7

```
O 0008      A 73 40 000001     8  E
COMMENT
          Comment starts here.
     *_/
          |         |         |         |         |         |         |         |
```

FORMAT ID: 8

Format after Enter key is pressed: 8

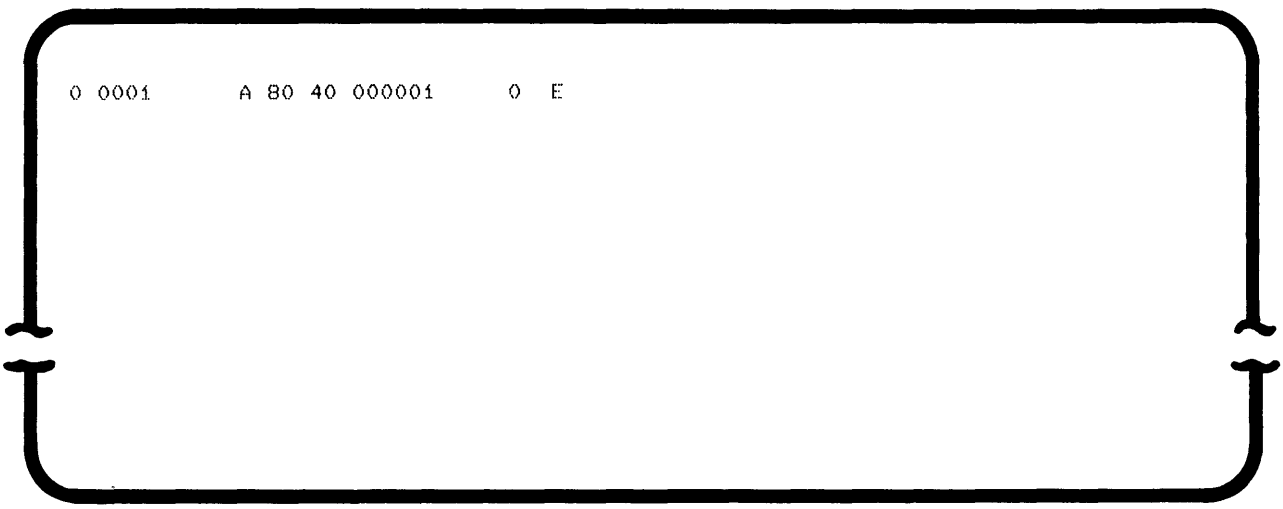
Format after Next Fmt key is pressed: 1



FORMAT ID: 9

Format after Enter key is pressed: 9

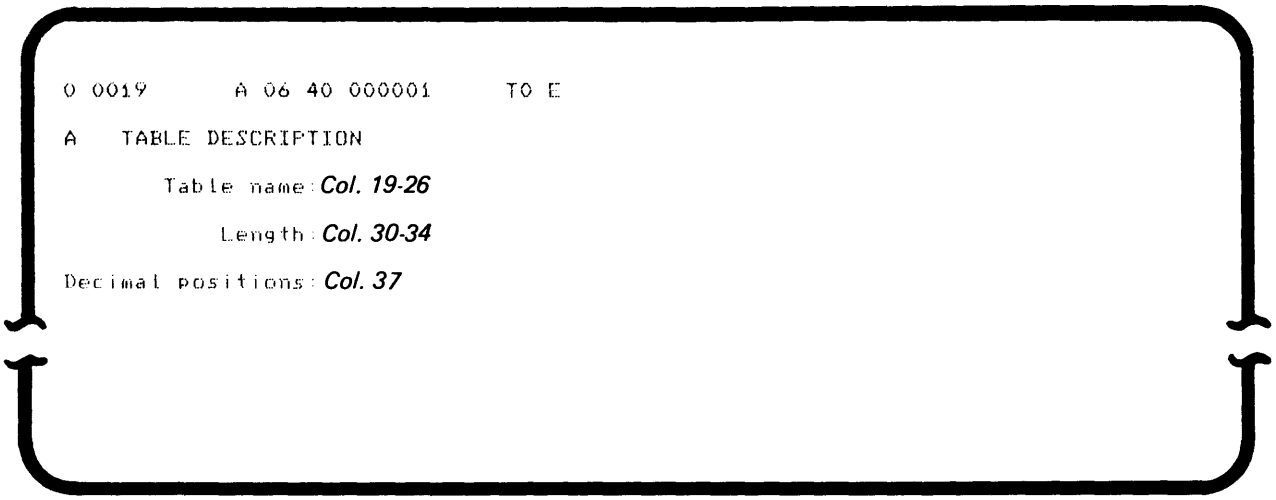
Format after Next Fmt key is pressed: 1



FORMAT ID: 0

Format after Enter key is pressed: 0

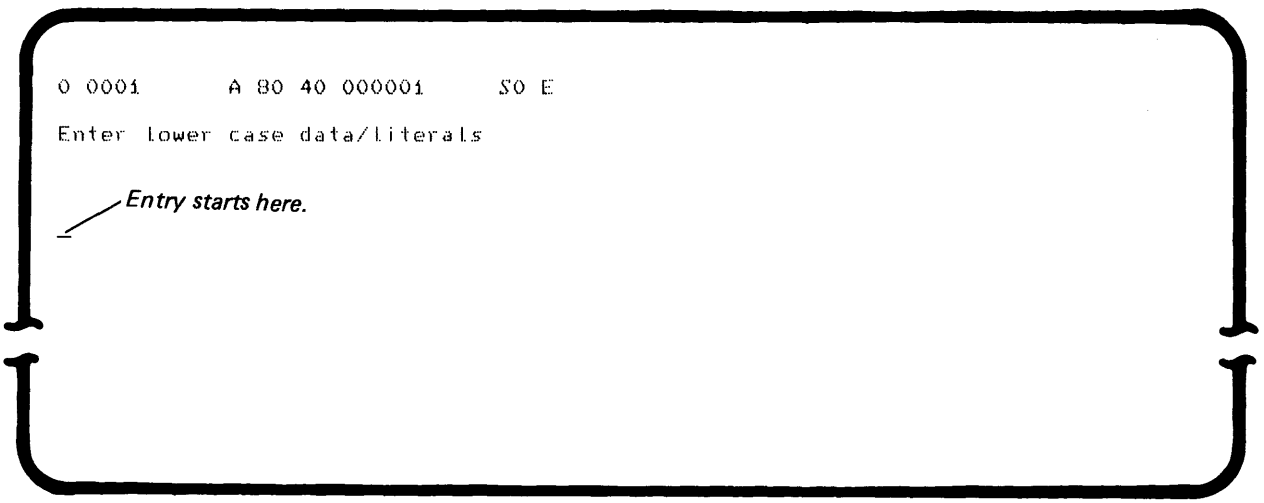
Format after Next Fmt key is pressed: 0



FORMAT ID: T0

Format after Enter key is pressed: T0

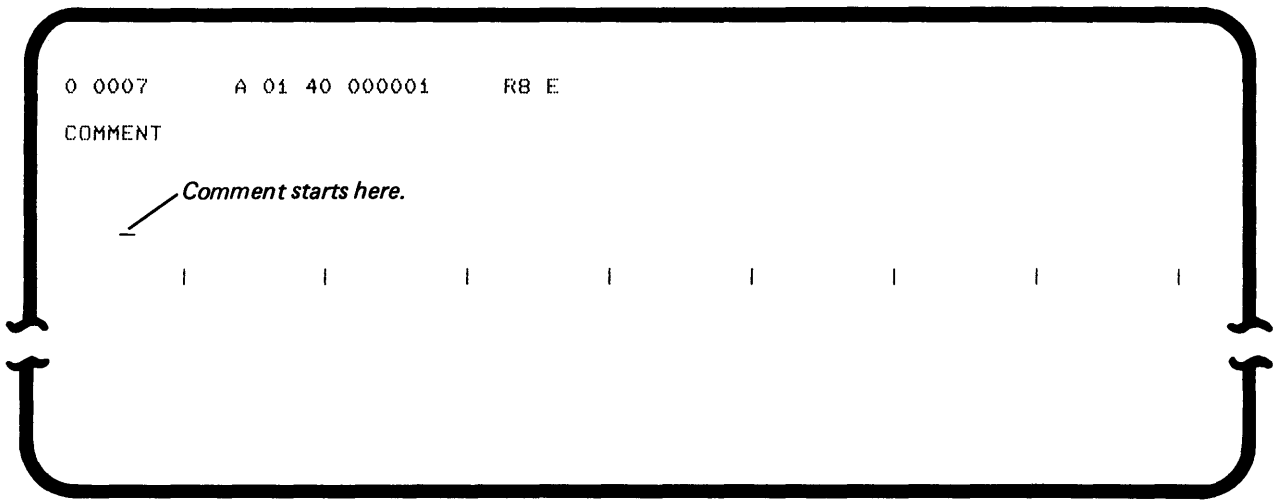
Format after Next Fmt key is pressed: 1



FORMAT ID: S0

Format after Enter key is pressed: S0

Format after Next Fmt key is pressed: 1



FORMAT ID: R8

Format after Enter key is pressed: R8

Format after Next Fmt key is pressed: 1

Appendix F. DE/RPG Compiler Error Messages

This appendix contains the messages that can be produced by the DE/RPG compiler. The messages are listed in order by the message number, which is shown on the compiler listing. For most of the messages, an explanation is provided that further defines the error that caused the messages.

Often an error in one DE/RPG source statement causes errors to be detected in one or more other statements. For example, if an error is made in a field description statement that causes the compiler to ignore the entire statement, statements that refer to that field (such as calculation specifications) will produce errors indicating an undefined field was specified. Correcting the field description statement will then correct the other statements as well.

Source errors are identified as informational, error, or severe. Informational errors do not cause the compiler to halt. If no severe errors are detected, the user has the option to let the compiler continue to completion without issuing a halt. Once compilation is complete, the display screen shows the number of errors found.

0001 No Z- and or A-specifications found in source

Severe

The source program data set must contain the following Z-specifications:

- One and only one job specification
- At least one entry format statement

The source program must also contain A-specifications

0002 Specification types not valid (Z, A, C, tables)

Severe

Specifications must be entered in the following order: Z, A, C, and tables. The source line is ignored.

0003 Z form statement does not contain *, J, or blank (Column 7)

Severe

Column 7 of the Z-specification (Name Type) must contain:

- * for a comment statement
- J for a job statement
- Blank for an entry format statement, a review format statement, a continuation line, or a secondary line.

Any other entry is invalid. The source line is ignored.

0004 Columns 8 and 9 (ID field) must be blank

Error

Columns 8 and 9 of the Z-specification must be blank in a job statement, a review format statement, a continuation line, or a secondary line. Blanks are assumed.

0005 Columns 10 through 17 (name field) does not contain a valid name

Error

The first character of a name must be A through Z, $, #, or @. The remaining characters may be A through Z, 0 through 9, $, #, or @. The name is ignored.

0006 Columns 20 through 54 must be blank

Error

No entries are allowed in columns 20 through 54 on a job specification statement. Blanks are assumed.

0007 Column 21 (mode field) must contain E or R

Severe

Column 21 must contain an E for an entry format statement or an R for a review format statement. The source line is ignored.

0008 Columns 8 and 9 (ID field) are invalid

Severe

If Column 8 is:	Column 9 can be:
Blank	Blank
A-Z	0-9
1 through 9	Blank

Entries in this field can be made only when the mode field contains an E. The field is ignored.

0009 Name is incorrectly specified in name field

Severe

Because an E was specified in column 21, the name field must contain a valid name. The name must be left-justified at column 10 and cannot exceed eight characters. To be valid, the first character of the name must be A-Z, $, #, or @. The remaining characters can be A-Z, 0-9, $, #, or @. The field is ignored.

0010 Column 20 (repeat field) is invalid

Error

The repeat field must be blank unless the mode field contains an E. When the mode field contains an E, valid entries are blank, N, or 1 through 9. The field is ignored.

0011 Column 22 (and field) is invalid

Error

Column 22 must be blank or contain an A (and). The field is ignored.

0012 Columns 7 through 54 must be blank on a continuation statement

Severe

The source line is ignored.

0013 Columns 7 through 20 must be blank for review format statements

Error

Blanks are assumed.

0014 Columns 23 through 32 (position to be tested field) are invalid

Error

This field must contain blanks or *POSnnnn, where nnnn is a character position within the record. The field is ignored.

0015 Columns 33 and 34 (test condition field) are invalid

Error

This field must contain EQ or blanks. The field is ignored.

0016 Columns 35 through 44 (character to test for field) are invalid

Error

A single character literal must be enclosed in apostrophes and must be left-justified at column 35. The field is ignored.

0017 Columns 45 through 54 (next field) are invalid

Error

Valid entries for the next format ID are 0 through 9 in column 45 or A0 through Z9 in columns 45 and 46. The field is ignored.

0018 Keyword or parameter specified is invalid on Z-form

Error

The specified keyword is not defined as a keyword or the parameters are specified incorrectly. Check the use of the apostrophes, ensure that the parameters are within parentheses, and ensure that both left and right parentheses are used. The keyword or parameter is ignored.

0019 No entry format specification entered (Z-specification)

Severe

Every source program must contain at least one entry format statement following the job specification statement.

0020 Column 7 of A-specification must be blank or *

Error

Column 7 must contain an asterisk if this is a comment line; otherwise column 7 must be blank. The field is ignored.

0021 Columns 7 through 44 must be blank in a continuation line statement

Error

The source line is ignored.

0022 Column 8 of A-specification must be blank

Error

Blanks are assumed.

0023 Columns 9 through 16 (indicator) are not 01 through 99

Error

Blanks are assumed.

0024 Columns 17 and 18 (name type field) are invalid

Severe

Column 17 must be blank, R, F, T, or K and 18 must be blank. The field is ignored.

0025 Columns 19 through 28 (name field) are invalid

Error

The name field must:

- Be left-justified at column 19
- Be eight characters or less for file and record names.
- Be six characters or less for field and table names.
- Begin with A-Z, $, #, @ or *RTN (field description statement only) and contain only the characters A-Z, $, #, @, 0-9. For fields, *TOTn, *STATnn, *RTN, and UPDATE are also valid. The field is ignored.

0026 Column 29 must be blank

Error

Blank is assumed.

0027 Columns 30 through 34 (length field) are invalid

Error

Data in this field must be right-justified and contain all blanks or only numeric data (not to exceed 8192). The field is ignored.

0028 Column 35 (data type field) is invalid; blank assumed

Error

The valid entries are A (alphabetic shift), C (character check), B (binary), D (digits only), H (hexadecimal), N (numeric shift), P (packed decimal), S (signed numeric), V (right half only), W (right half shift), X (alphabetic only), Y (numeric only), and blank.

0029 Columns 36 and 37 (decimal positions field) are invalid

Error

This field must be blank or contain a numeric entry less than or equal to 9. (Column 36 must be blank.) The field is ignored.

0030 Column 38 (usage field) must contain blank, I, O, B, or W

Error

The meanings of the entries are:

- I for input
- O for output
- B for both input and output
- W for work space

Blanks are assumed.

0031 Columns 39 through 44 (location/line/position field) are invalid

Error

This field must be blank or contain numeric data. Blanks are assumed.

0032 Keyword, literal or parameter invalid on A-specification

Error

Either an invalid keyword is specified or the literal or parameter is specified incorrectly. Data must be specified between parentheses, apostrophes must be specified correctly, and if the data cannot be completed on one line, a continuation character (+ or -) must be specified as the last nonblank character in the line. The keyword, parameter, or literal is ignored.

0034 New specification type in continuation

Severe

For a continuation line, primary line options are extended. A new specification type cannot be specified on a continuation line. The preceding source line is ignored.

0035 Compiler work buffer area exceeded

Severe

This problem can be solved by changing continuation lines into secondary lines for the keywords or by reducing the length of prompts or literals.

0036 Invalid secondary line; line ignored

Error

A secondary line was specified as the first noncomment line in the program or came after a line having an invalid name-type in column 7.

0051 Column 9 must be blank or contain an N

Error

Blanks are assumed.

0052 Column 12 must be blank or contain an N

Error

Blanks are assumed.

0053 Column 15 must be blank or contain an N

Error

Blanks are assumed.

0056 Columns 7 and 8 (control) must be blank

Error

Blanks are assumed.

0061 Columns 10 and 11 (condition indicator) not valid; must be 01 through 99

Error

The indicator specified must be 01 through 99. No other entries are valid. Blanks are assumed.

0062 Columns 13 and 14 (condition indicator) not valid; must be 01 through 99

Error

The indicator specified must be 01 through 99. No other entries are valid. Blanks are assumed.

0063 Columns 16 and 17 (condition indicator) not valid; must be 01 through 99

Error

The indicator specified must be 01 through 99. No other entries are valid. Blanks are assumed.

0065 Operation specified in columns 28 through 32 is invalid

Error

The operation must be entered correctly; the columns all contain blanks, or the operation is not left-justified at column 28. The field is ignored.

0068 Columns 18 through 27 (factor 1) are invalid

Error

Valid entries must be left-justified at column 18 and qualify as one of the following:

- A defined field name
- An alphameric literal enclosed in apostrophes
- A numeric literal not enclosed in apostrophes; may have a leading sign (+ or -) and a decimal position specified
- A table name
- Label for TAG, BEGSR, ENDSR operation

The field is ignored.

0069 Columns 33 through 42 (factor 2) are invalid

Error

Valid entries must be left-justified at column 18 and qualify as one of the following:

- A defined field name
- An alphameric literal enclosed in apostrophes
- A numeric literal not enclosed in apostrophes; may have a leading sign (+ or -) and a decimal position specified
- A table name
- Label for TAG, BEGSR, ENDSR operation
- A file name or record name for READ operation

The field is ignored.

0070 Columns 43 through 48 (result field) are invalid

Error

An entry in this field must be left-justified at column 43 and be a valid name. The field is ignored.

0071 Columns 49 through 51 (field length) are invalid

Error

The entry in these columns must be right-justified with no embedded blanks. Values of 1 through 256 are valid. The field is ignored.

0072 Column 52 (decimal positions) must be blank or 0 through 9

Error

Blank indicates a character field. The digits 0 through 9 indicate the field is numeric and specify the number of decimal positions. An entry of 0 must be made if a numeric field contains no decimal position. The field is ignored.

0073 Column 53 (half-adjust) invalid; must be blank or H

Error

Blank specified that half-adjust is not to be performed; H specifies half-adjust is to be performed. The field is ignored.

0075 Columns 54 and 55 (result indicator) not valid; must be 01 through 99

Error

The indicator specified must be 01 through 99. No other entries are valid. The field is ignored.

0076 Columns 56 and 57 (result indicator) not valid; must be 01 through 99

Error

The indicator specified must be 01 through 99. No other entries are valid. The field is ignored.

0077 Columns 58 and 59 (result indicator) not valid; must be 01 through 99

Error

The indicator specified must be 01 through 99. No other entries are valid. The field is ignored.

0080 Invalid file name specified

Severe

The file name on the table definition statement (**CTDATA) is more than 8 characters or contains invalid characters. The table is ignored.

0081 Table characters are not hexadecimal digits

Severe

Alternate collating sequence and file translation tables must consist of pairs of hexadecimal data (such as C1C2). ALTSEQ tables can contain the reserved word ASCII. The source line is ignored.

