

GX28-6400-2

X28-6400-2  
S 360-20

IBM System/360  
Time Sharing System

# TSS/360

*Quick Guide  
for Users*

**TERMINALS**

**COMMANDS**

**PL/I**

**FORTRAN**

**ASSEMBLER**

The IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font with horizontal stripes.

**Systems Development Division**

Printed in U.S.A. GX28-6400-2

INTERNATIONAL BUSINESS MACHINES CORPORATION  
Data Processing Division  
112 East Post Road  
White Plains, New York, 10601  
(U.S.A. only)

IBM WORLD TRADE CORPORATION  
821 United Nations Plaza  
New York, New York, 10017  
(International)

**IBM System/360  
Time Sharing System**



**TSS/360**  
**General-Purpose Operating System**

*Quick Guide  
for Users*

**TERMINALS**

**COMMANDS**

**PL/I**

**FORTRAN**

**ASSEMBLER**

---

Third Edition (June 1970)

This is a major revision of, and makes obsolete, X28-6400-1. This edition applies to Version 7, Modification 0, of IBM System/360 Time Sharing System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the specifications herein; before using this publication in connection with the operation of IBM systems, refer to the latest edition of *IBM System/360 Time Sharing System: Addendum*, Order No. GC28-2043, for the editions of publications that are applicable and current.

Text for this manual has been prepared with the IBM SELECTRIC® Composer.

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

A form is provided with this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Time Sharing System/360 Programming Publications, Department 643, Neighborhood Road, Kingston, New York 12401.

**TSS/360 System Reference Library**

Introducing TSS/360, GC28-2048  
Concepts and Facilities, GC28-2003  
Data Management Facilities, GC28-2056  
Assembler Language, GC28-2000  
Assembler User Macro Instructions, GC28-2004  
Assembler Programmer's Guide, GC28-2032  
IBM FORTRAN IV, GC28-2007  
FORTRAN IV Library Subprograms, GC28-2026  
FORTRAN Programmer's Guide, GC28-2025  
PL/I Reference Manual, GC28-2045  
PL/I Library Computational Subroutines, GC28-2046  
PL/I Programmer's Guide, GC28-2049  
Linkage Editor, GC28-2005  
Command System User's Guide, GC28-2001  
Manager's & Administrator's Guide, GC28-2024  
Operator's Guide, GC28-2033  
Independent Utilities, GC28-2038  
System Programmer's Guide, GC28-2008  
System Generation and Maintenance, GC28-2010  
Remote Job Entry, GC28-2057  
Multiterminal Task Programming and Operation, GC28-2034  
Terminal User's Guide, GC28-2017  
System Messages, GC28-2037  
Time Sharing Support System, GC28-2066  
Master Index, GC28-2023  
Quick Guide for Users, GX28-6400  
Quick Guide for System Programmers, GX28-6401  
Addendum, GC28-2043

**TSS/360 Program Logic Manuals**

System Logic Summary, GY28-2009  
Resident Supervisor, GY28-2012  
Task Monitor, GY28-2041  
System Service Routines, GY28-2018  
Dynamic Loader, GY28-2031  
Access Methods, GY28-2016  
Command System, GY28-2013  
Program Control System, GY28-2014  
Assembler, GY28-2021  
FORTRAN IV, GY28-2019  
FORTRAN IV Library, GY28-2020  
PL/I Compiler, GY28-2051  
PL/I Library Computational Subroutines, GY28-5052  
Linkage Editor, GY28-2030  
System Generation and Maintenance, GY28-2015  
Independent Utilities, GY28-2039  
On-Line Test Control System, GY28-2042  
System Control Blocks, GY28-2011  
Time Sharing Support System, GY28-2022  
Operator Task and Bulk I/O, GY28-2047



## Initiation Procedure—IBM 2741

1. Set terminal mode switch to LCL.
2. Set terminal power switch to ON.
3. Set margin stops at 0 and 130.
4. Set tab stops by using TAB key.
5. Press RETURN key to position typing element at left margin.
6. Set terminal mode switch to COM.
- 7A. Direct-wired terminals: press ATTN key to start LOGON.
- 7B. Dial-up terminals: press TALK button, lift receiver, dial time-sharing system; when continuous tone is heard, press DATA button and replace receiver.