0082 No data for table

Error

No data for a defined table was found following the **CTDATA, **FTRANS, or **ALTSEQ statement.

0083 Invalid data set name specified

Severe

A data set name may be from 1 to 8 character positions long. The first character must be A-Z, $, #, or @. The remaining characters must be A-Z, 0-9, $, #, or @. The source line is ignored.

0084 Columns 1 through 8 invalid; must contain **ALTSEQ

Severe

The title line of an alternate collating sequence table must contain **ALTSEQ in columns 1 through 8. The source line is ignored.

0085 Hexadecimal data and ASCII mixed, or ASCII specified more than once

Error

Alternate collating sequence tables must either have hexadecimal data provided or specify the keyword ASCII. Hexadecimal data and ASCII cannot be combined for one table. ASCII cannot be specified more than once for one table definition. The source line is ignored.

0086 Invalid compile time table header

Severe

The characters following the ** are not CTDATA, ALTSEQ, FTRANS, or SLFCHK; or more than one **ALTSEQ statement is specified. The table is ignored.

0091 Self-check keyword or parameter is in error

Severe

The valid keywords are MOD, DISP, WEIGHTS, and OPT. The parameter must be enclosed in left and right parentheses. The keyword is ignored.

0092 Self-check header record is invalid

Severe

The **SLFCHK header statement must follow the following format:

Column	Contents
9	Blank
10 and 11	Self-check number
12 through 18	Blank
19-80	Comments (optional

Blanks are assumed.

0100 More than 204 errors occurred, compile terminated

Severe

The compiler area reserved for errors will only hold 204 errors. Correct the detected errors and compile the program again.

0180 The first A-specification must be a file statement

Severe

All A-specifications that precede the first file description statement are ignored.

0181 This field statement has no length or name

Severe

Each field description statement must have either a name specified, a length specified, or both. If a name is not specified, the field cannot be referenced elsewhere in the program. If the length is not specified, the field must be defined elsewhere in the program. The statement is ignored.

0182 Invalid name entry

Error

A name of the form *xxxx was found but was not *RTN, *STATnn, or *TOTn. The name entry is ignored.

0197 INDEX keyword is valid only on a file statement; ignored

Error

0198 LABEL keyword is valid only on a file statement; ignored

Error

0199 LOGON keyword is valid only on a file statement; ignored

Error

0200 DEVICE keyword is valid only on a file statement; ignored

Error

0201 BLKING keyword is valid only on a file statement; ignored

Error

0202 DSPSIZ keyword is valid only on a file statement; ignored

Error

0203 NUMENT keyword is valid only on a file statement; ignored

Error

0204 FORM keyword is valid only on a file statement; ignored

Error

0206 SKIPA keyword is valid only on a record statement; ignored

Error

0207 SKIPB keyword is valid only on a record statement; ignored

Error

0208 SPACEA keyword is valid only on a record statement; ignored

Error

0209 SPACEB keyword is valid only on a record statement; ignored

Error

0210 RECID keyword is valid only on a record statement; ignored

Error

0213 DSPATR valid only on file, record, or field statements

Error

The keyword is ignored.

0214 CHECK valid only on file, record, or field statements

Error

The keyword is ignored.

0216 COMP keyword is valid only on a field statement; ignored

Error

0217 XCHK keyword is valid only on a field statement; ignored

Error

0218 RANGE keyword is valid only on a field statement; ignored

Error

0219 RANGET keyword is valid only on a field statement; ignored

Error

0220 LOOK keyword is valid only on a field statement; ignored

Error

0221 SEQ keyword is valid only on a field statement; ignored

Error

0222 SHIFT keyword is valid only on a field statement; ignored

Error

0223 PMT keyword is valid only on a field statement; ignored

Error

0224 ADD keyword is valid only on a field statement; ignored

Error

0225 SUB keyword is valid only on a field statement; ignored

Error

0226 AUXDUP keyword is valid only on a field statement; ignored

Error

0227 AUXST keyword is valid only on a field statement; ignored

Error

0228 EDTCDE keyword is valid only on a field statement; ignored

Error

0229 INSERT keyword is valid only on a field statement; ignored

Error

0230 RESET keyword is valid only on a field statement; ignored

Error

0231 SUBST keyword is valid only on a field statement; ignored

Error

0232 TADD keyword is valid only on a field statement; ignored

Error

0233 TSUB keyword is valid only on a field statement; ignored

Error

0234 ERROR keyword is valid only on a field statement; ignored

Error

0235 EXSR keyword is valid only on a field statement; ignored

Error

0237 BLKING cannot be specified for this file type; ignored

Error

BLKING is valid on a diskette, printer, or comm file. If not on a diskette file, the only valid parameter is *DBL.

0239 RECID valid only for diskette or comm files; ignored

Error

0240 NUMENT valid only for diskette or compile-time table files

Error

0242 FORM valid only for a printer file; keyword ignored

Error

0243 SKIPA valid only for a printer file; keyword ignored

Error

0244 SKIPB valid only for a printer file; keyword ignored

Error

0245 SPACEA valid only for a printer file; keyword ignored

Error

0246 SPACEB valid only for a printer file; keyword ignored

Error

0247 LABEL valid only for a diskette file; keyword ignored

Error

The LABEL keyword is used to specify a data set label for use as a header on a diskette when the header is different from the entry in the name field.

0249 DSPATR valid only for a CRT file; keyword ignored

Error

0250 CHECK valid only for a CRT file; keyword ignored

Error

0251 COMP valid only for a CRT file; keyword ignored

Error

0252 XCHK valid only for a CRT file; keyword ignored

Error

0253 RANGE valid only for a CRT file; keyword ignored

Error

0254 RANGET valid only for a CRT file; keyword ignored

Error

0255 RESET valid only for a CRT file; keyword ignored

Error

0256 LOOK valid only for a CRT file; keyword ignored

Error

0257 SEQ valid only for a CRT file; keyword ignored

Error

0258 SHIFT valid only for a CRT file; keyword ignored

Error

0259 PMT valid only for a CRT file; keyword ignored

Error

0260 ADD valid only for a CRT file; keyword ignored

Error

0261 SUB valid only for a CRT file; keyword ignored

Error

0262 AUXDUP valid only for a CRT file; keyword ignored

Error

0263 AUXST valid only for a CRT file; keyword ignored

Error

0264 EXSR valid only for a CRT file; keyword ignored

Error

0265 INSERT valid only for a CRT file; keyword ignored

Error

0266 SUBST valid only for a CRT file; keyword ignored

Error

0267 TADD valid only for a CRT file; keyword ignored

Error

0268 TSUB valid only for a CRT file; keyword ignored

Error

0269 DSPSIZ valid only for a CRT file; keyword ignored

Error

0270 ERROR valid only for a CRT file; keyword ignored

Error

0272 SETOF only for record statement or field statement; ignored

Error

0273 SETON only for record statement or field statement; ignored

Error

0275 SETOF only for CRT/DISK/COMM files; keyword ignored

Error

0276 SETON only for CRT/DISK/COMM files; keyword ignored

Error

0277 INDEX valid only for a diskette file; keyword ignored

Error

0279 DEVICE may only occur once per statement; keyword ignored

Error

0280 DSPSIZ may only occur once per statement; keyword ignored

Error

0281 BLKING may only occur once per statement; keyword ignored

Error

0282 NUMENT may only occur once per statement; keyword ignored

Error

0283 FORM may only occur once per statement; keyword ignored

Error

0284 SKIPA may only occur once per statement; keyword ignored

Error

0285 SKIPB may only occur once per statement; keyword ignored

Error

0286 SPACEA may only occur once per statement; keyword ignored

Error

0287 SPACEB may only occur once per statement; keyword ignored

Error

0288 SUBST may only occur once per statement; keyword ignored

Error

0289 VMARK may only occur once per statement; keyword ignored

Error

0290 LABEL may only occur once per statement; keyword ignored

Error

0291 RECID may only occur once per statement; keyword ignored

Error

0292 DSPATR may only occur once per statement; keyword ignored

Error

0293 SHIFT may only occur once per statement; keyword ignored

Error

0294 PMT may only occur once per statement; keyword ignored

Error

0295 A literal may only occur once per statement

Error

0296 MARK may only occur once per statement; keyword ignored

Error

0297 AUXDUP may only occur once per statement; keyword ignored

Error

0298 EDTCDE may only occur once per statement; keyword ignored

Error

0299 INSERT may only occur once per statement; keyword ignored

Error

0300 INDEX may only occur once per statement; keyword ignored

Error

0301 LOGON may only occur once per statement; keyword ignored

Error

0302 SEQ may only occur once per statement; keyword ignored

Error

0303 LOGON valid only for a COMM file; keyword ignored

Error

0304 EDTCDE cannot be specified for this file type; ignored

Error

EDTCDE is not valid with magnetic stripe files. This error will also be listed if the DEVICE keyword contains an invalid device specification.

0305 CHECK once per statement when not conditioned

Error

If no indicator is specified in columns 9 and 10, the CHECK keyword can only be specified once per statement (including secondary lines). Only the first unconditioned CHECK keyword is used.

0306 CHECK once per statement when conditioned

Error

If the CHECK keyword is used more than once per statement (including secondary lines) only one use of the keyword can be specified with an indicator specified in columns 9 and 10. Only the first conditioned CHECK keyword is used.

0308 DEVICE keyword contains invalid parameter(s)

Severe

Valid parameters for the keyword DEVICE are: CRT, DISK, MREAD, PRINTER, COMM or COMM3270. The invalid parameter is ignored.

0310 DSPSIZ keyword contains invalid parameter(s)

Error

Valid parameters for the keyword DSPSIZ are:

Line	Characters
6	80 (480-character display)
12	80 (960-character display)
24	80 (1920-character display)

The DSPSIZ keyword is ignored.

0311 BLKING keyword contains invalid parameter(s)

Error

Valid parameters for the keyword BLKING are:

- *DBL (double-buffering)
- For format, *FMTU (unblocked and unspanned) or *FMTS (blocked and spanned, which is also the default)

*DBL must be specified first if more than one parameter is specified. *FMTU and *FMTS are valid only for a diskette file.

The BLKING keyword is ignored.

0312 NUMENT keyword contains invalid parameter(s)

Error

A valid parameter for the keyword NUMENT specifies the number of records in the data set and must be a whole number. The NUMENT keyword is ignored.

0313 FORM keyword contains invalid parameter(s)

Error

Valid parameters (in the listed sequence) for the keyword FORM are:

1. Length (required), a whole number representing the number of possible print lines in the form.
2. Overflow, the line number for the overflow line.
3. Indicator, the overflow indicator. The overflow line and the indicator are optional.

If the length keyword is in error, the FORM keyword is ignored; otherwise the indicator is ignored.

0314 SKIPA keyword contains invalid parameter(s)

Error

A valid parameter for the keyword SKIPA indicates the line number to which you want to skip. This parameter must be a whole number from 1 through the length of the form specified as the first FORM keyword parameter. The SKIPA keyword is ignored.

0315 SKIPB keyword contains invalid parameter(s)

Error

The valid parameter for the keyword SKIPB indicates the line number on which the current record is to be printed. This parameter must be a number from 1 through the length of the form specified as the first FORM parameter. The SKIPB keyword is ignored.

0316 SPACEA keyword contains invalid parameter(s)

Error

A valid parameter for the keyword SPACEA is a whole number from 0 through 3. The SPACEA keyword is ignored.

0317 SPACEB keyword contains invalid parameter(s)

Error

A valid parameter for the keyword SPACEB is a whole number from 0 through 3. The SPACEB keyword is ignored.

0318 LABEL keyword contains invalid parameter(s)

Error

The parameter for the keyword LABEL must be specified as it appears in the header label.

This parameter can be in one of two formats: VOL.NAME or NAME, in which VOL is the volume ID specified in the form *Vaaaaa, where * is the ID identifier, V is any letter A-Z, and aaaaa is a 5-character alphabetic or numeric string.

NAME can be a simple name (up to 8 characters long) or two simple names connected by periods. The first character must be alphabetic (A-Z) and the remaining characters can be alphabetic or numeric (0-9). The LABEL keyword is ignored.

0319 RECID keyword contains invalid parameter(s)

Error

The first parameter for the keyword RECID must be specified as *POSnnnn, where nnnn is the character position within the record. The second parameter must be specified as a single character enclosed in apostrophes. The RECID keyword is ignored.

0320 MARK keyword contains invalid parameter(s)

Error

A valid parameter for the keyword MARK must be specified as *POSnnnn where nnnn is a number representing the position to be marked. The MARK keyword is ignored.

0321 VMARK keyword contains invalid parameter(s)

Error

A valid parameter for the keyword VMARK must be specified as *POSnnnn where nnnn is a number representing the position to be marked. The VMARK keyword is ignored.

0322 DSPATR keyword contains invalid parameter(s)

Error

Valid parameters for the keyword DSPATR are: BL, CS, HI, ND, RI, CA, and UL. The invalid parameter is ignored.

0323 CHECK keyword contains invalid parameter(s)

Error

One or more of the following valid parameters must be specified with the keyword CHECK: DD, AD, AS, BC, BY, BV, DR, FE, ME, MF, RB, RL, RZ, LC, Mxx, or GXX. The invalid parameter is ignored.

0324 COMP keyword contains invalid parameter(s)

Error

The keyword COMP allows you to specify three parameters: TEST, DATA, and INDICATOR. The test parameter must be specified and must be one of the following: EQ, NE, LT, GT, LE, or GE. The data parameter must also be specified and can be the name of a defined field or a variable, a constant, or an arithmetic expression that involves addition, subtraction, multiplication, and division, constants and named fields or variables. The indicator parameter is optional and can be specified as a number 01 through 99. If the first two parameters are invalid the keyword is ignored; otherwise the third parameter is ignored.

0325 XCHK keyword contains invalid parameter(s)

Error

The keyword XCHK requires three parameters in the indicated sequence.

- Tablename must be the name of a table that contains numeric values
- Indexname1 and indexname2 must be the names of fields of 15 or fewer positions

The XCHK keyword is ignored.

0326 RANGE keyword contains invalid parameter(s)

Error

The keyword RANGE requires that you specify low and high limit parameter values. The parameters can e constants or field names. The RANGE keyword is ignored.

0327 RANGET keyword contains invalid parameter(s)

Error

The keyword RANGET requires a tablename that refers to a table containing pairs of entries that define the low and high limits of a range. The indexname parameter is optional, must follow the tablename, and must name a numeric field to receive the index from the lookup operation. If the first parameter is invalid, the keyword is ignored; otherwise the second parameter is ignored.

0328 LOOK keyword contains invalid parameter(s)

Error

The keyword LOOK requires that you specify a tablename parameter whose entries correspond to the data in the compare field. The index name parameter is optional, must follow the tablename, and must be a numeric field. If the first parameter is invalid, the keyword is ignored; otherwise the second parameter is ignored.

0329 SEQ keyword contains invalid parameter(s)

Error

The keyword SEQ requires that you specify a test parameter to be chosen from one of the following:

EQ, NE, GE, LE, GT, LT. The SEQ keyword is ignored.

0330 SHIFT keyword contains invalid parameter(s)

Error

The keyword SHIFT requires that you specify a codes parameter made up of as many characters as there are character positions in the field. The following characters are valid: A, D, H, N, W, V (World Trade only), X, and Y. The SHIFT keyword is ignored.

0331 PMT keyword contains invalid parameter(s)

Error

The keyword PMT requires that you specify a message parameter that is any message with a length of 200 character positions or less. The PMT keyword is ignored.

0332 ADD keyword contains invalid parameter(s)

Error

The keyword ADD requires that you specify a counter parameter that is the name of a numeric field. The ADD keyword is ignored.

0333 SUB keyword contains invalid parameter(s)

Error

The keyword SUB requires that you specify a counter parameter that is the name of a numeric field. The SUB keyword is ignored.

0334 AUXDUP keyword contains invalid parameter(s)

Error

The keyword AUXDUP requires that you specify a source parameter that is the name of a defined field. The AUXDUP keyword is ignored.

0335 AUXST keyword contains invalid parameter(s)

Error

The keyword AUXST requires that you specify a target parameter that is the name of a field. The AUXST keyword is ignored.

0336 EDTCDE keyword contains invalid parameter(s)

Error

The keyword EDTCDE requires a code parameter (1, 2, 3, 4, A, B, C, D, J, K, L, M, X, Y, or Z). The float parameter is optional and must be an asterisk (*) or a currency symbol enclosed in apostrophes. If the first parameter is invalid, the keyword is ignored; otherwise the second parameter is ignored.

0337 INSERT keyword contains invalid parameter(s)

Error

Valid parameters for the INSERT keyword are:

- SOURCE, the name of any named field (character or numeric data)
- EXPRESSION, an arithmetic expression with a maximum of 30 terms or 356 characters
- CONSTANT, fixed numeric or character data; numeric constants must have numeric fields as destinations and character constants must have character fields as destinations

The INSERT keyword is ignored.

0338 SETOF keyword contains invalid parameter(s)

Error

The keyword SETOF requires that you specify any two-digit number from 01 through 99. The SETOF keyword is ignored.

0339 SETON keyword contains invalid parameter(s)

Error

The keyword SETON requires that you specify any two-digit number from 01 through 99. The SETON keyword is ignored.

0340 RESET keyword contains invalid parameter(s)

Error

The keyword RESET requires that the parameter specify the name of a numeric field. The RESET keyword is ignored.

0341 SUBST keyword contains invalid parameter(s)

Error

The keyword SUBST requires that you specify two parameters: tablename1, which contains the entries for comparison to the data entered in the field and tablename2, which contains substitution entries for the defined field. (Tablename1 must be specified first.) A third parameter, indexname, is optional and if used is specified as the name of a numeric field. If the third parameter is invalid, it is ignored; otherwise the keyword is ignored.

0342 INDEX keyword contains invalid parameter(s)

Error

The keyword INDEX can contain either the storage parameter, which must be a whole number, and/or the file parameter, which must be the name of the index file on the diskette. The file name is specified the same way as it is. The parameter is ignored.

0343 TADD keyword contains invalid parameter(s)

Error

The keyword TADD requires that you specify the counter parameter as *TOTn, where n is a number from 1 through 9. The TADD keyword is ignored.

0344 TSUB keyword contains invalid parameter(s)

Error

The keyword TSUB requires that you specify the counter parameter as *TOTn, where n is a number from 1 through 9. The TSUB keyword is ignored.

0345 ERROR keyword contains invalid parameter(s)

Error

The keyword ERROR requires that you specify the code parameter as a digit number from 01 through 99. The message parameter is an optional character string, up to 39 character positions long, enclosed in parentheses. If the first parameter is invalid, the keyword is ignored; otherwise the second parameter is ignored.

0346 EXSR keyword contains invalid parameter(s)

Error

The keyword EXSR requires that you specify the name of a calculation routine as a parameter. The EXSR keyword is ignored.

0347 LOGON keyword contains invalid parameter(s)

Error

The keyword LOGON requires that you specify either a character constant enclosed in apostrophes or the name of a field that contains the LOGON information as a parameter. The LOGON keyword is ignored.

0349 If on a CRT file, SETON is only valid for a field

Error

The SETON keyword is valid only on a field description statement, not on a record description statement for a CRT file. The SETON keyword is ignored.

0350 If not on a CRT file, SETON is only valid for a record

Error

For files other than CRT files, the SETON keyword is valid only on a record description statement, not on a field description statement. The SETON keyword is used in conjunction with the RECID keyword. The SETON keyword is ignored.