Terminal is now operational; enter LOGON command

## Initiation Procedure—IBM 1052

1. Set panel switches:

<i>Switch</i>	<i>Setting</i>	<i>Position</i>
SYSTEM	ATTEND	up
PRINTER	SEND REC	middle
KEYBOARD	SEND	up
READER1	ON	up
STOP CODE	OFF	down
SYSTEM	PROGRAM	up
SYSTEM	—	up
TEST	OFF	down
SINGLE CY	OFF	middle
RDR STOP	OFF	middle

- Set all other switches to OFF or HOME positions.
2. Set margin stops at 0 and 130.
  3. Set tab stops by using TAB key.
  4. Turn on main-line switch; POWER light should come on. If necessary turn off the data check light by pressing the DATA CHECK pushbutton.
  - 5A. Direct-wired terminals: press ATTENTION/LINE RESET key to start LOGON.
  - 5B. Dial-up terminals: press TALK button, lift receiver, dial time-sharing system; when continuous tone is heard, press DATA button and replace receiver.

Terminal is now operational; enter LOGON command

## Initiation Procedure—Teletypewriter

1. Check paper supply.
2. Press ORIG button on control unit; the button lamp should light.
3. Dial tone should be heard; volume adjustment can be made with SPKR VOL control.
4. Dial system number. A high-pitched sound will be heard when connection with the computer is made.

Teletypewriter terminal is now operational; enter LOGON command

### Initiation Procedure—IBM 1056

1. Set AUTO EOB switch on card reader, as appropriate. ON causes end-of-block code to be sent automatically after a card is read or when an EOB code is detected on the card program tape.\*
2. Initialize at terminal, as described under "Initiation Procedure—IBM 1052."
3. To begin card reading:
  - A. Press EJECT button to clear any card that might be in card reader.
  - B. Place cards in hopper, face down, with 12-edge toward reading station. Cards *must* have an upper-left corner cut.
4. Type in the card-character transfer code C, CA, or CB to indicate SYSIN as the card reader and the character set as either EBCDIC (IBM 029 Card Punch) or PTTC/6 (IBM 1057 Card Punch). If the user types C, CA or CB before logging on, he must supply LOGON information on first card.
5. A single card can be read on 1052 by pressing READER START/LINR.
6. Press RETURN key. System checks card reader for input; cards will be read without user intervention.

\*If the 1056 has a program tape switch, set it to ON to indicate additional input data or control of card reading; set it to OFF to indicate that only cards will be used as input. If a program tape is used, open the right-hand side panel and place tape column 1 over the contact-roller center line.

### Terminal Procedures—IBM 2741 and 1052

#### Entering Line

1. Enter 1-130 characters.
2. Press RETURN key. Where TSS/360 has set length limits (e.g., 120 characters for line data sets) overlength records are rejected.

#### Continuation Lines

1. Enter 1-129 characters, followed by continuation character: hyphen.
2. Press RETURN key; enter continuation line

#### Canceling Line (before RETURN key has been pressed)

1. Enter line-kill character: #.
2. Press RETURN key or (for 1052 only) hold ALTN CODING key and push CANCEL key.

#### Canceling Line (after RETURN key has been pressed)

1. If line was part of data set, cancel line indirectly by canceling or modifying it in data set.
2. If line was command, it can be canceled only by attention interrupt that may or may not be effective.

#### Canceling Characters (before RETURN key has been pressed)

1. Press BACKSPACE key past incorrect characters, thereby erasing them.
2. Correct the line as indicated below or, if no correction is desired, press RETURN key

#### Correcting Line (before RETURN key has been pressed)

1. Backspace past incorrect characters, thereby erasing them.
2. Turn typewriter roller up 1 or 2 lines; maintain character alignment resulting from backspacing.
3. Type corrected characters
4. Press RETURN key

#### Terminal Shutdown

*Normal:* Issue LOGOFF command; when execution is completed, press terminal power switch to OFF.

*Emergency:* Press ATTN key (2741) or ATTENTION/LINE RESET key (1052). When system types underscore character, proceed with normal shutdown.