0351 If on a CRT file, SETOF is only valid on a field

Error

The SETOF keyword is valid only on a field description statement, not on a record description statement for a CRT file. The SETOF keyword is ignored.

0352 If not on a CRT file, SETOF is only valid on a record

Error

For files other than CRT files, the SETOF keyword is valid only on a record description statement, not on a field description statement. The SETOF keyword is used in conjunction with the RECID keyword. The SETOF keyword is ignored.

0353 A literal is valid only for a field statement; ignored

Error

0366 RECID parameter exceeds record length; keyword ignored

Error

The *POSnnnn parameter specified a character position that exceeded the defined logical record length.

0367 MARK parameter exceeds record length; keyword ignored

Error

The *POSnnnn parameter specified a character position that exceeded the defined logical record length.

0368 VMARK parameter exceeds record length; keyword ignored

Error

The *POSnnnn parameter specified a character position that exceeded the defined logical record length.

0391 MARK keyword is valid only on the TFILE statement; ignored

Error

0392 VMARK keyword is valid only on the TFILE statement; ignored

Error

0401 CHECK keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0402 COMP keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0403 XCHK keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0404 RANGE keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0405 RANGET keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0406 LOOK keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0407 SEQ keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0408 DSPATR keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0409 SHIFT keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0410 ADD keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0411 SUB keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0412 AUXDUP keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0413 AUXST keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0414 INSERT keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0415 EDTCDE keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0416 ERROR keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0417 PMT keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0418 SUBST keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0419 TADD keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0420 TSUB keyword cannot be used with *RTN; keyword ignored

Error

The only keywords that apply to a *RTN field are ones such as SETON, SETOFF, RESET, and EXSR. These keywords require no interaction with data just entered. *RTN is not a data field.

0423 INSERT keyword is valid only for usage I, B, or W; ignored

Error

The keyword INSERT is only valid for use in fields specified as input, both, or workspace in column 38.

0424 RESET keyword is valid only for usage I, B, or W; ignored

Error

The keyword RESET is only valid for use in an input, both, or workspace field.

0425 SETON keyword is valid only for usage I, B, or W; ignored

Error

The keyword SETON is only valid for use in an input, both, or workspace field.

0426 SETOF keyword is valid only for usage I, B, or W; ignored

Error

The keyword SETOF is only valid for use in an input, both, or workspace field.

0427 EXSR keyword is valid for usage I, B, or W; ignored

Error

The keyword EXSR is only valid for use in an input field, both, or workspace field.

0430 ADD keyword is valid for usage I or B; keyword ignored

Error

The keyword ADD is only valid for use in an input field or both.

0431 CHECK keyword is valid for usage I or B; keyword ignored

Error

The keyword CHECK is only valid for use in an input field or both.

0432 COMP keyword is valid for usage I or B; keyword ignored

Error

The keyword COMP is only valid for use in an input field or both.

0433 ERROR keyword is valid for usage I or B; keyword ignored

Error

The keyword ERROR is only valid for use in an input field or both.

0434 LOOK keyword is valid for usage I or B; keyword ignored

Error

The keyword LOOK is only valid for use in an input field or both.

0435 PMT keyword is valid for usage I or B; keyword ignored

Error

The keyword PMT is only valid for use in an input field or both.

0436 RANGE keyword is valid for usage I or B; keyword ignored

Error

The keyword RANGE is only valid for use in an input field or both.

0437 RANGET keyword is valid for usage I or B; keyword ignored

Error

The keyword RANGET is only valid for use in an input field or both.

0438 SEQ keyword is valid for usage I or B; keyword ignored

Error

The keyword SEQ is only valid for use in an input field or both.

0439 SHIFT keyword is valid for usage I or B; keyword ignored

Error

The keyword SHIFT is only valid for use in an input field or both.

0440 SUB keyword is valid for usage I or B; keyword ignored

Error

The keyword SUB is only valid for use in an input field or both.

0441 SUBST keyword is valid for usage I or B; keyword ignored

Error

The keyword SUBST is only valid for use in an input field or both.

0442 TADD keyword is valid for usage I or B; keyword ignored

Error

The keyword TADD is only valid for use in an input field or both.

0443 TSUB keyword is valid for usage I or B; keyword ignored

Error

The keyword TSUB is only valid for use in an input field or both.

0444 XCHK keyword is valid for usage I or B; keyword ignored

Error

The keyword XCHK is only valid for use in an input field or both.

0445 AUXDUP keyword is valid only for usage I or B; keyword ignored

Error

The keyword AUXDUP is only valid for use in an input field or both.

0446 AUXST keyword is valid only for usage I or B; keyword ignored

Error

The keyword AUXST is only valid for use in an input field or both.

0447 EDTCDE was specified; the CRT record must be referenced by a WRITE; keyword ignored

Error

For a CRT, the keyword EDTCDE is only valid for use in a record written from C-specifications with a WRITE operation code. The EDTCDE keyword is ignored.

0449 DSPATR(ND) will result from this combination

Error

A combination of the RI, UL, and HI display attributes will result in nondisplay of the field.

0450 SHIFT data type must be C; keyword ignored

Error

The SHIFT keyword is required in field description statements in which the letter C is specified for the data type field. It is not valid if any other letter is specified in the data type field.

0451 DSPATR is valid only for usage I, O, or B; ignored

Error

The keyword DSPATR is valid only for use in fields specified as input, output, or both in column 38.

0452 A literal is valid only for usage O; ignored

Error

A literal is valid only for use in an output field.

0454 A literal cannot occur with a named field; ignored

Error

A literal is not a data field, it is a constant. It therefore cannot be referenced or changed, except by changing the source program.

0455 SHIFT must be specified; DATA TYPE of blank assumed

Error

A field with DATA TYPE C was specified, but the SHIFT keyword was not. The DATA TYPE is changed to blank.

0480 DEVICE keyword parameter(s) are missing; keyword ignored

Severe

The required parameter for the keyword DEVICE must be one of the following: CRT, DISK, MREAD, PRINTER, or COMM.

0501 DSPSIZ keyword parameter(s) are missing; keyword ignored

Error

Required parameters for the keyword DSPSIZ are:

Line	Characters
6	80 (480-character display)
12	80 (960-character display)
24	80 (1920-character display)

0502 FORM keyword parameter(s) are missing; keyword ignored

Error

The length parameter is required for the keyword FORM and must be a whole number representing the number of possible print lines on the form.

0503 RECID keyword parameter(s) are missing; keyword ignored

Error

Required parameters of the keyword RECID are: *POSnnnn (where nnnn is the character position within the record) and a single character enclosed in apostrophes.

0504 COMP keyword parameter(s) are missing; keyword ignored

Error

Required parameters for the keyword COMP are test, specified as EQ, NE, LT, GT, LE, or GE; and data, specified as the name of a defined field or variable, a constant, or an arithmetic expression.

0505 XCHK keyword parameter(s) are missing; keyword ignored

Error

The keyword XCHK requires three parameters in the following sequence:

- Tablename must be the name of the table that contains numeric values
- Indexname1 and indexname2 must be the names of numeric fields

0506 RANGE keyword parameter(s) are missing; keyword ignored

Error

Required parameters for the keyword RANGE are the low- and high-limit parameter values.

0507 EDTCDE keyword parameter(s) are missing; keyword ignored

Error

The required parameters for the keyword EDTCDE is a single character code chosen from one of the following: 1, 2, 3, 4, A, B, C, D, J, K, L, M, X, Y, or Z.

0508 INSERT keyword parameter(s) are missing; keyword ignored

Error

Required parameters for the keyword INSERT are: source, the name of a field; expression, an arithmetic expression having as many as 30 terms or a maximum length of 256 characters; and constant, a fixed character or numeric data.

0509 SUBST keyword parameter(s) are missing; keyword ignored

Error

Required parameters for the keyword SUBST are: tablename1 and tablename2.

0510 BLKING keyword parameter(s) are missing; keyword ignored

Error

One or two parameters can be specified for the BLKING keyword. The first, *DBL, specifies double buffering. The second, format, describes blocking and spanning characteristics. *FMTU specifies unblocked and unspanned; *FMTS specifies blocked and spanned.

0511 NUMENT keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword NUMENT is a whole number specifying the number of records in the data set.

0512 CHECK keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword CHECK must be one of the following: DD, AD, AS, BC, BY, BV, DR, FE, ME, MF, RB, RZ, Mxx, Gxx, or RL.

0513 DSPATR keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword DSPATR must be specified as one or more of the following: BL, CS, HI, ND, RI, CA, and UL.

0514 MARK keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword MARK must be specified as *POSnnnn where nnnn is a number representing the position to be marked.

0515 VMARK keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword VMARK must be specified as *POSnnnn where nnnn is a number representing the position to be marked.

0516 SHIFT keyword parameter(s) are missing; keyword ignored

Error

The required code parameter for the keyword SHIFT contains as many characters as there are character positions in the field. Valid characters are: A, D, H, N, W, V (World Trade), X, and Y.

0517 LABEL keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword LABEL must be specified as it appears on the header label.

0518 ADD keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword ADD must specify the name of a numeric field.

0519 SUB keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SUB must specify the name of a numeric field.

0520 AUXDUP keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword AUXDUP must specify a source parameter that is the name of a defined field.

0521 AUXST keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword AUXST is a target parameter that must specify the name of a defined field.

0522 RANGET keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword RANGET is the name of a table containing pairs of entries that define the low and high limits of a range.

0523 LOOK keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword LOOK is a tablename.

0524 SPACEA keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SPACEA is a whole number from 0 through 3.

0525 SKIPA keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SKIPA is a whole number from 1 through the length of the form. (The form length is specified as the first FORM keyword parameter.)

0526 SKIPB keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SKIPB specifies the line on which the current record is to be printed. The line number is a whole number from 1 through the length of the form. (The form length is specified as the first FORM parameter.)

0527 ERROR keyword parameter(s) are missing; keyword ignored

Error

The required code parameter for the keyword ERROR must specify a 2-digit number from 01 through 99.

0528 TADD keyword parameter(s) are missing; keyword ignored

Error

The required counter parameter for the keyword TADD must specify *TOTn, where n is a number from 1 through 9.

0529 TSUB keyword parameter(s) are missing; keyword ignored

Error

The required counter parameter for the keyword TSUB must specify *TOTn where n is a number from 1 through 9.

0530 RESET keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword RESET must specify a numeric field.

0531 SEQ keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SEQ must be specified as one of the following: EQ, NE, GE, LE, GT, or LT.

0532 INDEX keyword parameter(s) are missing; keyword ignored

Error

The required parameters for the keyword INDEX are: storage, (a whole number), or file, (the name of the index file on the diskette).

0533 EXSR keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword EXSR is the name of a calculation subroutine.

0534 SETOF keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SETOF is a two-digit number from 01 through 99.

0535 SETON keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SETON is a two-digit number from 01 through 99.

0536 PMT keyword parameter(s) are missing; keyword ignored

Error

The required message parameter for the keyword PMT is any message with a length of 200 character positions or less.

0537 SPACEB keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword SPACEB is a whole number from 0 through 3.

0538 LOGON keyword parameter(s) are missing; keyword ignored

Error

The required parameter for the keyword LOGON is: a character constant enclosed in apostrophes or the name of a field that contains the LOGON information as a parameter.

0550 BLKING keyword contains too many parameters; extra ignored

Error

You can specify one or two parameters for the BLKING keyword. The first, *DBL, specifies double buffering. The second, format, specifies unblocked and unspanned (*FMTU) or blocked and spanned (*FMTS).

0551 DSPSIZ keyword contains too many parameters; extra ignored

Error

The keyword DSPSIZ requires two parameters. The first is the number of lines, the second is the characters per line.

0552 FORM keyword contains too many parameters; extra ignored

Error

You may specify only three parameters for the keyword FORM: one whole number for length, one line number for overflow, and one number (01 through 99) for indicator.

0553 NUMENT keyword contains too many parameters; extra ignored

Error

You may specify only one parameter for the keyword NUMENT; it must be a whole number.

0554 SPACEA keyword contains too many parameters; extra ignored

Error

You may specify only one parameter for the keyword SPACEA; it must be a whole number from 0 through 3.

0555 SPACEB keyword contains too many parameters; extra ignored

Error

You may specify only one parameter for the keyword SPACEB; it must be a whole number from 0 through 3.

0556 RECID keyword contains too many parameters; extra ignored

Error

You may specify only two parameters for the keyword RECID: one in the form *POSnnnn and the second as a single character enclosed in apostrophes.

0557 MARK keyword contains too many parameters; extra ignored

Error

You may specify only one parameter for the keyword MARK; it must be in the form *POSn.

0558 VMARK keyword contains too many parameters; extra ignored

Error

You may specify only one parameter for the keyword VMARK; it must be in the form *POSnnnn.

0559 SHIFT keyword contains too many parameters; extra ignored

Error

You may only specify as many characters (in codes parameter) as there are character positions in the field. There must be no embedded blanks in the parameter.

0560 EDTCDE keyword contains too many parameters; extra ignored

Error

You may only specify two parameters for the keyword EDTCDE. The first parameter specifies the edit code; the second specifies an asterisk (*) or a currency symbol.

0561 LABEL keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword LABEL; it must duplicate the header label.

0562 ADD keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword ADD; it must be the name of a numeric field.

0563 SUB keyword contains too many parameters; extra ignored

Error

You may specify only one parameter for the keyword SUB; it must be the name of a numeric field.

0564 AUXDUP keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword AUXDUP; it must be the name of a defined field.

0565 AUXST keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword AUXST; it must be the name of a defined field.

0566 XCHK keyword contains too many parameters; extra ignored

Error

The keyword XCHK requires three parameters in the following sequence:

- Tablename must be the name of a table that contains numeric values
- Indexname1 and indexname2 must be the names of numeric fields

0567 RANGET keyword contains too many parameters; extra ignored

Error

You may only specify two parameters for the keyword RANGET; one for tablename and one for indexname.

0568 SUBST keyword contains too many parameters; extra ignored

Error

You may only specify three parameters for the keyword SUBST; the first must be tablename1, the second must be tablename2, and the third (optional) is indexname.

0569 LOOK keyword contains too many parameters; extra ignored

Error

You may only specify two parameters for the keyword LOOK; one must be tablename and the other indexname.

0570 COMP keyword contains too many parameters; extra ignored

Error

You may only specify three parameters for the keyword COMP: test, data, and indicator.

0571 INSERT keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword INSERT. The parameter can specify a field name, an arithmetic expression, or a constant.

0572 SKIPA keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword SKIPA: it must be a whole number from 1 through length of the form. (The form length is specified as the first FORM parameter.)

0573 SKIPB keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword SKIPB; it indicates the line number on which the current record is to be printed.

0574 ERROR keyword contains too many parameters; extra ignored

Error

You may only specify two parameters for the keyword ERROR; code (required), and message (optional).

0575 RANGE keyword contains too many parameters; extra ignored

Error

You may only specify two parameters for the keyword RANGE: low- and high-limit values.

0576 TADD keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword TADD; it must be in the *TOTn form.

0577 TSUB keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword TSUB; it must be in the *TOTn form.

0578 RESET keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword RESET; it must be a numeric field.

0579 SEQ keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword RESET; it must be a test parameter.

0580 EXSR keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword EXSR; it must be the name of a calculation routine.

0581 INDEX keyword contains too many parameters; extra ignored

Error

You may only specify two parameters for the keyword INDEX: storage and file.

0582 SETOF keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword SETOF; it must be a two-digit number from 01 through 99.

0583 SETON keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword SETON; it must be a two-digit number from 01 through 99.

0584 LOGON keyword contains too many parameters; extra ignored

Error

You may only specify one parameter for the keyword LOGON; either a character string enclosed in apostrophes or the name of a field containing the LOGON information.

0585 EDTCDE second parameter is invalid with codes X, Y, or Z

Error

The second parameter is ignored.

0596 A literal may not be specified for this device type

Error

A literal was specified for either a device of MREAD or an invalid device type; that is, the parameter on the DEVICE keyword was invalid. The literal is ignored.

0597 DEVICE must specify a device address when *NOPMT used

Error

When JOBOPT(*NOPMT) is specified in the job specification statement, the DEVICE keyword must specify a device address in all file description statements except the file description statements associated with compile-time tables. The DEVICE keyword in the file description statement for the CRT must not specify a device ID. No action is taken.

0601 First Z-specification not job specification; job specification considered missing

Severe

The first statement in a source program must be a Z-specification containing a job specification statement. The compilation is terminated.

0602 A misplaced job specification was found; it was ignored

Error

A job specification statement was found after the first Z-specification. Only one job specification is allowed per job.

0605 One or more invalid keywords on a job specification

Error

Only these keywords are valid for a job specification statement: JOBOPT, TFILE, CFILE, PRTFILE, DATE, EDITC, SHARE(R), STATUS, ENTRATR, and EXITATR. The invalid keyword is ignored.

0606 One or more invalid keywords on an entry specification

Error

Only these keywords are valid for an entry format specification: SLNO, CLRL, WRITE, and EOJ. The invalid keyword is ignored.

0607 One or more invalid keywords on a review specification

Error

Keywords were found on a review format statement. No keywords are valid; the keywords are ignored.

0610 Only keyword EOJ with calculations routine reference

Error

Only the EOJ keyword is valid on an entry format statement when columns 10 through 19 of the statement contain a reference to a BEGSR statement. The invalid keyword is ignored.

0612 JOBOPT had no parameters; must be *NOPMT or *NOOPEN

Error

The JOBOPT keyword is ignored.

0613 JOBOPT keyword contains invalid parameter(s)

Error

On a job specification, the keyword JOBOPT must contain the *NOPMT or *NOOPEN parameter. The invalid keyword is ignored.

0614 JOBOPT keyword contains too many parameters; ignored

Error

You may only specify two parameters for the keyword JOBOPT: *NOPMT and/or *NOOPEN.

0617 TFILE keyword contains no parameters

Error

The keyword TFILE must include the name parameter and may include a number parameter that specifies the number of records to be written between deleted records. The TFILE keyword is ignored.

0618 TFILE not a diskette data set

Severe

The name parameter for the TFILE keyword must be the name of a file that is described as a diskette data file in a file description statement. The TFILE keyword is ignored.

0619 Second parameter on TFILE not valid; assume 0

Error

The second parameter for the TFILE keyword must specify the number of data records to be written to the data set between deleted records.

0620 TFILE not specified when required

Severe

The keyword TFILE is required to specify the use of the transaction file.

0621 TFILE is only valid when accessed

Severe

The keyword TFILE cannot be specified unless entry lines reference keyboard/display records.

0623 TFILE keyword contains invalid parameter(s)

Error

Valid parameters for the keyword TFILE are: name (the name of a described diskette file) and number (the number of records written to the file between the insertion of deleted records). The extra parameter(s) are ignored.

0624 CFILE keyword contains invalid parameter(s)

Error

The valid parameter for the keyword CFILE is the name of the file from which data is to be copied. The CFILE keyword is ignored.

0625 PRTFILE keyword contains invalid parameter(s)

Error

The only valid parameter for the keyword PRTFILE is name, which specifies the file to be printed. The PRTFILE keyword is ignored.