## Methods for Terminating Card-Reader Control

1. Run cards until hopper is empty; if in command mode, system will prompt 1052-terminal user with underscore character.
2. Supply input card with:  
col 1 = \_ K (keyboard will get control; PROCEED light will go on).
3. Press ATTENTION/LINE RESET key; system will then poll terminal for input (providing user has not taken control of attention interrupts).

## Resumption of Card-Reader Control

1. Type C, CA, or CB.
2. Press RETURN key.

## Terminal Procedures--Teletypewriter

### Entering Line

1. Enter 1-80 characters.
2. Terminate line by pressing CTRL key and X OFF key simultaneously. System will issue carrier return and line feed before prompting for additional input.

### Continuation Lines

1. Enter 1-80 characters, followed by hyphen.
2. Press RETURN and LINE FEED keys; then press CTRL and X OFF keys simultaneously.
3. Enter continuation line.

### Canceling Line (before end-of-line sequence)

1. Enter line-kill character: #
2. Press RETURN and LINE FEED keys; then press CTRL and X OFF keys simultaneously.
3. Retype correct line.

If end-of-line sequence has been used, cancel line indirectly (e.g., use MODIFY command).

### Correcting Line (before end-of-line sequence)

1. Press backspace key (←) the number of characters to be replaced.
2. Type in correct characters; continue entering line.

### Terminal Shutdown

1. Normal: Issue LOGOFF command; when LOGOFF is complete, press CLR on control unit.
2. Emergency: Press BREAK button; when system prompts with underscore character, follow normal termination procedure.  
If unable to terminate task from terminal, call system operator.



## Special Function Keys

**IBM 2741 and 1052, and teletypewriter**  
ATTN (2741), ATTENTION/LINE RESET (1052), BREAK  
(teletypewriter)—Generates attention interrupt to stop processing.

*System Response (for 2741 and 1052)*

␣ ! or \_

*System Response (for teletypewriter)*

\ ! or ] ←

*User Actions*

GO	resumes processing
ABEND	terminates task
REPEAT	repeats interrupted message
null (carriage return)	resumes processing
any other command	should be accepted and processed
attention key	press key and get system responses, five times without intervening commands, to terminate task ( for AETD routines only )

If user has his own interrupt-handling program, response is determined by that program.

RETURN (2741, 1052)—causes a typing-element return, line feed, and an end-of-transmission character. RETURN key ends every line of input from keyboard and marks defaults when replying to system messages. Keyboard is unlocked when system is ready for input lines.

BACKSPACE (2741, 1052)—cancels a line or corrects erroneous characters in a line. Backspace characters are edited out of input stream and do not appear in stored data; they are transmitted and included in 260-character maximum for line.

RESEND (1052)—used with associated light during block checking. Light comes on when end-of-block character is sent by terminal; turned off when receipt is acknowledged by system. If light remains on, or if it and DATA CHECK light are on, an error may be indicated. While RESEND is on, system will not accept input.

LINE FEED (1052)—moves paper up, according to line-space setting, without moving typing element.

CANCEL (1052)—cancels line, while ALTN CODING key is depressed before RETURN key has been used to indicate end-of-line. See "Terminal Procedures—IBM 2741 and 1052."

## Teletypewriter Control Unit—Buttons and Keys

ORIG	energizes terminal and dial tone
CLR	disconnects terminal from computer
LCL	places terminal in local mode; can be used as typewriter, without connection to computer
BUZ-RLS	silences paper-supply buzzer; light will remain on until more paper has been inserted
Telephone-type dial	dials in computer
OUT OF SERV	used when inserting paper or changing ribbon
NORMAL-RESTORE	used when inserting paper or changing ribbon

BRK-RLS	resumes keyboard operation after break signal; computer can transmit break signal and lock out teletypewriter; the BRK-RLS button, followed by K, unlocks keyboard
BREAK	generates attention interrupt; usually, system responds with underscore character
SPKR VOL	controls volume of speaker
CTRL	used in combination with function key; e.g., CTRL and X OFF must be pressed simultaneously for end-of-block signal
LOC LF	causes line feed at the teletypewriter without signaling computer
←	backspace to correct erroneous characters
TAB	for tabulation; must be pressed with CTRL
LINE FEED	moves paper up, according to line-space setting; part of end-of-line sequence
RETURN key	returns printing element to left margin; part of end-of-line sequence
X OFF	sends end-of-block character to computer; must be pressed with CTRL; part of end-of-line sequence
LOC CR	returns printing element to left margin
REPT	used with character key, causes repetition of character until key is released