0626 DATE keyword contains invalid parameter(s)

Error

The valid parameters for the keyword DATE are either *DMY, *YMD, or *MDY. The DATE keyword is ignored.

0627 EDITC keyword contains invalid parameter(s)

Error

The valid parameter for the keyword EDITC must specify five characters in the following order:

1 and 2 – currency symbol to be used
3 – decimal point character
4 – thousands separator character
5 – date separator

The parameter must be enclosed in apostrophes. The EDITC keyword is ignored.

0628 STATUS keyword contains invalid parameter(s)

Error

The valid parameter for the keyword STATUS is name, which identifies a numeric field with a length of 4 and no decimal positions. The STATUS keyword is ignored.

0629 ENTRATR keyword contains invalid parameter(s)

Error

Valid parameters for the keyword ENTRATR are: BL, CS, HI, ND, RI, and UL. The ENTRATR keyword is ignored.

0630 SLNO keyword contains invalid parameter(s)

Error

The valid parameter for the keyword SLNO is any number that specifies the starting line number; that is, a whole number greater than or equal to 2 (or greater than or equal to 3 if you plan to display prompts by using the PMT keyword) and not greater than the number of lines in the display. The SLNO keyword is ignored.

0631 CLRL keyword contains invalid parameter(s)

Error

Valid parameters for the keyword CLRL are number or *NO. The CLRL keyword is ignored.

0632 WRITE keyword contains invalid parameter(s)

Error

The valid parameter for the keyword WRITE is either a name or *NO. The WRITE keyword is ignored.

0636 TFILE keyword contains too many parameters

Error

You may only specify name and number as parameters. The extra parameter(s) are ignored.

0637 CFILE keyword contains too many parameters

Error

You may only specify one parameter for the keyword CFILE, that being the name of the file to be copied. The CFILE keyword is ignored.

0638 PRTFILE keyword contains too many parameters

Error

You may only specify the name parameter for the keyword PRTFILE. The PRTFILE keyword is ignored.

0639 EDITC keyword contains too many parameters

Error

You may only specify one parameter for the keyword EDITC.

0640 STATUS keyword contains too many parameters

Error

You may only specify one parameter for the keyword STATUS: name. The STATUS keyword is ignored.

0641 SLNO keyword contains too many parameters

Error

You may only specify one starting line number for the keyword SLNO. The SLNO keyword is ignored.

0642 CLRL keyword contains too many parameters

Error

You may only specify one parameter for the keyword CLRL: either number or *NO. The CLRL keyword is ignored.

0643 WRITE keyword contains too many parameters

Error

You may only specify one parameter for the keyword WRITE: name or *NO. The extra parameter(s) are ignored.

0644 EOJ keyword contains too many parameters

Error

You may specify a maximum of three parameters for the keyword EOJ: name, device, and *PASS. The extra parameter(s) are ignored.

0645 DATE keyword contains too many parameters

Error

You may only specify one parameter for the keyword DATE; it must be *DMY, *YMD, or *MDY. The extra parameter(s) are ignored.

0651 CFILE keyword contains no parameters; keyword ignored

Error

The keyword CFILE requires that you specify the name of the file from which data is to be copied.

0652 PRTFILE keyword contains no parameters; keyword ignored

Error

The required parameter for the keyword PRTFILE is the name of the file to be printed.

0653 EDITC keyword contains no parameters; keyword ignored

Error

The keyword EDITC must specify a five-character parameter, enclosed in apostrophes. The characters must be specified in the following order:

1 and 2 – currency symbol to be used
3 – decimal point character
4 – thousands separator character
5 – data separator

0654 SHARE(R) keyword contains no parameters; keyword ignored

Error

The keyword SHARE(R) requires that you specify a name parameter.

0655 STATUS keyword contains no parameters; keyword ignored

Error

You must specify the name parameter for the keyword STATUS.

0656 ENTRATR keyword contains no parameters; keyword ignored

Error

The keyword ENTRATR requires that you specify one or more of the following attributes: BL, CS, HI, ND, RI, or UL.

0657 CLRL keyword contains no parameters; keyword ignored

Error

The keyword CLRL requires that you specify either the number of lines to be cleared or the *NO parameter to retain the current display.

0658 WRITE keyword contains no parameters; keyword ignored

Error

The keyword WRITE requires that you specify either the name or the *NO parameter.

0660 CFILE invalid; TFILE not specified or invalid

Error

The keyword TFILE was not specified or was invalid when the keyword CFILE was specified. The CFILE keyword is ignored.

0661 CFILE device type must be DISK; keyword ignored

Error

0662 CFILE invalid; specified same file as TFILE; ignored

Error

The name parameter for the keyword CFILE must be different than that for the keyword TFILE, because the transaction file is copying data from a distinct copy file.

0663 PRTFILE invalid; TFILE not specified or invalid

Error

If the keyword PRTFILE is used, a valid TFILE keyword must be specified. The keyword is ignored.

0664 PRTFILE device type must be printer; keyword ignored

Error

The PRTFILE keyword must specify a file name that is defined on a file description statement that identifies a printer.

0665 PRTFILE invalid; specifies same file as TFILE; ignored

Error

The PRTFILE keyword cannot specify the same file name as the TFILE keyword.

0666 PRTFILE invalid; specified same file as CFILE; ignored

Error

The PRTFILE keyword cannot specify the same file name as the CFILE keyword.

0667 DATE contained no parameters; MMDDYY assumed

Error

The DATE keyword requires a parameter of *DMY (DDMMYY), *YMD (YYMMDD), or *MDY (MMDDYY). *MDY is the default.

0668 Duplicated parameter in keyword SHARE(R)

Error

The duplicated parameter is ignored.

0670 JOBOPT keyword was a duplication and was ignored

Error

0671 TFILE keyword was a duplication and was ignored.

Error

0672 CFILE keyword was a duplication and was ignored

Error

0673 PRTFILE keyword was a duplication and was ignored

Error

0674 DATE keyword was a duplication and was ignored

Error

0675 EDITC keyword was a duplication and was ignored

Error

0676 STATUS keyword was a duplication and was ignored

Error

0677 ENTRATR keyword was a duplication and was ignored

Error

0678 EXITATR keyword was a duplication and was ignored

Error

0679 SLNO keyword was a duplication and was ignored

Error

0680 CLRL keyword was a duplication and was ignored

Error

0681 WRITE keyword was a duplication and was ignored

Error

0682 EOJ keyword was a duplication and was ignored

Error

0684 Keyword SHARE(R) contains more than 15 parameters

Error

The extra parameter(s) are ignored.

0685 Keyword SHARE(R) contains an invalid name parameter

Error

The name parameters for the keyword SHARE(R) must be file names. The invalid parameter(s) are ignored. If all parameters are invalid, the keyword is ignored.

0688 ENTRATR(ND) will result from this combination

Error

When you specify the HI, RI, and UL attributes for the keyword ENTRATR, the results will be equivalent to ND (nondisplay).

0689 Keyword EXITATR valid parameters; BL, CS, HI, ND, RI, and UL

Error

An invalid parameter was specified for the EXITATR keyword. The valid parameters are BL, CS, HI, ND, RI, and UL. The invalid parameters are ignored. If all parameters are invalid, the keyword is ignored.

0690 Keyword EXITATR has no parameters, may or may not be ignored

Error

The EXITATR keyword contains no parameters on a job specification. If the ENTRATR keyword is specified, a default of normal is assumed for EXITATR. Otherwise, EXITATR is ignored. The keyword is ignored.

0691 EXITATR(ND) will result from this combination

Error

When you specify the HI, RI, and UL parameters for the keyword EXITATR, the result will be equivalent to ND (nondisplay).

0693 Keyword EXITATR specified without ENTRATR; ignored

Error

The use of the EXITATR keyword without the ENTRATR keyword is invalid.

0694 Keyword SLNO contains no parameters; default used

Error

The number parameter of the SLNO keyword specifies the uppermost physical display line that the format can use. The default number is 2 (line 1 is always reserved for the status line).

0706 Keyword WRITE parameter is not a record in the TFILE

Error

The name parameter for the keyword WRITE must name a record that is described in a record description statement for the transaction file. The keyword is ignored.

0724 EOJ first parameter is invalid

Error

When the keyword EOJ is specified with the name and device parameters, the name parameter must name the data set as it appears on the diskette. This name can be either a constant name (name parameter enclosed in apostrophes) or a variable name (name parameter not enclosed in apostrophes). In addition, if the first parameter is *PASS, it must be the only parameter specified with the EOJ keyword.

0725 Second parameter of EOJ invalid device address

Error

All parameters are ignored.

0726 Third parameter of EOJ not *PASS

Error

If three parameters are specified for the EOJ keyword, the third parameter must be *PASS. This parameter is ignored.

0727 First parameter on EOJ invalid; cannot use second parameter

Error

The valid first parameter for the keyword EOJ is the name of the data set as it appears on the diskette. All parameters are ignored.

0728 Second parameter missing on EOJ, all parameters ignored

Error

The keyword EOJ requires that you also specify the device address parameter if the name parameter is used.

0730 *PASS is first parameter on EOJ, all other parameters ignored

Error

If the *PASS parameter is used with the keyword EOJ, it must follow the name and device parameters or be the only parameter specified.

0740 Entry specification out of order; statement ignored

Severe

The entry mode Z-specification test sequence line was ignored for one of the following reasons:

- It follows a job line
- It follows an entry line with a BEGSR label in the name field (columns 10 through 19)
- The data source field (columns 23 through 30) is blank or invalid

0742 Columns 23 through 30 and 45 through 46 are blank or invalid with review specification; statement was ignored

Error

The review mode Z-specification was ignored because the position to test field (columns 23 through 30) and for the next format ID field (columns 45 through 46) are both blank or invalid. The position to test field requires entry of the position to be tested using this format: *POSnnnn where nnnn is a whole number from 51 through 8192 representing the position in the record. The next format ID field requires entry of C where C is a character or number.

0743 Columns 23 through 30 and/or 45 through 46 invalid with column 22 in review specification; statement was ignored

Error

The review mode Z-specification was ignored because the field (column 22) is specified and the position to test field (columns 23 through 30) and the next format ID field (columns 45 and 46) are both blank or invalid. The position to test field requires entry of the position to be tested using this format: *POSnnnn where nnnn is a whole number from 1 through 8192 representing the position in the record. The next format ID field requires entry of C where C is a character or number.

0744 Column 22 invalid because of sequence of review specification; statement was ignored

Error

The review mode Z-specification was ignored because the and field (column 22) is specified and the preceding line was a job specification statement or an invalid review specification.

0745 Columns 45 and 46 (next format ID) blank or invalid in review specification; statement was ignored

Error

The review mode Z-specification was ignored because the next format ID field (columns 45 and 46) are blank or invalid and the next statement is not a review specification with the and field (column 22) specified.

0746 Unconditional branch encountered earlier on a review specification; statement was ignored

Error

The review mode Z-specification was ignored because an unconditional branch review specification is specified that cannot be referenced.

0747 Invalid entry specification referred to from review specification; statement was ignored

Error

The review mode Z-specification was ignored because the format ID to which it refers is an entry mode Z-specification that specifies the keyword WRITE(*NO).

0750 ID in columns 8 and 9 of an entry specification is a duplicate

Error

Each format ID used in a program must be unique. Valid entries are 1 through 9 and A0 through Z9. The ID is ignored.

0751 ID in columns 8 and 9 of an entry specification cannot be 0

Error

Valid format ID entries are 1 through 9 and A0 through Z9. The ID is ignored.

0752 ID in columns 8 and 9 is blank/invalid; columns 10 through 19 present

Severe

The ID field and the name field should both be either absent or present. No action is taken.

0755 Name in columns 10 through 17 not found on CRT record name or BEGSR

Error

The name in columns 10 through 17 of an entry mode Z-specification must be either a CRT record name or a BEGSR label. The name is ignored.

0756 Columns 10 through 17 blank or invalid when ID columns 8 and 9 present

Severe

The ID field and the name field must both be either absent or present. No action is taken.

0758 Column 20 (repeat field) must be blank if 8 through 17 blank

Error

On an entry mode Z-specification, the repeat field (column 20) must be blank when the ID field (columns 8 and 9) and the name field (columns 10 through 19) are both blank. The repeat field is ignored.

0759 Column 20 must be 1 if BEGSR label referenced; 1 assumed

Error

If column 20 is blank, no action is taken. If it is not blank, 1 is assumed.

0760 Column 22 must be blank for entry specification

Error

The and field is ignored.

0763 nnnn of *POSnnnn invalid; columns 23 through 30 ignored

Error

The position to be tested specified in columns 23 through 30 of an entry or review mode Z-specification must be specified in the form *POSnnnn, where nnnn is within the range of 1 and the record length of the transaction file (TFILE).

0764 Test condition field must be blank if BEGSR label in 10 through 17

Error

No action is taken.

0767 Test data in columns 33 and 34 ignored if columns 23 through 30 are blank

Error

The position to be tested field (columns 23 through 30) must be specified if a test condition (columns 33 and 34) is specified. No action is taken.

0770 Columns 35 through 37 (test data) blank; columns 23 through 30 ignored

Severe

If the character to be tested for field (columns 35 through 37) is blank, the position to be tested field (columns 23 through 30) must be blank.

0771 Test data in columns 35 through 37 ignored if columns 23 through 30 not present

Error

The position to be tested field (columns 23 through 30) must be specified if a character to test for (columns 35 through 37) is specified. No action is taken.

0775 Columns 45 and 46 (next format ID) is not defined in columns 8 and 9

Error

The ID specified in columns 45 and 46 must be defined in columns 8 and 9 of an entry format statement. On an entry line, the next format ID is ignored. On a review line, the specification is ignored.

0776 Columns 45 and 46 and EOJ are blank when columns 8 through 17 are blank

Error

On an entry mode Z-specification, either the next format ID field (columns 45 through 54) or the keyword EOJ must be specified when the ID field (columns 8 and 9) and the name field (columns 10 through 19) are both blank. No action is taken.

0777 Columns 45 and 46 and EOJ are blank when columns 23 through 30 are present

Error

Either the next format ID field (columns 45 through 54) or the EOJ keyword must be specified when the position to be tested field (columns 23 through 30) is specified. No action is taken.

0778 Columns 45 and 46 (next format ID) present; EOJ ignored

Error

When the EOJ keyword is specified, no ID can be specified as the next format ID.

0779 Columns 45 and 46 and EOJ are blank when BEGSR label in 10 through 17

Error

Either the next format ID field (columns 45 through 54) or the EOJ keyword must be specified when the name field (columns 10 through 19) specifies the label of a BEGSR statement. No action is taken.

0782 Columns 55 through 80 blank if columns 8 through 17 blank and columns 45 and 46 present

Error

The options field (columns 55 through 80) on an entry mode Z-specification must be blank when the ID and name fields (columns 8 through 17) are blank and the next format ID field (columns 45 and 46) is present. The keyword is ignored.

0783 Entry in columns 55 through 80 invalid; only keyword EOJ allowed

Error

The options field (columns 55 through 80) on an entry mode Z-specification must contain only a valid EOJ keyword when the ID field (columns 8 and 9) and the name field (columns 10 through 19) are both blank. No other keywords are allowed. The invalid keywords are ignored.

0788 TFILE record, fields not one-for-one with CRT record

Severe

The fields from the CRT file record in the name field must match one-for-one with the fields from the transaction file record for the keyword WRITE. (All other uses of a mismatched pair of records will also be in error.)

0800 With SKIPA keyword, the form length is exceeded; keyword ignored

Error

0801 With SKIPB keyword, the form length is exceeded; keyword ignored

Error

0802 BLKING (*FMTS/*FMTU) only valid on a diskette file

Error

The keyword is ignored.

0804 File statement must specify a length

Severe

An entry is required in the length field (columns 30 through 34) of a file description statement to specify the length of the logical records in the file. No action is taken, but the program will not compile. The error may cause other errors.

0805 File length is 0 or greater than 8192

Severe

Valid entries for the length field (columns 30 through 34) are 1 through 8192. The length is ignored. See error 0804.

0806 Data type has no meaning; ignored

Error

Data type (column 35) is not a valid entry on a file description statement.

0807 Decimal positions has no meaning; ignored

Error

Decimal positions (column 37) is not a valid entry on a file description statement.

0808 Position has no meaning; ignored

Error

The position field (columns 39 through 44) is not a valid entry on a field description statement.

0809 File length must be 80 or less; 80 assumed

Error

For compile-time table files only. A compile-time table is limited to 80-character records because the data is in the source.

0810 Table length invalid or greater than file length

Severe

You must specify a length to describe the length of a table entry. This length field must be numeric and must be right-adjusted (either leading blanks or leading zeros are allowed). A numeric table entry cannot exceed 15 positions. Character entries cannot exceed 256 positions. The length is ignored.

0811 Decimal position was specified, length > 15; decimal-position ignored

Error

The table will be alphabetic.

0812 Decimal-position with no length or > length; 0 assumed

Error

An entry was specified for decimal positions (column 37) and no length was specified (columns 30 through 34) or the position specified is greater than the length. A decimal positions entry of 0 is assumed, but this assumption is valid only if a length was specified in the length field.

0813 Data type has no meaning; ignored

Error

Data type (column 35) is not a valid entry on a table description statement that defines a table.

0814 Usage has no meaning; ignored

Error

The usage entry (column 38) is not a valid entry on a table description statement that defines a table.

0815 Position has no meaning; ignored

Error

The position field (columns 39 through 44) is not a valid entry on a table description statement that defines a table.

0816 A name must be specified with usage B; usage I assumed

Error

If usage B (both) is specified, a name must also be specified.

0817 Record statement has no name; no action taken

Error

A record must have a name so that the record can be referenced. This will cause errors to occur later in the compile.

0818 Length has no meaning; ignored

Error

The length field cannot be coded for record description statements unless they are on a COMM file.

0819 Data type has no meaning; ignored

Error

The data type entry cannot be specified on a record description statement.

0820 Decimal position has no meaning; ignored

Error

The decimal positions entry cannot be specified on a record description statement.

0821 Position has no meaning; ignored

Error

The position entry cannot be specified on a record description statement.

0822 Length is invalid

Severe

The length field must be 1 through 256 for a field/key. The length is ignored.

0823 Decimal-position was specified, length > 15; decimal-position ignored

Error

The field will be alphabetic.

0824 Decimal-position with no length or > length; 0 assumed

Error

A decimal positions entry is invalid if no length is specified. The decimal positions entry cannot be greater than 9 or the length specified. Because 0 is assumed, the field is still considered numeric.

0825 Working storage field must be a named field

Error

The specification is ignored.

0826 Length is invalid; ignored

Error

The length field is invalid on a COMM or COMM3270 file record.

0827 *RTN is only valid on a CRT file; *RTN is ignored

Error

*RTN is only valid for a field in a record that is associated with a CRT file.

0828 *RTN is only valid on a field; *RTN is ignored

Error

*RTN is only valid on a field description statement.

0830 Length cannot be used with *RTN; ignored

Error

0831 Data type cannot be used with *RTN; ignored

Error

0832 Decimal position cannot be used with *RTN; ignored

Error

0833 Usage cannot be used with *RTN; ignored

Error

0834 Position cannot be used with *RTN; ignored

Error

0835 Usage field has no meaning; usage ignored

Error

The usage field only has meaning on a CRT file, except for working-storage (W). However, usage is allowed (no message appears) on a diskette and COMM file. This message appears when:

1. Usage is specified on a printer file, and is other than O.
2. Usage is other than W on a compile-time table file.

This message is informational only and will not affect execution.