### Error Light Indicators—IBM 1052

**RECEIVE ALARM**—Incorrect switch setting, more paper required, or paper not held down by roller.

**DATA CHECK**—Normally on when power is applied to keyboard; if both DATA CHECK and RESEND lights are on, redundancy-check error may have occurred; system will try to correct, if it was sending output to terminal or receiving input from card reader; otherwise, user must take corrective action. If error continues, system will terminate task because of communication line failure. *Action:* Press DATA CHECK and RESEND buttons to turn off lights and reenter line.

**RESEND**—(See DATA CHECK above.)

**PROCEED**—When out for abnormally long time, indicates equipment failure. *Action:* Try to key first letter of next line to be entered; if keyboard operates, light itself has failed. If DATA light on Data-Phone is not on, connection with system may be broken; if light is on, press ATTENTION/LINE RESET; then, if system does not print an underscore, request operator to terminate task.

**POWER**—If off, main-line switch not set to POWER ON, power cord not plugged in, indicator-light or equipment failure.

### Teletypewriter Error Light Indicators

DIAL lights during dial tone  
BY lights during busy signal  
NO CON lights when no connection (i.e., connection not established within specified time)  
SVC lights to indicate malfunction during call  
PA lights to indicate low paper supply  
red light lights to indicate end of line

#### Unused Keys and Lights

ANS	INCPT light	WRU	RU	FORM
TST	HERE	TAPE	BELL	RUB OUT
REST	ALT MODE	EOT	VT	

### Error Conditions—1056 Card Reader

1. Running out of cards or jamming: the system checks the keyboard. Proceed to:
  - A. Resupply cards in hopper
  - B. Type C, CA, or CB
  - C. Press RETURN key
2. Transmission error: DATA CHECK and RESEND lights go on, and system checks keyboard. Correct by using steps in A or B, below.
  - A. Correct information from terminal  
Press RESEND and DATA CHECK keys to turn off lights  
Type correct data  
Press EJECT key to stack error card  
Type C, CA, or CB  
Press RETURN key
  - B. Fix error card; insert corrected card  
Press EJECT key to stack error card  
Remove error card, correct it, and place corrected card in hopper as first card to be read  
Press RESEND and DATA CHECK keys on keyboard to turn off lights  
Type C, CA, or CB  
Press RETURN key

### Character Sets—IBM 2741 and 1052

Full EBCDIC character set is specified by using KA command.  
Folded EBCDIC character set is specified by issuing KB command;  
default: KB.

### Character Set—Teletypewriter

All EBCDIC upper-case letters.  
Special characters  $\square$   $\#$  are represented by /] [.  
All lower case EBCDIC numbers and special characters.  
No lower-case letters.

### Character Set—IBM 1056

The 1056 card punches (PTTC/8) are translated to upper- and lower-case EBCDIC characters. The card punches from 1057 and 029 are translated to internal EBCDIC code when read from 1056; 1057 punches must be read in CA mode and 029 punches in CB mode; punch codes for 029 and 1057, and related internal graphic representations, are in table later in this section.

LOWER CASE CHARACTERS

EBCDIC Graphic (1052/2741 or Printer)	Internal EBCDIC Hex Codes	TTY Graphic	IBM 1056 Card Reader			
			IBM 1057		IBM 029	
			Key- board	Punch	Key- board	Punch
.	4B	.	.	12-8-3	.	12-8-3
\$	5B	\$	\$	11-8-3	\$	11-8-3
,	6B	,	,	0-8-3	,	0-8-3
#	7B	#	#	8-3	#	8-3
@	7C	@	@	8-4	@	8-4
&	50	&	&	12	&	12
-	60	-	-	11	-	11
/	61	/	/	0-1	/	0-1
0	F0	0	0	0	0	0
1	F1	1	1	1	1	1
2	F2	2	2	2	2	2
3	F3	3	3	3	3	3
4	F4	4	4	4	4	4
5	F5	5	5	5	5	5
6	F6	6	6	6	6	6
7	F7	7	7	7	7	7
8	F8	8	8	8	8	8
9	F9	9	9	9	9	9
a	81	.....	a	12-1	a <sup>1</sup>	12-0-1
b	82	.....	b	12-2	b	12-0-2
c	83	.....	c	12-3	c	12-0-3
d	84	.....	d	12-4	d	12-0-4
e	85	.....	e	12-5	e	12-0-5
f	86	.....	f	12-6	f	12-0-6
g	87	.....	g	12-7	g	12-0-7
h	88	.....	h	12-8	h	12-0-8
i	89	.....	i	12-9	i	12-0-9
j	91	.....	j	11-1	j	12-11-1
k	92	.....	k	11-2	k	12-11-2
l	93	.....	l	11-3	l	12-11-3
m	94	.....	m	11-4	m	12-11-4
n	95	.....	n	11-5	n	12-11-5
o	96	.....	o	11-6	o	12-11-6
p	97	.....	p	11-7	p	12-11-7
q	98	.....	q	11-8	q	12-11-8
r	99	.....	r	11-9	r	12-11-9
s	A2	.....	s	0-2	s	11-0-2
t	A3	.....	t	0-3	t	11-0-3
u	A4	.....	u	0-4	u	11-0-4
v	A5	.....	v	0-5	v	11-0-5
w	A6	.....	w	0-6	w	11-0-6
x	A7	.....	x	0-7	x	11-0-7
y	A8	.....	y	0-8	y	11-0-8
z	A9	.....	z	0-9	z	11-0-9
blank	40	blank	.....	.....	.....	.....
.....	17 <sup>2</sup>	.....	.....	0-8-2	.....	0-8-2
.....	17	.....	.....	0-8-5	.....	12-0
.....	17	.....	.....	8-7	.....	11-0
.....	17	.....	.....	11-0	.....	.....
.....	17	.....	.....	12-0	.....	.....
.....	17	.....	.....	12-8-7	.....	.....

1. a-z not on keyboard; must be punched using multiple punch; print as A-Z at terminal during input.
2. No assigned graphics.

UPPER CASE CHARACTERS

EBCDIC Graphic (1052/2741 or Printer)	Internal EBCDIC Hex Codes	TTY Graphic	IBM 1056 Card Reader			
			IBM 1057		IBM 029	
			Key- board	Punch	Key- board	Punch
⌋	5F	⌋ <sup>1</sup>	.	12-8-1	⌋ <sup>5</sup>	11-8-7
!	5A	!	!	11-8-2	!	11-8-2
or ± <sup>10</sup>	4F	↑	,	0-8-1		12-8-7 <sup>6</sup>
>	6E	>	>	0-8-7	>	0-8-6
¢	4A	[ <sup>2</sup>	¢	11-8-7	¢	12-8-2
+	4E	+	+	12-8-6	+	12-8-6
-	6D	] <sup>3</sup>	- <sup>4</sup>	0-8-6	-	0-8-5 <sup>7</sup>
?	6F	?	?	12-8-2	?	0-8-7
)	5D	)	)	11-8-5	)	11-8-5
=	7E	=	=	8-6	=	8-6
° or < <sup>11</sup>	4C	<	°	12-8-4	<	12-8-4
:	5E	:	:	11-8-6	:	11-8-6
:	7A	:	:	8-2	:	8-2
%	6C	%	%	0-8-4	%	0-8-4
''	7D	''	''	8-5	''	8-5
*	7F	*	*	8-1	*	8-7 <sup>8</sup>
(	5C	*	*	11-8-4	*	11-8-4
A	4D	(	(	12-8-5	(	12-8-5
B	C1	A	A	12-0-1	A <sup>9</sup>	12-1
C	C2	B	B	12-0-2	B	12-2
D	C3	C	C	12-0-3	C	12-3
E	C4	D	D	12-0-4	D	12-4
F	C5	E	E	12-0-5	E	12-5
G	C6	F	F	12-0-6	F	12-6
H	C7	G	G	12-0-7	G	12-7
I	C8	H	H	12-0-8	H	12-8
J	C9	I	I	12-0-9	I	12-9
K	D1	J	J	12-11-1	J	11-1
L	D2	K	K	12-11-2	K	11-2
M	D3	L	L	12-11-3	L	11-3
N	D4	M	M	12-11-4	M	11-4
O	D5	N	N	12-11-5	N	11-5
P	D6	O	O	12-11-6	O	11-6
Q	D7	P	P	12-11-7	P	11-7
R	D8	Q	Q	12-11-8	Q	11-8
S	D9	R	R	12-11-9	R	11-9
T	E2	S	S	11-0-2	S	0-2
U	E3	T	T	11-0-3	T	0-3
V	E4	U	4	11-0-4	U	0-4
W	E5	V	V	11-0-5	V	0-5
X	E6	W	W	11-0-6	W	0-6
Y	E7	X	X	11-0-7	X	0-7
Z	E8	Y	Y	11-0-8	Y	0-8
blank	E9	Z	Z	11-0-9	Z	0-9
	40	blank	.....	.....	.....	.....