0836 A file statement must have a name

Severe

The program will not compile. This error may cause other errors.

0837 Name begins with TAB, or no length or name for table

Severe

A table description statement that defines a table must have a name and length specified. A table name cannot begin with TAB. The table is ignored.

0838 File length is greater than 1923; 1923 is assumed

Error

The record length of a COMM3270 file must not exceed the screen size of that terminal.

0839 Data type has no meaning with usage O or W; ignored

Error

You may only specify a data type with usage field entries I or B.

0840 Conditioning may only be on a field; indicator ignored

Error

An indicator entry (columns 9 and 10) is valid only in field description statements.

0841 Condition must have CHECK or ERROR keyword specified

Error

When an indicator entry (columns 9 and 10) is specified, either the CHECK or ERROR keyword must also be specified for the field description statement. The indicator is ignored.

0842 Position is invalid; ignored

Error

For a field for a non-CRT device, the position can be a number from 1 through 8192. For a CRT device, the first three numbers in the position field specify the line; the second three numbers specify the position in the line.

0846 CHECK(AD) and SUBST are incompatible; AD ignored

Error

CHECK(AD) specified with SUBST produces unpredictable results.

0849 INSERT parameter is longer than the field

Error

The INSERT keyword allows data to be supplied for the field being defined. The data (the INSERT parameter) should not be longer than the field. If the field is numeric, it is decimal-aligned. Extra data characters are ignored.

0850 CHECK(BY) and data type are incompatible; BY ignored

Error

0851 CHECK(BY/BV) only allowed; parameter(s) ignored

Error

A secondary line was specified with an indicator in columns 9 and 10 but the CHECK keyword on the same line contains parameter(s) other than BY or BV.

0852 CHECK(BY/BV) must appear alone when conditioned

Error

When an indicator is specified in columns 9 and 10 of a secondary line, CHECK (BV) must be the only other entry on the secondary line. All other keyword(s) and parameter(s) are ignored.

0853 CHECK(RL) incompatible with data type; RL is ignored

Error

CHECK (RL) is not valid if V (right half only) or W (right half shift) is not specified in the data type field.

0854 CHECK(RL) incompatible with RB and/or RZ; RB/RZ ignored

Error

Right to left fill (RL) is invalid with right adjust with blank field (RB) or right adjust with zero fill (RZ).

0855 CHECK(MF) and data type are incompatible; MF is ignored

Error

CHECK (MF) is not valid if S (signed numeric) is specified in the data type field.

0856 CHECK(MF) incompatible with RB and/or RZ; RB/RZ ignored

Error

Mandatory fill (MF) is invalid with right adjust with blank fill (RB) or right adjust with zero fill (RZ).

0857 CHECK(RB/RZ) are incompatible; RZ is ignored

Error

Right adjust with blank fill (RB) and right adjust with zero fill (RZ) cannot both be specified for the same field.

0858 CHECK(RB/RZ) invalid with data type C; RB/RZ ignored

Error

Right adjust with blank fill (RB) and right adjust with zero fill (RZ) are not valid if character check (C) is specified as the data type for the field.

0859 CHECK(AS) incompatible with AUXDUP keyword; AS ignored

Error

Automatic skip (AS) is an invalid CHECK parameter when the AUXDUP keyword is specified.

0860 CHECK(AS/AD) incompatible; AD is ignored

Error

Automatic skip (AS) and automatic duplication cannot both be specified for the same field.

0861 CHECK may only contain one Mxx/GXX type parameter

Error

Self check (Mxx) and self check generate (GXX) cannot both be specified for the same field. Only one of these parameter can be specified. All extra parameters are ignored.

0862 A literal is invalid on the TFILE

Severe

A literal is invalid for the transaction file. The literal is ignored.

0863 Data type incompatible with the length; N assumed

Error

Data type S is not compatible with a length of 1. Data type N is assumed.

0864 A literal is invalid with a length; length is ignored

Error

A literal is invalid if a length is specified in the length field (columns 30 through 34).

0865 CHECK (RB/RZ) incompatible with length 1; RB/RZ ignored

Error

On a field of length 1, right adjust cannot be specified.

0871 ERROR must be only keyword on this line; line ignored

Error

When the ERROR keyword is conditioned by an indicator in columns 9 and 10, it must be the only keyword specified on the secondary line.

0877 output field must have a name or a literal

Severe

Data must be present for an output field. The field is ignored.

0878 Invalid keyword(s) specified with INSERT

Error

The following keywords are valid with INSERT: DSPATR, PMT, RESET, SETOP, SETON, TADD, TSUB, ADD, SUB, AUXST, and EXSR. All invalid keywords are ignored.

0880 TADD may not occur with SUBST; TADD is ignored

Error

If the SUBST keyword is specified for a field, the TADD keyword is invalid for the field.

0881 TSUB may not occur with SUBST; TSUB is ignored

Error

If the SUBST keyword is specified for a field, the TSUB keyword is invalid for the field.

0887 Conditioning valid only on a secondary field line

Error

An indicator cannot be specified on a primary line or a continuation line. The indicator is ignored.

0890 Position has no meaning; ignored

Error

A workspace cannot be displayed, read from, or written to a file. Therefore, position has no meaning.

0900 64K of object storage exceeded during assignment

Severe

The program is too large. It must be recorded or divided into more than one program.

0910 Length and decimal-position must match earlier field definition

Error

A named variable (or field) is defined more than once. The attributes of the variable must always be defined the same way. The earlier definition is used.

0911 Invalid combination of implicit field definitions

Error

An undefined field name in a LOOK, RANGET, SUBST, or AUXST keyword has caused an implicit definition of the field. Implicit definitions must match in length and decimal positions. AUXST assumes the definition of the field for which the keyword is specified or, if none is present, the definition of the unnamed field. Because the current definition does not match the earlier implicit definition, the keyword is ignored.

0912 Table name duplicates a field name; table ignored

Severe

A table cannot have the same name as a field.

0913 Implicit field requires length or defined field name

Error

The AUXST keyword contains an undefined field name. The field is implicitly defined using the definition of the field for which the AUXST keyword is specified. A definition should be specified for the named field.

0920 File name duplicates a file or record name; ignored

Severe

A file cannot have the same name as a record or another file. Any records or tables specified for this file are assigned to the previous file, if present.

0921 Record name duplicates a file or record name; ignored

Severe

A record cannot have the same name as a file or another record.

0923 Maximum number of file definitions exceeded

Severe

For compile-time table files, at most 127 are allowed. for other files, the maximum number of non-keyboard/display files must not exceed 14. Single volume and offline multivolume files each count as 1 file. Online multivolume files count as the number of volumes specified in the DEVICE keyword. The file definition is ignored.

0924 Record is invalid in a file containing a table; ignored

Severe

A record description cannot follow a table description statement (T in column 17).

0925 Field must follow a valid record or field statement

Severe

A field record description statement must be preceded by a valid record description or another field description statement. The field definition is ignored.

0926 Field is invalid in a file containing a table; ignored

Severe

A field description statement cannot follow a table description statement (T in column 17).

0927 Table is invalid in a file containing a record; ignored

Severe

A table description statement cannot follow a record description statement.

0928 Key invalid with this device type; key changed to field

Error

Key fields are valid only if the DEVICE keyword on the associated file description statement specifies DISK.

0929 Table invalid with this device type; table ignored

Severe

The file description statement with which a table description statement is associated must specify DISK for the DEVICE keyword, or the DEVICE keyword must be omitted.

0930 TFILE, CFILE, PRTFILE cannot be multivolume

Severe

The file specified for the TFILE, CFILE, or PRTFILE keyword cannot have more than one address specified for the DEVICE keyword on the file description statement. The file definition is ignored. Any records or tables for this file are assigned to the previous file, if present.

0931 DEVICE specified more than 8 volumes for this file

Error

0933 SHARE file not defined

Error

You must specify a file name for the SHARE parameter. The undefined file name is ignored.

0934 SHARER file not defined

Error

You must specify a file name for the SHARER parameter. The undefined file name is ignored.

0935 Device specified cannot support SHARE file

Error

You must specify a file name for the SHARE parameter. The specified file is not a SHARE file.

0936 Device specified cannot support SHARER file

Error

You must specify a file for the SHARER parameter. The specified file is not a SHARER file.

0937 SHARE and SHARER cannot specify same file; SHARER used

Error

SHARER is more restrictive than SHARE, and the two cannot be specified for the same file.

0938 BEGSR label name duplicates a record name; ignored

Severe

The name of a subroutine cannot be the same as the name of a record defined on a record description statement.

0945 Duplicate table name found; ignored

Severe

Each table in a program must have a unique name.

0946 NUMENT must be specified on a table file

Severe

The NUMENT keyword must be specified on the file description statement if the file contains a table. The NUMENT keyword specifies the number of entries in the table. The table definition is ignored.

0947 Maximum number of tables exceeded

Severe

The maximum number of tables allowed is 128. The table definition is ignored.

0948 Maximum number of tables exceeded by key file index tables

Severe

The maximum number of tables allowed is 128. No second level index is defined.

0955 Duplicate label definition; label is ignored

Severe

The labels used in factor 1 of BEGSR, ENDSR, and TAG operations must be unique.

0958 Z-specification name is undefined or an invalid type; name ignored

Severe

The name must be a defined CRT record or subroutine name.

0959 A-specification name is undefined or an invalid type; name ignored

Severe

A field must be previously defined or must have length (and decimal position) specified.

0961 Factor 1 contains an undefined name or invalid type; name ignored

Error

The name must be previously defined and its type must be allowed with the specified operation. This statement is ignored.

0962 Subscript 1 contains an undefined name or invalid type; name ignored

Error

The name specified must be a defined field. This statement is ignored.

0963 Factor 2 contains an undefined name or invalid type; name ignored

Error

The name must be previously defined and its type must be allowed with the specified operation. This statement is ignored.

0964 Subscript 2 contains an undefined name or invalid type; name ignored

Error

The specified name must be a defined field. This statement is ignored.

0965 Result field contains an undefined name or invalid type; name ignored

Error

The name must be previously defined and its type must be allowed with the specified operation, or the name must have length (and decimal position) specified. Statement is ignored.

0966 Result position contains an undefined name or invalid type; name ignored

Error

The name specified must be a defined field. Statement is ignored.

0967 CTDATA or FTRANS file name is an undefined name or invalid type; name ignored

Error

The specified name must be a defined file.

0969 CFILE contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined file.

0970 PRTFILE contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined file.

0971 TFILE contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined file.

0972 WRITE contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined record.

0973 ADD contains an undefined name or invalid type; keyword ignored

Error

The specified named must be a defined field.

0974 SUB contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined field.

0975 AUXDUP contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined field.

0976 AUXST contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined field, or the field on which the AUXST keyword is used must be defined.

0977 COMP contains an undefined name or invalid type; keyword ignored

Error

The specified name(s) must be (a) defined field(s).

0978 INSERT contains an undefined name or invalid type; keyword ignored

Error

The specified name(s) must be (a) defined field(s).

0979 LOGON contains an undefined name or invalid type; keyword ignored

Error

The specified name must be a defined field.

0980 LOOK contains an undefined name or invalid type; keyword ignored

Error

The first name specified must be a defined table. The second name, if specified, must be a defined field.

0981 RANGE contains an undefined name or invalid type; keyword ignored

Error

The specified name(s) must be (a) defined field(s).

0982 RANGET contains an undefined name or invalid type; keyword ignored

Error

The first name specified must be a defined table. The second name, if specified, must be a defined field.

0983 SUBST contains an undefined name or invalid type; keyword ignored

Error

The first two names specified must be defined tables. The third name, if specified, must be a defined field.

0984 XCHK contains an undefined name or invalid type; keyword ignored

Error

The first name specified must be a defined table. The second two names must be defined fields.

0985 EXSR contains an undefined name or invalid type; keyword ignored

Error

The name specified must be a defined subroutine.

0986 EOJ contains an undefined name or invalid type; keyword ignored

Error

The program name parameter must be a defined field or a constant.

1000 ADD must be on a numeric field; ADD is ignored

Error

The ADD keyword is valid only on a field description statement for a numeric field.

1001 ADD parameter not numeric; ADD is ignored

Error

The ADD parameter must specify the name of a field that is defined as a numeric field.

1002 ADD field longer than the parameter

Error

The required parameter for the keyword ADD is a counter, the name of a numeric field. In this case, the data in the current numeric field is longer than the counter. No action is taken.

1003 SUB must be on a numeric field; SUB is ignored

Error

The SUB keyword is valid only on a field description statement for a numeric field. This statement is ignored.

1004 SUB parameter not numeric; SUB is ignored

Error

The SUB parameter must specify the name of a field that is defined as a numeric field. This statement is ignored.

1005 SUB field longer than the parameter

Error

The required parameter for the keyword SUB is a counter, the name of a numeric field. In this case, the data in the current numeric field is longer than the counter. This statement is ignored. No action is taken.

1006 INDEX cannot have a numeric parameter on ADDROUT file

Error

When an ADDROUT file is specified by the INDEX keyword, the storage parameter is invalid. The parameter is ignored.

1007 INDEX must give a file name on ADDROUT file

Severe

When an ADDROUT file is used, the name of the ADDROUT file must be specified in the INDEX parameter. The keyword is ignored.

1008 COMP non-numeric parameter was used in an expression

Error

When an expression is specified as the second parameter of the COMP keyword, any named fields, variables, or constants used in the expression must be numeric. The keyword is ignored.

1009 PRTFILE and CFILE may not have record/field definitions

Error

Record and field definitions have no meaning on the PRTFILE and CFILE and are ignored.

1010 FORM not valid for the PRTFILE

Error

The form keyword is valid only on SCS files, that is, not the PRTFILE. The keyword is ignored.

1011 Only the first 16 positions will be sequence checked

Informational

The SEQ keyword checks only the first 16 positions of the field being sequence checked. This message appears only if the total field length exceeds 16 positions.

1012 INDEX not valid for CFILE, TFILE, or PRTFILE

Error

The file referenced by the TFILE, CFILE, or PRTFILE keyword cannot be a keyed or ADDROUT file. Therefore, the INDEX keyword is invalid. The keyword is ignored.

1013 ERROR message may only have 39 characters; first 39 used

Error

The second parameter (help text) of the ERROR keyword can contain 39 characters. The help text is truncated after the first 39 characters.

1015 Field must be numeric if the Data Type is P or B; Data Type of blank assumed

Error

Packed decimal and binary fields must be defined as numeric. A blank is substituted for the Data Type.

1016 Data types P and B are invalid for this file; Data Type of blank assumed

Error

Data types P and B are invalid on CRT and Printer files and TFILE and CFILE. A blank is substituted for the Data Type.

1017 Data Type B is valid only when the field length is <= #9; Data Type of blank assumed

Error

Binary fields are not defined for fields longer than 9 digits.

1018 INSERT non-numeric parameter was used in an expression

Error

When an expression is specified as the INSERT parameter, any named fields, variables, or constants used in the expression must be numeric. The keyword is ignored.

1019 INSERT truncation of high order digit(s) may result

Informational

One or more terms in the expression have greater length(s) than the field for which INSERT is coded. This error does not include the case where terms in the expression are the same length as the field. No action is taken.

1020 LOGON parameter not an alphabetic field; LOGON is ignored

Error

The LOGON parameter must be either a character constant enclosed in apostrophes or the name of an alphabetic field that contains the log-on information.

1022 LOOK field and table are not the same type; ignored

Error

The table specified by the LOOK parameter must contain the same type of data (numeric or alphabetic) as the field being described.

1023 LOOK optional table index receiver is invalid

Error

The index parameter for the keyword LOOK must be numeric with zero decimal positions. The parameter is ignored.

1026 RANGET field and table are not the same type; ignored

Error

The required parameter for the keyword RANGET is tablename, the name of a table that contains pairs of entries which indicate the low and high limits of ranges. In this case, the current field and the table are not the same type.

1027 RANGET number of table entries is not an even number

Error

The table specified by the RANGET keyword must contain pairs of entries that constitute the low and high limits of ranges. Each pair in the table must be complete, which requires an even number of entries. No action is taken.

1028 RANGET table index receiving field is invalid

Error

If the indexname parameter is specified for the keyword RANGET, it must name a numeric field to be used as a variable. The field must have zero decimal positions. The parameter is ignored.

1029 A keyed file may not be multivolume

Severe

Only one device address can be specified for the DEVICE keyword if the diskette file is a keyed file.

1030 RESET parameter is not numeric; RESET is ignored

Error

Only a numeric field is to be specified in the RESET parameter.

1031 Field specified for date editing must be length 3 through 7

Error

A date field must be 3 to 7 positions long. The EDTCDE keyword is ignored.

1032 EDTCDE must be on a numeric field; EDTCDE is ignored

Error

The EDTCDE keyword only allows you to specify editing to be applied to data in numeric fields.

1033 Indexed file cannot be offline multivolume

Severe

The diskette that contains the index file for a multivolume file and all the diskettes containing the data for the file must be online when the file is opened.

1034 SUBST table 2 and field are not the same type; ignored

Error

The required tablename2 parameter for the keyword SUBST contain the entries substituted for the data in the current field. In this case, the current field data and the data to be substituted are not the same type. The SUBST keyword is ignored.

1035 SUBST field and table 2 are not the same length

Error

For the keyword SUBST, the current field and the substituted field (tablename2) must be the same length.

1036 SUBST tables do not have the same number of entries

Error

For the keyword SUBST, tablename1 and tablename2 parameters must have the same number of entries.

1037 SUBST table index receiving parameter is invalid

Error

The optional indexname parameter for the keyword SUBST must be the name of a numeric field with zero decimal positions. The parameter is ignored.

1038 This prompt may overlay fields outside the prompt line

Informational

The length of the prompt exceeds 80 characters and therefore must overlay fields outside the prompt line. Note that another warning is produced if fields fall within the prompt line. No action is taken.

1040 TADD field is longer than 15 characters

Error

No action is taken.

1041 TSUB field is longer than 15 characters

Error

No action is taken.

1042 XCHK table does not have an even number of entries

Error

The XCHK keyword allows you to compare the values contained in a pair of fields to pairs of entries in a table. Each pair in the table must be complete, which requires an even number of entries. No action is taken.

1043 XCHK table index is invalid; ignored

Error

Although the indexes specified by the XCHK keyword can be character or numeric data, the length of the entries cannot exceed 15.

1044 A table may not be specified for this file

Severe

Tables are not valid on files that contain record statements. A table also cannot be specified on the TFILE, CFILE, or PRTFILE.

1045 UDATE and *STATnn may not be altered

Severe

UDATE or *STATnn was coded as a field name which names a field that would normally be altered by the program.

1047 *TOTn may only be used as a field name

Error

The name is ignored.

1048 *STATnn may only be used as a field name

Error

The name is ignored.

1051 AUXDUP field and parameter are not the same type

Error

The required parameter for the keyword AUXDUP is source, the name of a defined field. In this case, the data in the current field is not the same type as that in the source. The keyword is ignored.

1052 AUXST field and parameter are not the same type

Error

The required parameter for the keyword AUXST is target; the name of a defined field. In this case, the data in the current field is not the same type as that in the target. The keyword is ignored.

1053 COMP field and parameter are not the same type

Error

The required parameters for the keyword COMP are test and data. In this case, the data in the current field is not the same type as the specified data parameter. The keyword is ignored.

1054 INSERT field and parameter are not the same type

Error

The parameter for the keyword INSERT can be: source, the name of any named field; expression, any arithmetic expression with a maximum of 30 terms or a maximum length of 256 characters; or constant, any numeric or character data. In this case, the data in the current field is not the same type as the specified parameters. The keyword is ignored.

1055 RANGE field and parameter are not the same type

Error

The required parameters for the keyword RANGE are low and high numeric constants, character constants, or names of numeric or character fields containing the value. In this case, the data in the current field and the low and high values are not the same type. The keyword is ignored.

1057 AUXDUP field and parameter are not the same length

Error

For the keyword AUXDUP, the current and source fields are not the same length. No action is taken.

1058 AUXST field and parameter are not the same length

Error

For the keyword AUXST, the current and target fields are not the same length. No action is taken.

1059 SHIFT field and parameter are not the same length

Error

The required parameter for the keyword SHIFT is codes, which is a string of characters, one for each character position in the field. The current field and the codes parameter must be the same length. No action is taken.

1060 EDTCDE cannot be specified with data types P or B

Error

Neither binary nor packed decimal data may be edited.

1061 This file may only contain TABLE statements

Severe

This applies to a file where DEVICE is not coded.

1062 No valid A-specifications in this program

Severe

The valid A coding specifications are file, record, table, and field description statements.

1063 CRT file specified more than once

Severe

Only one CRT file can be specified in a source program.

1064 CRT file field(s) fall outside the display

Severe

The lines and positions field (columns 39 through 44) must specify locations that are valid for the display. The fields should not overlap or be longer than the screen size allows.

1065 CRT file fields overlap

Informational

The field being defined specifies one or more positions that are already specified for a different field in this record. No action is taken.

1066 CRT file record length exceeded

Severe

The sum of the input or both field lengths in a record is greater than the record length.

1067 This file must have the same length as the CRT file

Severe

This applies to the TFILE, CFILE, or PRTFILE.

1068 CRT field is in the prompt line

Informational

A field on a CRT file totally or partially falls within the prompt line (line 2 of the screen). This message should be ignored if the record is always referenced with SLNO greater than 2.

1069 A literal cannot be used as input

Severe

A literal may not be specified on a record which is read from the C-specifications. The literal is ignored. This could produce erroneous results.

1070 DISK/COMM/PRINTER file field falls outside the record

Severe

The location field (columns 39 through 44) must specify a number such that both the starting and ending positions of the field are equal to or less than the record length. The field being defined exceeds the defined record length.

1071 DISK/COMM/PRINTER file fields overlap

Informational

Space defined for this field has already been assigned to another field (partially or completely). The location field (columns 39 through 44) must not specify a position in a record that is equal to or greater than the ending position of the previous field in the record. No action is taken.

1072 Fields are not checked for overlap beyond 4096 bytes

Informational

This applies to records longer than 4096 bytes. DE/RPG cannot check fields on this record that cause the 4096-byte limit to be exceeded. No action is taken.

1073 Length on record is larger than the length on the file

Severe

The length specified on a record statement is greater than the length specified on the file statement.

1074 Key was already specified on this record; field assumed

Error

Only one key field (K in column 17) can be specified for a record. Column 17 is assumed to be blank.

1075 A key is invalid on this record; field assumed

Error

The type is changed from a key type to a field type.

1076 Key position/length not the same in all records

Severe

1077 This record must have a key

Severe

Every record in a keyed field must have a key.

1078 Accumulative table length exceeds the file length

Severe

The table and the entire file are ignored.

1079 Key fields may not be longer than 28 bytes

Severe

1080 File statement has no associated record(s) or table(s)

Severe

The TFILE need not have, and the CFILE and PRTFILE must not have, records and fields. All other files, however, must have them.

1081 Record Statement has no associated field statements

Severe

A record description statement on a CRT file must be followed by at least one field description statement.

1083 LOOK field and table not the same length; keyword ignored

Error

The length referred to is the actual length, not just the integer length.

1084 RANGET field and table not the same length; keyword ignored

Error

The length referred to is the actual length, not just the integer length.

1085 Keyword not valid for WRITE operation; keyword ignored

Error

This applies to a CRT record written from C-specifications. Only EDTCDE, a literal, and DSPATR are valid here.

1087 Record contains neither named field(s) nor literal(s) or contains only working storage fields

Error

A record not on the CRT file should contain (if that record contains any fields) at least one named field or one literal. In addition, the record should not contain all working storage fields.

1090 Referenced file must not specify a device

Error

The file name specified on a **CDTDATA statement must refer to a file for which the DEVICE keyword is not specified.

1091 More data records specified than indicated by NUMENT

Error

1092 Insufficient data for this table file

Error

The number of records of data is less than the number specified by NUMENT. The extra entries are unusable. If no data records are supplied, the table is filled with blank entries (for character data) or zeros (for numeric data).

1093 No data specified for this table file

Error

A compile-time table was specified, but no table entries were included in the source.

1094 Length of this data exceeds the table file length

Error

The length of a record must be equal to the accumulative length of the tables for this file. The data specified exceeds this length. No action is taken.

1095 Non-numeric table element found for a numeric table

Error

The column(s) used for a table element contain non-numeric data. The rightmost column may contain the digits 0 through 9, or negative digits (which appear as the right brace and the letters J through R, for negative 0 to 9, respectively). The other columns in the same table element may only contain the digits 0 through 9. If the data is shorter than the length specified on the table definition, blanks or character data from another table can cause this error. No action is taken.

1096 Data already defined for this file

Error

Two sets of compile-time table data reference the same file; only the first set will be used. No action is taken.

1100 First C-specification subroutine operation not BEGSR

Severe

No calculation statements for a subroutine are allowed to precede the beginning statement of the subroutine (BEGSR). This line is ignored.

1101 Operation field missing

Severe

Each calculation statement must have a valid operation code entered in the operation field (columns 28 through 32). This line is ignored.

1102 N in columns 9, 12, or 15 with no corresponding indicator

Error

An N (not) is specified in column 9, 12, or 15 but no indicator is specified in the associated field (columns 10 and 11, 13 and 14, or 16 and 17). The N is ignored.

1103 *TOT in result field is not followed by a digit 1 through 9

Error

*TOT in the result field is an invalid entry. It must be *TOT1 through *TOT9. The statement is ignored.

1104 Invalid entry in result field; statement ignored

Error

Valid entries for arithmetic operations are the name of a defined numeric field or array element or the name of a new numeric field; for compare operations, the destination label for a branch; for move operations, the name of a defined field or array element or the name of a new field; for bit operations, the name of a one-position character field or array element or the name of a new character field.

1105 UDATE is invalid in result field; statement ignored

Error

UDATE is a system variable that contains the date. This field cannot be changed.

1106 Decimal position field has an entry, but length field is blank

Error

If no length (columns 49 through 51) is specified, the decimal position field (column 52) is invalid. This line is ignored.

1107 Decimal position entry > length entry; assume equal to length

Error

The number of decimal positions in a field cannot be greater than the length of the field.

1108 Length too large for numeric field; 15 assumed

Error

The maximum length of a numeric field is 15 positions.

1109 Three resulting indicators are the same

Error

The resulting indicator has no meaning because it will be turned on regardless of the results of the operation. No action is taken.

1110 Last C-specification operation not ENDSR

Severe

An ENDSR operation code is required to end a subroutine.

1111 BEGSR operation is not preceded by ENDSR operation

Severe

A subroutine cannot exist within another subroutine. The first subroutine must be ended with an ENDSR operation code before the next subroutine is started with a BEGSR operation code.

1112 Missing label definition in factor 1 field

Error

This statement is ignored. The line is ignored.

1113 Invalid label definition in factor 1 field

Error

This statement is ignored.

1114 MVR only valid immediately following DIV operation

Error

The MVR operation must immediately follow the DIV operation to which it applies. Otherwise, the remainder is lost and the MVR operation is invalid. This statement is ignored.

1115 Half adjust invalid for DIV operation followed by MVR operation

Error

When a MVR operation follows a DIV operation, the result of the DIV operation cannot be half-adjusted because the remainder from the DIV operation is required for the MVR operation. The half adjust entry is ignored.

1116 No resulting indicators were specified

Error

The entry in the resulting indicators field of a LOKUP operation is required. This entry specifies the type of match to be searched for by the operation. The line is ignored.

1117 Both high and low result indicators specified for LOKUP

Error

A search for low and equal or for high and equal can be specified, but not for high and low.

1118 Invalid bit literal specified in factor 2 field

Error

A valid bit literal must consist of whole numbers 0 through 7; one number cannot be specified more than once per literal; and the literal must be enclosed in apostrophes. For example, '1236' is valid, '09' is not. This statement is ignored.

1119 Same bit is designated more than once in factor 2 field

Error

Factor 2 of BITON, BITOF, or TESTB operation specifies the mask that identifies the bits to be set on, set off, or tested. Specifying the same bit more than once in the mask does not affect the result of the operation.

1120 Invalid entry in factor 2 field; statement ignored

Error

For BITON, BITOF, and TESTB operations, factor 2 must be a 1-position alphameric field or a bit literal.

1121 Division by zero is invalid; statement ignored

Error

The contents of factor 2 of a DIV operation cannot be zero.

1122 Conditioning indicator ignored for BEGSR, ENDSR, TAG

Error

BEGSR, ENDSR, and TAG statements identify the beginning, ending, or branching points in a subroutine and are not affected by indicators in columns 9 through 17.

1124 Factor 1 field should be blank for this operation

Error

Blanks are assumed for the entry.

1125 Factor 2 field should be blank for this operation

Error

Blanks are assumed for the entry.

1126 Result field should be blank for this operation

Error

Blanks are assumed for the entry.

1127 Half adjust should be blank for this operation

Error

Blanks are assumed for the entry.

1128 Length field should be blank for this operation

Error

Blanks are assumed for the entry.

1129 Decimal position field should be blank for this operation

Error

Blanks are assumed for the entry.

1130 High indicator field should be blank for this operation

Error

Blanks are assumed for the entry.

1131 Low indicator field should be blank for this operation

Error

1132 Equal indicator field should be blank for this operation

Error

1134 Entry following * in factor field not TOTn or STATnn

Error

The asterisk (*) is invalid in any names other than *TOT1 through *TOT9 and *STAT01 through *STAT29. This statement is ignored.

1135 In factor field *TOT is not followed by a digit 1 through 9

Error

*TOT is only valid when it is followed by a digit 1 through 9. This statement is ignored.

1136 In factor field *STAT not followed by 01 through 29

Error

*STAT is only valid when it is followed by the digits 01 through 29. This statement is ignored.

1137 Missing label reference in factor 2 field

Error

In factor 2 field, a GOTO operation does not reference a TAG or ENDSR label; or an EXSR operation does not reference BEGSR label. This statement is ignored.

1138 Invalid label reference in factor 2 field

Error

In factor 2 field. A GOTO operation does not reference a TAG or ENDSR label; or an EXSR operation does not reference BEGSR label. This statement is ignored.

1139 Condition indicator different for MVR and preceding DIV

Error

The DIV operation and its associated MVR operation should be conditioned by the same indicators in columns 9 through 17 so that the two operations are performed together. The MVR operation is invalid if the DIV operation is not executed immediately preceding it. This statement is ignored.

1140 End of file indicator not specified

Severe

On a READ or READP operation, an end of file indicator was not specified in columns 58 and 59. This statement is ignored.

1141 Indicator is not specified in columns 54 and 55

Severe

On a chaining operation, the 'no record found' indicator must be specified in columns 54 and 55.

1147 Both Factor 2 and the result field must be alphameric; statement ignored

Error

MOVEA is only valid with alphameric data.

1148 MOVEA cannot move a field to a field; statement ignored

Error

To move a field to another field, use MOVE or MOVEL.

1149 MOVEA cannot move a table by itself; statement ignored

Error

MOVEA can only move a table to a field or go to another table.

1150 An entry is required in factor 1 field for this operation

Error

An entry in the factor 1 field is required for the following operations: ADD, SUB, MULT, DIV, CABxx, COMP, BEGSR, TAG, LOKUP, DELET, SETLL, and CHAIN. This statement is ignored.

1151 An entry is required in factor 2 field for this operation

Error

The only operations that do not require an entry in the factor 2 field are: BEGSR, ENDSR, SETOF, SETON, and TAG. This statement is ignored.

1152 An entry is required in result field for this operation

Error

An entry in the result field is required for the following operations: ADD, SUB, MULT, DIV, Z-ADD, Z-SUB, MOVE, MOVEL, CABxx, BITOF, BITON, and TESTB. This statement is ignored.

1153 Factor 1 not numeric; statement ignored

Error

Factor 1 must contain a numeric constant or the name of a numeric field or array element.

1125 Factor 2 field should be blank for this operation

Error

Blanks are assumed for the entry.

1126 Result field should be blank for this operation

Error

Blanks are assumed for the entry.

1127 Half adjust should be blank for this operation

Error

Blanks are assumed for the entry.

1128 Length field should be blank for this operation

Error

Blanks are assumed for the entry.

1129 Decimal position field should be blank for this operation

Error

Blanks are assumed for the entry.

1130 High indicator field should be blank for this operation

Error

Blanks are assumed for the entry.

1131 Low indicator field should be blank for this operation

Error

1132 Equal indicator field should be blank for this operation

Error

1134 Entry following * in factor field not TOTn or STATnn

Error

The asterisk (*) is invalid in any names other than *TOT1 through *TOT9 and *STAT01 through *STAT29. This statement is ignored.

1135 In factor field *TOT is not followed by a digit 1 through 9

Error

*TOT is only valid when it is followed by a digit 1 through 9. This statement is ignored.

1136 In factor field *STAT not followed by 01 through 29

Error

*STAT is only valid when it is followed by the digits 01 through 29. This statement is ignored.

1137 Missing label reference in factor 2 field

Error

In factor 2 field, a GOTO operation does not reference a TAG or ENDSR label; or an EXSR operation does not reference BEGSR label. This statement is ignored.

1138 Invalid label reference in factor 2 field

Error

In factor 2 field. A GOTO operation does not reference a TAG or ENDSR label; or an EXSR operation does not reference BEGSR label. This statement is ignored.

1139 Condition indicator different for MVR and preceding DIV

Error

The DIV operation and its associated MVR operation should be conditioned by the same indicators in columns 9 through 17 so that the two operations are performed together. The MVR operation is invalid if the DIV operation is not executed immediately preceding it. This statement is ignored.

1140 End of file indicator not specified

Severe

On a READ or READP operation, an end of file indicator was not specified in columns 58 and 59. This statement is ignored.

1141 Indicator is not specified in columns 54 and 55

Severe

On a chaining operation, the 'no record found' indicator must be specified in columns 54 and 55.

1147 Both Factor 2 and the result field must be alphameric; statement ignored

Error

MOVEA is only valid with alphameric data.

1148 MOVEA cannot move a field to a field; statement ignored

Error

To move a field to another field, use MOVE or MOVEL.

1149 MOVEA cannot move a table by itself; statement ignored

Error

MOVEA can only move a table to a field or go to another table.

1150 An entry is required in factor 1 field for this operation

Error

An entry in the factor 1 field is required for the following operations: ADD, SUB, MULT, DIV, CABxx, COMP, BEGSR, TAG, LOKUP, DELET, SETLL, and CHAIN. This statement is ignored.

1151 An entry is required in factor 2 field for this operation

Error

The only operations that do not require an entry in the factor 2 field are: BEGSR, ENDSR, SETOF, SETON, and TAG. This statement is ignored.

1152 An entry is required in result field for this operation

Error

An entry in the result field is required for the following operations: ADD, SUB, MULT, DIV, Z-ADD, Z-SUB, MOVE, MOVEL, CABxx, BITOF, BITON, and TESTB. This statement is ignored.

1153 Factor 1 not numeric; statement ignored

Error

Factor 1 must contain a numeric constant or the name of a numeric field or array element.

1154 Factor 2 not numeric; statement ignored

Error

Factor 2 must contain a numeric constant or the name of a numeric field or array element.

1155 Result field not numeric; statement ignored

Error

The result field must specify the name of a numeric field or array element or it must define a new numeric field.

1156 Name longer than 6 characters; name ignored

Error

Label, field, and table names must be from 1 to 6 characters in length.

1164 Invalid entry in factor 2 field; statement ignored

Error

The factor 2 field must specify the mask for the BITOF, BITON, or TESTB operation. The entry can be the bit numbers, enclosed in apostrophes, or the name of a one-character field or array element that contains the mask.

1166 Result field must be a 1 position alphanumeric field

Error

This statement is ignored.

1168 Factor 2 field is not a BEGSR label reference

Error

The only valid entry in factor 2 of an EXSR operation is the name of another subroutine (factor 1 of a BEGSR operation). This statement is ignored.

1169 Factor 2 must be a record name for this operation

Error

Factor 2 must specify a record name defined on a record description statement. This statement is ignored.

1170 Device invalid for this operation; statement ignored

Severe

The record name or file name specified in factor 2 is associated with a device for which this I/O operation is invalid.

1171 C-specifications reference TFILE, CFILE, or PRTFILE

Severe

A calculation statement cannot refer to a file specified as the TFILE, CFILE, or PRTFILE on the job specification statement. This statement is ignored.

1172 Factor 2 must be a file name; statement ignored

Error

The name specified in factor 2 must be defined on a valid file description statement.

1174 Half adjust not necessary

Error

1175 Factor 1 and factor 2 must both be numeric or character

Error

The constants, field names, or array names specified in factors 1 and 2 must be the same data type. This statement is ignored.

1176 Relative record number in factor 1 must have zero decimal positions

Error

When the name of a field or array element is specified in factor 1, the named field or array element must be defined as numeric with 0 decimal positions. The statement will be ignored.

1178 Factor 1 length is not equal to key length of file

Error

The constant, field, or array element specified in factor 1 must be the same length as the key field defined on a field description statement. This statement is ignored.

1179 For non-keyed file, factor 1 must be numeric record number

Error

A relative record number can be specified in factor 1 as either a numeric constant or the name of a numeric field or array element containing the relative record number. This statement is ignored.

1180 Reference must be to TAG or ENDSR label in same subroutine

Error

GOTO operations and compare and branch (CABxx) operations cannot refer to a label outside the subroutine in which the operation occurs. This statement is ignored.

1181 Position is greater than number of elements in array

Error

The position portion of an array-element name cannot be larger than the number of elements in the array. This statement is ignored.

1182 Position decimal positions must be zero

Error

The position portion of an array-element name must be a whole number or the name of a field containing a whole number. The field must be defined as numeric with 0 decimal positions. This statement is ignored.

1183 Position must be numeric field or literal

Error

The position portion of an array-element name must be a whole number or the name of a numeric field containing a whole number. This statement is ignored.

1184 Array name is specified without a position

Error

Except for the LOKUP operation, if an array name is specified, it must be specified as name position in which name is the name of the array, and position is the relative position of the element in the array. This statement is ignored.

1185 Position is only valid with an array name

Error

The name is specified as the name, position. However, the name is not defined as the name of an array (table) on a table description statement. This statement is ignored.

1186 Factor 1 incompatible with array specification

Error

Length (and decimal positions) of factor 1 and factor 2 must match. This statement is ignored.

1187 Subroutine cannot call itself

Severe

The name specified in factor 2 of the EXSR operation must be the name of a subroutine other than the one in which the EXSR operation occurs. This statement is ignored.

1188 Factor 1 should be blank

Error

For Z-ADD/Z-SUB operations, the factor 1 field should be blank.

1189 Only READ and UPDAT may reference ADDROUT file

Severe

This statement is ignored.

1190 Keyed access invalid for multivolume file

Severe

The file specified in factor 2 is a multivolume file, which is not valid in a keyed-access operation. This statement is ignored.

1191 Random access invalid for offline multivolume file

Severe

The multivolume file specified for a random access operation must be online. This statement is ignored.

1201 Algorithm number previously specified

Error

Each algorithm number specified in a program (**SLFCHKnn) must be unique. The algorithm is ignored.

1202 More than 1 MOD keyword; last MOD is used

Error

The MOD keyword assigns the modulus used in check-character calculations. One modulus must be specified in each self-check algorithm definition. No more than one can be specified.

1203 MOD parameter must be greater than 2 and less than 127

Error

The algorithm is ignored.

1204 More than 1 DISP keyword; last DISP is used

Error

Only one DISP keyword can be specified in each self-check algorithm definition. If DISP is not specified, a value of 1 is assumed.

1205 DISP parameter must be less than 32

Error

Zero is assumed.

1206 More than 1 WEIGHTS keyword; last WEIGHTS is used

Error

The WEIGHTS keyword specifies the weights to be used during product calculations. One WEIGHTS keyword must be specified in each self-check algorithm definition. No more than one can be specified.

1207 More than 32 weights specified; extra ignored

Error

1208 WEIGHT value not numeric; parameter ignored

Error

The WEIGHTS parameter can be any whole number that is less than the value assigned to the modulus.

1209 WEIGHT value not less than MOD value

Error

The WEIGHTS parameter can be any whole number that is less than the value assigned to the modulus. The algorithm is ignored.

1210 More than 1 OPT keyword; last OPT is used

Error

Only one OPT keyword can be specified in each self-check algorithm definition.

1211 OPT parameter control character exclusion error

Error

The OPT parameter consists of a string of up to 5 characters. By specifying a certain character in one position, you preclude some characters in other positions, as follows:

- If D in character 1, no E in character 2
- If E in character 2, no 2 in character 4

No action is taken.

1212 OPT parameter invalid positional character; ignored

Error

Valid positional characters are:

- For position 1 – blank, D, U
- For position 2 – blank, D, K, E
- For position 3 – blank, C
- For position 4 – blank, 1, 2
- For position 5 – blank, D

1213 OPT parameter too long; maximum is 5 characters

Error

Extra characters are ignored.

1214 Required MOD keyword is missing; algorithm ignored

Error

The MOD keyword assigns the modulus used in check-character calculations. One modulus must be specified in each self-check algorithm definition.

1215 Required WEIGHTS keyword is missing; algorithm ignored

Error

The WEIGHTS keyword specifies the weights to be used during product calculations. One WEIGHTS keyword must be specified in each self-check algorithm definition.

1216 DISP parameter is greater than field length

Error

The DISP keyword specifies the relative location of the rightmost check character in the self-check field. The number specified cannot be greater than the length of the self-check field. No action is taken.

1217 Referenced algorithm number is not defined

Error

A self-check algorithm number is referenced in the program but is not defined. Only self-check numbers 10 and 11 are IBM-supplied and can be referenced without being defined in the source program. No action is taken.

1218 More than 13 algorithms defined; extra ignored

Error

1219 OPT keyword apostrophes incorrect or missing

Error

The parameter for the OPT keyword must be a string of up to 5 character positions enclosed in apostrophes. No action is taken.

This page is intentionally left blank.

Appendix G. A-, Z-, and C-Specification Forms

This page is intentionally left blank.

IBM International Business Machines Corporation

IBM 5280 DATA DESCRIPTION SPECIFICATIONS

GX21-9362-2
Printed in U.S.A.

Job No.	Dataset	Keying Instruction	Graphic								
Operator	Date		Key								

Source Document	Page of

A

Sequence	Form Type	Comment (*)	Reserved	Indicator (for CHECK (BY, BV) or ERROR)	Reserved	Name Type (F/R/K/T)	Reserved	Dataset/Record/ Field/Table Name	Reserved	Length	Data Type	Reserved	Decimal Positions (0-9)	Usage (I/O/B/W)	Location: Line	Location: Pos	Editing
1 2 3 4 5	6	7	8	9 10	11 12 13 14 15 16	17	18	19 20 21 22 23 24 25 26	27 28 29	30 31 32 33 34	35	36	37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Checks=CHECK (code. . .)

Auto Dup	=AD	Mandatory Entry	=ME
Auto Skip	=AS	Mandatory Fill	=MF
Blank Check	=BC	Rt Adj–Blank Fill	=RB
Bypass	=BY	Right to Left	=RL
Bypass on Verify	=BV	Rt Adj–Zero Fill	=RZ
Data Required	=DR	[1]Self-Check	=nxx
Dup Disable	=DD	D=M/G (Check/Gen)	
Field Exit Required	=FE	xx=Modulus	
Lower Case	=LC		

Functions

[1]ADD (name)
[1]AUXDUP (name)
[1]AUXST (name)
[1]COMP ([2]test fld 1[3]@...fldn 'literal' [indicator])
DSPATR ([4]attr...)
[1]EDTCDE (code 'float')
ERROR (code ['message'])
[1]EXSR (subroutine)
[1]INSERT (fld 1[3]@...fldn 'literal')
LOOK (table [index])
PMT (prompt)
[1]RANGE (low high)
[1]RANGET (table [index])
[1]RESET ([*TOTn] [name])
[1]SEQ ([2]test)
[1]SETOF (ind)
[1]SETON (ind)
SHIFT ([5]shift)
[1]SUB (name)
[1]SUBST (table 1 table 2 [index])
[1]TADD ([*TOTn] [name])
[1]TSUB ([*TOTn] [name])
[1]XCHK (table index 1 index 2 'literal')

[1]not valid for COBOL programs
[2]test=EQ,GE,GT,LE,LT,NE
[3]@=+,–,*,/
[4]attr=BL,CA,CS,HI,ND,RI,UL
[5]shift=A,D,H,N,V,W,X,Y

0 1	A																
0 2	A																
0 3	A																
0 4	A																
0 5	A																
0 6	A																
0 7	A																
0 8	A																
0 9	A																
1 0	A																
1 1	A																
1 2	A																
1 3	A																
1 4	A																
1 5	A																
	A																
	A																
	A																
	A																
	A																

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly

UM/050

1-5 Identifies the source statement order.

6 Identifies the type of source statement.

7 An * indicates a user comment.

8 Reserved.

9-10 Specifies the indicator that is used to control field bypassing or displaying user error codes.

11-16 Reserved.

17 Defines the type of statement: F = data set, R = record, K= key field, T = table, blank = field.

18 Reserved.

19-26 Specifies the name for: data set (max 8 characters), record (max 8 characters), field (max 6 characters) or table (max 6 characters).

27-29 Reserved.

30-34 Specifies the length:
data set = maximum record length is required.
record = number of characters (1-8192)
field = number of characters (1-256 for alphameric or 1-15 for numeric).

35 Defines the data type for the field:

A = alpha	S = signed numeric
B = binary	V = right half only
C = use SHIFT keyword	W = right half shift
D = digits only	X = alpha only
H = hexadecimal	Y = numeric only
N = numeric	blank = alpha or numeric depending on the field type.
P = packed	

36 Reserved.

37 Specifies the number of decimal positions (0-9).

38 Specifies how the data in a field on the display screen is processed. I = input, O = output, B = both, W = workspace.

39-44 Specifies the location of the field within a record or on the display screen.

45-80 Specifies parameters for data sets, files, records, tables, and fields:

Data sets and files (F in column 17)

BLKING ([*DBL] [*FMTU or *FMTS])—specifies blocking characteristics for data sets:
- *DBL specifies to use two physical buffers
- *FMTU specifies that the records are unblocked (Basic or H data exchange)
- *FMTS specifies that the records are blocked and spanned (I data exchange)

DEVICE (dev-type address)—physical device type for the data set:
- dev-type is COMM or COMM 3270 (communications), CRT (keyboard/display), DISK (diskette), MREAD (magnetic stripe reader), PRINTER (printer).
- address is the 2-character logical ID or the 4-character device address (X'xxxx' where xxxx is the physical address).

DSPSIZ (lines 80)—specifies display size: lines = 6, 12, or 24.

FORM (length [overflow-line overflow-ind])—Specifies the printer page size; length specifies the lines available on the page, overflow-line specifies the line that sets the overflow indicator on, and overflow-ind specifies the indicator that is set on.

INDEX ([storage] [data set])—At least one parameter must be specified. Specifies the storage reserved for the sparse index and the index data set name: storage specifies the space required for the index, data set specifies the name of the index data set.

LABEL (name of data set)—diskette data set name.

LOGON ('message' or name)—Specifies the log on information when required for communications. The parameter can be either a message enclosed in single quotes or a variable name.

MARK (*POSnnnn)—Specifies the position in a data record where an E is place if the Field Mark key is pressed.

NUMENT (number)—number of records in a data set when used for dynamic allocation of the data set or the number of entries in a table.

VMARK (*POSnnnn)—Specifies the position in a data record where a V is placed after the record is verified.

Records (R in column 17)

DSPATR (attr...)—Specifies the display attributes that apply to all the fields in the record.

RECID (*POSnnnn 'c')—Specifies the position that identifies the single character record type 'c' from a data set with more than one record type (nnnn is 1 to 1892).

SPACEA (n)—Causes the printer to space n lines after the record is printed.

SPACEB (n)—Causes the printer to space n lines before the record is printed.

SKIPA (n)—Causes the printer to skip to line n after the record is printed.

SKIPB (n)—Causes the printer to skip to line n before the record is printed.

Field (Blank in column 17)

ADD (name)—Adds the data in the current field to the named field with decimal alignment.

AUXDUP (name)—Duplicates data from the named field if the Dup key is pressed or the Auto Dup/Skip switch is on.

AUXST (name)—Stores the current field in the named field if the Auto/Dup switch is on.

CHECK (parameter)—Specifies the keyboard edits to be applied to the field.

COMP (test fld 1 @...fldn 'literal' [indicator])—Compares the current field with a named field, the specified expression, or a literal and optionally turns on an indicator if the compare is true.

DSPATR (attr...)—Controls the display attributes for each field.

EDTCDE (code 'float')—Specifies the editing that is to be applied to data in numeric fields, where:
- code is a single character that controls the use of editing characters specified by the EDITC keyword.
- float can be either:
  - *, which places asterisks in the character positions to the left of the first digit
  - cu, which floats the two-character currency symbol used on EDITC.

ERROR (code ['message'])—Locks the keyboard, displays an error code, and optionally displays an error message (when the Help key is pressed) if the specified indicator is turned on.

EXSR (subroutine)—Branches to the named calculation subroutine.

INSERT (fld 1 @...fldn 'literal')—Inserts the named field, expression or literal into the current field.

LOOK (table [index])—Compares the current field for a match in a table, and optionally places the index value of the table entry in index.

PMT (prompt)—Displays the prompt message when the current field is entered.

RANGE (low high)—Specifies the low and high limits for data that can be entered into the current field.

RANGET (table[index])—Compares the current field for a match in a table of low and high limits, and optionally places the index value of the table entry in index.

RESET ([*TOTn] [name])—Only one parameter is allowed. Sets the named counter to 0..

SEQ (test)—Sequence checks the data in the current field against the data from the previous sequence check using the specified test.

SETOF (ind)—Turns the specified indicator off.

SETON (ind)—Turns the specified indicator on.

SHIFT (shift)—Specifies the shift and character set for each character in a field when C is specified for data type.

SUB (name)—Subtracts the data in the current field from the named field with decimal alignment.

SUBST (table 1 table 2 [index])—Compares the current field for a match in table 1. If there is a match, replaces the current field with data from the corresponding entry in table 2. Optionally places the index value of the table entry in index.

TADD ([*TOTn] [name])—Only one parameter is allowed. Adds the current field to the named counter.

TSUB ([*TOTn] [name])—Only one parameter is allowed. Subtracts the current field from the named counter.

XCHK (table index1 index2)—Compares the indexes to see if they match an entry in a named table of index pairs.

Continuation—Specifies to continue on the next line:
- \+ specifies to continue with the first nonblank character in position 45-80 on the next line (ignore leading blanks).
- \- specifies to continue from position 45 on the next line (leading blanks are included).

IBM International Business Machines Corporation

IBM 5280 GENERAL UTILITY SPECIFICATIONS

GX21-9361-1
Printed in U. S. A.

Job		Keying Instruction	Graphic							
Operator	Date		Key							

Description	Page of

Z

Sequence	Form Type	Name Type	Format ID (1-9, A0-Z9)	Job/Format/ Subroutine Name	Reserved	Repeat (1-9, N)	Mode (E/R)	Test Conditions: AND(A)	Test Conditions: Position to be Tested (*POSnnnn)	Test Conditions: Reserved	Test Conditions: Condition	Test Conditions: Character to Test for ('C')	Reserved	Next Format ID (0-9, A0-Z9)	Reserved	Options
1 2 3 4 5	6	7	8 9	10 11 12 13 14 15 16 17	18 19	20	21	22	23 24 25 26 27 28 29 30	31 32	33 34	35 36 37	38 39 40 41 42 43 44	45 46	47 48 49 50 51 52 53 54	55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
01	Z								* P O S		E Q	' '				
02	Z								* P O S		E Q	' '				
03	Z								* P O S		E Q	' '				
04	Z								* P O S		E Q	' '				
05	Z								* P O S		E Q	' '				
06	Z								* P O S		E Q	' '				
07	Z								* P O S		E Q	' '				
08	Z								* P O S		E Q	' '				
09	Z								* P O S		E Q	' '				
10	Z								* P O S		E Q	' '				
11	Z								* P O S		E Q	' '				
12	Z								* P O S		E Q	' '				
13	Z								* P O S		E Q	' '				
14	Z								* P O S		E Q	' '				
15	Z								* P O S		E Q	' '				
	Z								* P O S		E Q	' '				
	Z								* P O S		E Q	' '				
	Z								* P O S		E Q	' '				
	Z								* P O S		E Q	' '				
	Z								* P O S		E Q	' '				

Options

Job Line

CFILE (data set)
DATE (*DMY/*YMD)
EDITC (cuptd)
ENTRATR ([1]attr...)
EXITATR ([1]attr...)
JOBOPT ([*NOPMT][*NOOPEN])
PRTFILE (data set)
SHARE (names)
SHARER (names)
STATUS (name)
TFILE (data set [delfreq])

[1]attr=BL,CS,HI,ND,RI,UL

Entry Lines

CLRL (number)
EOJ [{['job' dev| [*PASS]}]
SLNO (line)
WRITE (name)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

*Number of sheets per pad may vary slightly.

1-5 Identifies the source statement order.

6 Identifies the type of source statement.

7 Names the type of source statement:
*–User comment
J–Job specification
blank–Format specification

8-9 The identification associated with this format:
1 through 9–A single numeric character ID.
A0 through Z9–A two-character ID consisting of an alphabetic character followed by a numeric character.

10-17 The name used to:
- identify the job (J in column 7).
- identify the format or subroutine (blank in column 7).

These columns are not used if column 21 contains an R.

18-19 Reserved.

Note: Columns 20-54 are not used if column 7 contains a J.

20 Specifies the number of times the format is repeated before the next format is used:
1 through 9–Repeat the format for the specified number of times unless the SEL FMT or NEXT FMT key is pressed.
blank or N–Repeat the format until the SEL FMT or NEXT FMT key is pressed.

21 Specifies how the format is used:
E–(Entry) used to enter and display data.
R–(Review) used to select a format for scan, update, or verify of existing records.

22-37 Used for logical selection of a format. Multiple tests are allowed. In enter mode, the format selected is used to format the *next* record entered. In review mode, the format selected is used to display the *current* record.

22 In review mode (column 21 contains an R), an A specifies the *anding* of two characters in the data record to create a unique record identifier.

23-30 *POSnnnn identifies the position in the data record to be tested, where nnnn is a numeric value from 1 to 1024.

31-32 Reserved.

33-34 The characters EQ or blank when a character to test for is specified in position 35-37.

35-37 Specifies the character that controls format selection if it matches the character in the data record.

38-44 Reserved.

45-46 Specifies the identification of the format used for the entry or display of the next record. If columns 22-37 are specified, the format is selected when a match occurs. If columns 22-37 are not specified in enter mode (E in column 21), the format is selected when the repeat count (column 21) is met or the NEXT FMT key is pressed. If columns 22-37 are not specified in review mode (R in column 21), the format is selected if no previous match occurs.

47-54 Reserved.

55-80 Keywords that specify information used for jobs or formats:

JOB specifications (J in column 7):

CFILE (data set)–Includes the COPY function in the job. The parameter data set is the data set name from which records will be copied.

DATE(*DMY/*YMD)–The format of the date available in UDATE. The default is *MDY, where M = month, D = day, and Y = year.

EDITC(cuptd)–Five characters that define the editing control for output fields, where:
- cu is a two-character currency symbol (default = b$).
- p is the decimal point character (default = .).
- t is the thousand separator character (default = ,).
- d is the date separator character (default = /).

The system default for this option is b$.,/ if EDITC is not specified.

ENTRATR (attr...)–Specifies the attributes that are applied to all input/both fields only when the fields are being entered, where attr is:
BL (blink)
CS (column separators)
HI (high intensity)
ND (nondisplay)
RI (reverse image)
UL (underline)
A combination of attributes can also be used.

EXITATR (attr...)–Specifies the attributes that are applied to all input/both fields after the fields have been entered. See the ENTRATR for a description of the attr parameter.

JOBOPT([*NOPMT] [*NOOPEN])–At least one of the parameters must be specified. Where
- *NOPMT specifies to bypass the prompts for data set information at the beginning of the job.
- *NOOPEN specifies to bypass the automatic opening of all files except the transaction file specified by the TFILE keyword.

PRTFILE (data set)–Includes the PRINT function in the job. The parameter data set is the data set name to be assigned to the printer.

SHARE(names)–Allows other programs to read or write records in the data set specified by the names parameter while this program is executing.

SHARER(names)–Allows other programs to *read* records in the data set specified by the names parameter while this program is executing.

STATUS(name)–Establishes a variable that can be used to check the status of an I/O device after an I/O operation. The parameter name is the name assigned to the variable.

TFILE(data set [delfreq])–Specifies the data set where records will be written after a format is completed, where:
- data set is the name of the data set that receives the transaction records.
- delfreq specifies how often deleted records are automatically inserted in the transaction data set.

Format Specifications (blank in column 7):

CLRL (number)–Specifies the number of display lines cleared, starting from the first line of the display, when a new record is to be entered. If *NO is specified, none of the display lines are cleared.

EOJ [([' job ' dev] [*PASS])]–Causes the end of the job upon completion of the format. The optional parameters are:
job–name of the next job job to execute.
dev–the device address where the next job is located.
*PASS–suppress job production statistics.

SLNO (line)–Specifies the uppermost display line that can be used. All display line references are based on the specified line as line one.

WRITE (name)–Specifies that the current data is written to the data set in the record format specified by *name*. If *NO is specified, the current data is not written to the data set.

Continuation can be specified by a + or - as the last character on the line, where:
+ specifies to continue with the first nonblank character in positions 55-80 on the next line (ignore leading blanks).
- specifies to continue from position 55 on the next line (including leading blanks).

RPG CALCULATION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9093-3 UM/050*
Printed in U.S.A.

Program		Keying Instruction	Graphic	Card Electro Number
Programmer	Date		Key	

Page (1 2) of ___ Program Identification (75 76 77 78 79 80)

C

Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7 8)	Indicators: And (Not 9 10 11)	Indicators: And (Not 12 13 14)	Indicators: And (Not 15 16 17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field: Name (43-48)	Result Field: Length (49-51)	Result Field: Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic: Plus (54 55)	Resulting Indicators: Arithmetic: Minus (56 57)	Resulting Indicators: Arithmetic: Zero (58 59)	Resulting Indicators: Compare: 1 > 2	Resulting Indicators: Compare: 1 < 2	Resulting Indicators: Compare: 1 = 2	Resulting Indicators: Lookup(Factor 2)is: High	Resulting Indicators: Lookup(Factor 2)is: Low	Resulting Indicators: Lookup(Factor 2)is: Equal	Comments (60-74)
01	C																					
02	C																					
03	C																					
04	C																					
05	C																					
06	C																					
07	C																					
08	C																					
09	C																					
10	C																					
11	C																					
12	C																					
13	C																					
14	C																					
15	C																					
16	C																					
17	C																					
18	C																					
19	C																					
20	C																					
	C																					
	C																					
	C																					
	C																					
	C																					

(22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21)

*Number of sheets per pad may vary slightly.

This page is intentionally left blank.

Glossary

access method: A technique for moving data between main storage and input/output devices.

addrout file: A record address diskette file produced by the sort program. An addrout file contains the binary relative record numbers of records in a diskette file, and can be used to process the file designated as the input file to the sort program.

alphabetic characters: Letters and other symbols, excluding digits, used in a language.

alphabetic field: One or more alphabetic characters of related information in a record.

alphabetic shift: A control (attribute or key) for selecting the alphabetic character set in an alphameric keyboard.

alphameric characters: Same as alphabetic characters, with the addition of digits 0 through 9.

alphameric field: One or more alphameric characters of related information in a record. Any character that can be entered from the keyboard is valid in an alphameric field.

alternate collating sequence: A user-defined collating sequence that alters the standard EBCDIC collating sequence.

apostrophe: For 5280, this character (') is used to enclose character strings such as 'NUMBER'. Two consecutive apostrophe characters are coded to form an apostrophe in a character constant such as 'DRIVER''S LICENSE'.

array: The term used in RPG to refer to the function provided by tables in DE/RPG.

arithmetic expression: An expression that contains arithmetic operations and that can be reduced to a single numeric value. An arithmetic expression is evaluated from left to right with multiplication and division preceding addition and subtraction.

ASCII: (ANSI definition) American National Standard Code for Information Interchange. The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

attribute: A characteristic. For example, attributes of a data set include record length, label, and creation date. Attributes of a displayed field could include high intensity, reverse image, and column separators.

attribute byte: A control position which describes attributes to the system.

auto dup: Automatic duplication. 1. The process of automatically copying the contents of a field in a previous record into the corresponding positions of the current record. 2. The process of automatically verifying the contents of a field in the current record with the contents of the corresponding positions of a previous record.

auto record advance: Automatic record advance. A movement forward to the next sequential record without manual intervention when current record is completely entered and the AUTO REC ADV switch is on.

auto skip: Automatic skip. In enter mode, if the AUTO SKIP/DUP switch is on, the process of automatically filling an auto skip field with blanks and advancing to the next field. In verify mode, the process of verifying that all the positions in the field are blank. The DE/RPG program specifies which fields are auto skip.

auto verify: Automatic verify. In verify mode, auto dup fields are checked against the same fields in the previous record. See *automatic duplication,2.*

auxiliary duplication: The process of copying or verifying data from a named storage location into a field.

blank check: A check of a field to ensure that there are no blank characters (hex 40) in the field.

blank fill: The act of filling a field with blank characters (hex 40).

block: 1. A set of things, such as words, characters, or digits, handled as a unit. 2. A collection of contiguous records recorded as a unit. Each block can contain one or more records.

blocking: Combining two or more records into one block.

branch instruction: An instruction that changes the sequence in which the instructions in a computer program are executed. Execution of instructions continues at the address specified in the branch instruction.

buffer: 1. (ANSI definition) Storage or programming that compensates for a difference in rate of flow of data, or time of occurrence of events, when transmitting data from one part of a computer system to another. 2. An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written.

calculation: The source statement used for building subroutines.

calculation specifications sheet: An RPG II coding sheet used to describe calculation processing to be done by the program.

collating sequence: The position each character holds in relation to other characters according to the bit structure.

column separator: A display screen attribute that shows vertical lines preceding each position in a field on the display. These lines do not occupy positions on the display. For example |A|B|C.

command function keys: The 14 keys on the top row of the data station keyboard that are used with the command key to request functions.

comments: Words or statements in a program that serve as documentation rather than instructions to an assembler or compiler.

compilation: The execution of the DE/RPG compiler that processes source statements and produces an object program. The compiler allocates data areas for files, records, and fields used in the program.

compile: (ANSI definition) To prepare a machine language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one machine instruction for each symbolic statement, or both, as well as performing the function of an assembler.

compile-time table: A table whose data is supplied with the source program and that is compiled to be part of the object program.

compiler: 1. A program that translates a series of instructions, written in a programming language, into a program the system can execute. 2. (ANSI definition) A program that translates a source program written in a specific programming language into an object program.

constant: A data item that does not change during the execution of a program. This item represents itself and is actually used in processing rather than being a field name representing the data. For example, *cost* is a name representing a field containing data that changes. The constant 100 is actual data used that does not change.

continuation line: A DE/RPG source statement that extends the options field of a Z-specification or the editing field of an A-specification.

continuation character: A plus character (+) or a minus character (–) specified as the last nonblank character of a DE/RPG source statement line to indicate that the source statement is continued on the next line.

continuation line: A DE/RPG source statement that extends the options field of a Z-specification or the editing field of an A-specification.

copy: To read data from a source, leaving the source data unchanged, and to write the same data elsewhere in a physical form that may differ from that of the source.

copy file: Designated by the keyword CFILE on the Z-specification. The temporary storage of information read from a data set. This information is merged into the associated transaction file.

copy mode: The mode in which the operator is allowed to select records from a copy file to be copied into a transaction file.

copy search mode: The mode in which the operator is allowed to specify a record (by position) in a copy file to copy the record to a transaction file.

copy transfer mode: The mode during which records selected from a copy file are copied to the transaction file.

counter: A register or storage location used to accumulate the number of occurrences of an event.

CRT file: The collection and temporary storage of data from the entry process. This data is also shown on the display screen.

cursor: A movable horizontal line (underscore) on a display screen, used to indicate where the next character entered by the operator will appear. It blinks when no additional entry is allowed and the system is awaiting the Enter key.

data required: A field attribute that indicates an operator must enter at least one nonblank character into the displayed field.

data set: An organized collection of related data records treated as a unit and existing on a diskette. In other systems, this is sometimes referred to as a file.

data set label: A 128-byte area on the diskette index cylinder that describes a data set.

data set name: The name associated with a data set. The first character must be alphabetic, and the remaining characters can be any combination of alphabetic or numeric characters. Blanks cannot appear between characters in a name.

default value: A value automatically chosen by the system when a value is not specified by the user.

DE/RPG: Data entry with RPG subroutines. A 5280 program product that provides a means for writing data entry and application programs for the 5280 system.

device address: Two EBCDIC characters *such as D1) or four hex characters *such as 4C00) used to identify a 5280 input/output device such as a diskette drive, printer, or magnetic stripe reader.

direct access: The ability to obtain data from a storage device directly by key or relative record. Contrast with *sequential access method.*

direct access method: An access method for processing files by specifying the address (record number) of each record to be accessed.

diskette: A permanent storage medium used on 5280. 1. A thin, flexible magnetic disk permanently sealed in a cover that gives protection. 2. A single removeable disk contained in its own envelope.

diskette drive: The mechanism used to read and write diskettes.

display attributes: The characteristics assigned to a field or record that control the way the data is displayed (such as high intensity, reverse image, or column separators).

EBCDIC (extended binary-coded decimal interchange code): A character set containing 256 eight-bit characters.

edit: To modify the form or format of data; for example, to insert or delete characters such as page numbers or decimal points. See also *simple edit; extended edit.*

editing field: The field in A-specifications in which keywords and their parameters are coded.

Enter mode: The mode in which the operator initially enters data through a display station. Some editing and interaction may occur. See also *verify mode; update mode.*

entry format: The source statement that specifies the sequence of record entry for data entry jobs or the sequence of subroutine use during job execution.

execute: To cause an instruction, program, utility, or other machine function to be performed.

execute mode: The mode during which subroutines (built with calculation statements in the source program) are executed.

execution-time table: A table whose data is supplied in a data file when the program that uses the table is executed.

Field: One or more bytes of related information in a record.

field attribute: Display attributes (such as column separators and highlight) that appear in designated fields as controlled by the program.

field description: The source statement that describes the characteristics of a field and specifies the location of the field within the record.

field length: The number of positions allowed for a given field, determined by maximum length of information that will be entered in the field.

field separator: A blank character position preceding every field of an enter record. This position is required for the attribute byte.

file: The collection and temporary storage of related information within the 5280. A file is usually associated with a specific function and is named by either the function or the device supported. See *copy file, CRT file, print file,* and *transaction file.* Also see *data set.*

file description: The source statement that names a file, describes the file characteristics, and assigns the I/O device for the file.

fixed position prompt: A user-written message that appears on row 2 of the display.

format: A specific arrangement of information in a record or on a display screen.

format chaining: Defined sequence of formats to be used in entering, verifying, or updating a batch.

format control: The field edit and controls that determine the type of information that can be entered and the appearance of the information for a formatted display.

format ID: An identifier assigned to an entry format and used to select the format during both operator-controlled entry format selection during program execution.

format level: The identification associated with a format.

format 0 (zero): A format for display stations that allows entering information on an unformatted display.

hex: Hexadecimal. A number system using 16 symbols: 0–9, A–F each representing 4 bits (one-half byte).

identifier: A string of characters used to identify an item of data and possible to indicate certain properties of that data. Also see *mask.*

indexed data set: A data set in which the keys from another data set and their record position within that data set are recorded. When index data sets are used, the following access methods can be used: sequential; direct by relative record number; and direct by key value.

input record: A data record that is transferred to computer storage for processing.

insert field: A field not present in the enter record, but which will be inserted by the system and will be present in the output record.

job: For the 5280, a program and associated data that can be executed in a partition.

job specification: The source statement that names a job and specifies certain job characteristics.

key: 1. (noun) One or more characters included in a data record that are used to identify or control the use of that data. 2. (verb) To enter information from the keyboard.

key field: The field within a record that identifies that record when the direct access method or key value is used. The key and record location for each record in the data set are stored in the index when an index is used.

left adjust: (verb) To shift the contents of a register or field so that the character at the left end of the data is at the leftmost position in the register or field.

left justify: 1. The adjustment of positions of characters on a page so that the left margin of the page is regular. 2. The adjustment of positions of characters so that the leftmost character entered is at the extreme left of a field.

literal: Character data that is not associated with a name in a program.

logical record: A record independent of its physical environment. Portions of the same logical record may be located in different physical records, or several logical records or parts of logical records may be located in one physical record depending on the exchange type being used.

mandatory entry: A field attribute that indicates an operator must enter at least one character into the displayed field.

mark: To flag a record or field when it is known or suspected that the data in it is incomplete or incorrect.

marked record: A record that contains a flag indicating that one or more data fields in the record are incomplete or incorrect.

mask: A pattern of characters used to control the retention or elimination of another group of characters.

menu: A displayed list of items from which the operator makes a selection.

mode: The operational category of a station: enter, verify, update, or rerun.

modulus 10 and 11 self-checking: Formulas used to calculate the self-check digit for a self-check field. See *self-check field.*

nondisplay: A field attribute that prevents display of data. It can be used for fields containing confidential information.

numeric constant: Any combination of the numbers 0 through 9 including a decimal point and sign, if needed. Blanks cannot be included in a numeric constant.

numeric fields: A field that contains one or more numeric characters. Valid numeric characters are the digits 0–9 and + (plus sign), – (minus sign), . (decimal point), blank and , (comma).

numeric shift: A control (attribute or key) for selecting the numeric character set in an alphameric keyboard.

object program: 1. A set of instructions in machine language (object code). The object program is produced by the compiler from the source program. 2. In the 5280 the executable program produced by the DE/RPG compiler from a set of source statements. The object program can be executed to control the operation of the 5280 system to perform user-designed functions.

options field: The field in Z-specifications in which keywords and their parameters are coded.

output: Data delivered or ready to be delivered from a device or program, usually after some processing.

output data set: A data set containing the data that results from processing.

overflow indicator: An indicator that is set when a specified line is reached on a printer form. The indicator can be tested to allow totals to be printed at the bottom of a page and headings to be printed at the top of the next page.

pad: To fill unused positions in a field with dummy data, usually zeros or blanks.

parameter: Information coded in parenthesis directly following a keyword to define and control the function of the keyword.

partition: An area of storage in which only one program at a time can execute.

primary line: The first line of any source statement in a DE/RPG program. The primary line can be supplemented with continuation lines and/or secondary lines.

print file: The organization of data for formatted printing and for printer forms control.

print mode: The mode during which the current record is printed as it exists on the diskette.

production statistics: Statistics related to activities occurring during the key entry operation.

program: 1. (noun) A set of sequential instructions that tells the controller where to get input, how to process it, and where to put the results. 2. (verb) To design, write, and test computer programs.

prompt: (noun) A message issued by a program that requests either information or an operator action to continue processing.

record: A collection of related date, treated as a unit.

record description: The source statement that names a record and describes the attributes of the record.
record length:
record length: The number of words or characters (or bytes) forming a record.

relative record number: A number that specifies the location of a record in relation to the beginning of the data set.

repeat count: A count used to define the number of times a format is repeated.

rerun display mode: The mode during which the records in the data set assigned as the transaction file are automatically processed from the beginning of the data set through the last record.

review format: The source statement that specifies the characteristics of a record and the display format to be used during verify and update operations.

reverse image: The display attribute that causes characters to be displayed as dark characters on a light background.

review mode: The operating status during which operators can selectively display transaction data set records or can page from one transaction data set record to another.

right adjust: 1. (noun) The placement of data in a register or field, or the shipfting of the contents of a register or field, so that the least significant byte at the right end of the data is placed in the rightmost position of the register or field. See also *right justify.* 2. (verb) To shift the contents of a field so that the last character keyed is in the rightmost position of the field.

right justify: The adjustment of positions of characters so that the rightmost character entered is at the extreme right of a field.

search argument: The data to be compared to specific parts of each record.

secondary line: A DE/RPG source statement that allows additional test conditions or alternatives to be specified.

self-check digit: The rightmost or leftmost digit of a self-check field. See *self-check field.*

slef-check field: A field, such as an account number, consisting of a base number and a self-check digit. For data entry applications, the self-check digit entered by the operator is compared to the self-check digit computed by the system. If the operator makes a mistake when entering (keying) a self-check field, an error message is displayed.

sequence number: A number coded in the first five positions of a DE/PRG source statement line to indicate the position of the line in a set of source statements.

sequential by key: A method of data set processing that reads records in the order in which a keyed on indexed data set is arranged.

source entry program: A part of the DE/PRG Program Product that assists the user in entering DE/RPG source statements onto a diskette.

source program: A set of instructions that represents a particular job as defined by the programmer. These instructions are written in a programming language, such as DE/RPG.

source statements: User-coded statements used as input data to the DE/PRG compiler from which the compiler produces an object program.

spanned record: 1. A record that crosses a block boundary. 2. A record that is stored in more than one block.

special character: A character other than a digit, a letter, or #, $, and @. For example, *, +, and % are special characters.

status line: For the 5280, the first line on a display screen. This line provides operational information.

subroutine: A common group of instructions always returns control to the calling routine.

table: A collection of data in which each item is uniquely identified by its position relative to the other items.

table check: A check to determine whether data in an input field matches an entry in a table.

table description: The source statement that identifies a table to be used during program execution.

transaction: An item of business. Customer orders and customer invoices are examples of transactions. Transactions saved in a transaction data set are usually processed along with a master data set by programs. For example, in a payroll application, a transaction data set could indicate the number of yours worked by each employee and the master data set could indicate each employee's name and pay rate.

transaction data set: A data set that contains records associated with a specific transaction. These records are less permanent information, such as customer orders. Contrast with master data set.

transaction file: The collection and temporary storage of data records that result from the completion of each entry process. These records are then written to the associated data set.

truncate: To shorten a statement by reducing it to a predetermined length.

underscore: A field attribute of a display field that places a line under all positions of the field.

unformatted display: A display on which no field has been defined by the user.

update insert mode: The mode in which the operator is allowed to enter new records between two existing records in the data set assigned as the transaction file.

update mode: The mode in which the operator selects certain records for review and correction. See also *enter mode; verify mode.*

update search mode: The mode in which the operator is allowed to specify a record (by position) in the data set assigned as the transaction file and then operate as in update mode.

user exit: A point in an IBM program at which a user's routine may be given control.

variable: A character or group of characters (such as a name) that refers to a value that can be changed during the execution of a program.

verify: To determine whether a transcription of data or other operation has been accomplished accurately.

verify bypass field: A field that was entered, but does not need to be verified.

verify correct mode: The mode in which the operator is allowed to change the contents of the current record without the normal comparisons between the data in the record and the keyed data that occur during verify mode.

verify display mode: The mode in which the operator is allowed to view an entire record on the display without any verification or correction functions.

verify insert mode: The mode in which the operator is allowed to enter new records between two existing records in the data set assigned as the transaction file.

verify mark: An indication on a record or a batch that it has been verified.

verify mode: The mode in which the operator rekeys data from a source document that has already been keyed in order to check that the data has been entered correctly. See also *enter mode; update mode.*

verify search mode: The mode in which the operator is allowed to specify a record (by position) in the data set assigned as the transaction file for verification.

work file: An area on diskette that is reserved for temporary storage of data being processed. For example, the DE/PRG compiler uses two work files.

work space: A named area in a program that is reserved for the program's use. A work space cannot be displayed or have data entered into it by the operator.

zero fill: The action of filling a field with the numeric value zero.

zero suppress: The elimination of preceding zeros in a number. For example, 0057 becomes 57 when zero suppressed.

This page is intentionally left blank.

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*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name _____

Address _____

_____

● No postage necessary if mailed in the U.S.A.

Cut or Fold Along Line

Reader's Comment Form

Fold and tape | **Please Do Not Staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

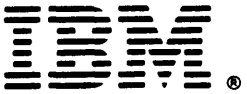
BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 , ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 506, Building 998
11400 Burnet Rd.
Austin, Texas 78758

Fold and tape | **Please Do Not Staple** | Fold and Tape



International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30055
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
(International)

SC21-7787-2

Printed in U.S.A.