1. ⌋ is used as NOT sign ( ¬ ); it is upper-case L
2. [ is upper-case K
3. ] is upper-case M
4. ⌋ prints at terminal during input
5. ¢ prints at terminal during input
6. Nothing printed at terminal during input; use multiple-punch 0-8-1 to print | (OR) at terminal during input

7. Nothing printed at terminal during input; use multiple-punch 12-8-1 to print  $\lrcorner$  at terminal during input
8. Nothing printed at terminal during input; use multiple-punch 8-1 to print  $\rceil$  at the terminal during input
9. A-Z print as lower-case letters at terminal during input
10. Vertical bar on 1052 and 2741, except 2741 correspondence terminal, on which it is the plus-or-minus sign ( $\pm$ )
11. Degree sign on 2741 correspondence terminal is folded into less-than sign

### Functional Character Sets

Function characters	EBCDIC equivalent	Punch codes 029/1057
<b>IBM 2741/1052</b>		
TAB	05	
SHIFT (up) <sup>1</sup>	36	
SHIFT (down) <sup>1</sup>	06	
BACKSPACE <sup>2</sup>	16	
RETURN (new line)	15	
LINE FEED	25	
BYPASS (stop printer)* <sup>3</sup>	24	
RESTORE (start printer)* <sup>3</sup>	14	
EOA (end of address)* <sup>3,4</sup>	7B	
EOB (end of block)* <sup>3,5</sup>	26	
EOT (end of transmission) <sup>3,8</sup>	37	
PREFIX* <sup>3,6</sup>	27	
CANCEL* <sup>3,7</sup>	none	
RDR STOP* <sup>3</sup>	35	
<b>Teletypewriter</b>		
EOT (punch off)	04	
DCA (idle, not used)	17	
PN (punch on)	34	
TAPE ON (punch on)	34	
LF (line feed)	25	
CR (carrier return)	0D	
X OFF (end of transmission)	13	
$\leftarrow$ (backspace)	16	
<b>IBM 1056 Card Reader</b>		
punch off	04	12-9-1
horizontal tab	05	12-9-5
lower case	06	12-9-6
delete	07	12-9-7
restore	14	11-9-4
new line (carrier return and line feed)	15	11-9-5
backspace	16	11-9-6
idle	17	11-9-7
blank	40	blank
bypass	24	0-9-4
line feed	25	0-9-5
end of block	26	0-9-6
prefix	27	0-9-7
punch on	34	9-4
reader stop	35	9-5
upper case	36	9-6
end of transaction	37	9-7

\* Applies to 1052 only

1. For translation; not kept in lines entered from terminal
2. To delete and replace characters in input line; not retained in lines entered from terminal

3. (1052 only) entered by pressing and holding ALT N CODING key, then pressing appropriate character key
4. Prints as # ; not normally used with TSS/360
5. Nonprinting; usually originated automatically from 1052 terminal
6. Used in terminal-component-selection codes; not normally used
7. Prints as -; causes cancellation by transmitting parity error; not normally used in TSS/360
8. Nonprinting; usually originated automatically from 2741 terminal

### Task Management

LOGON	identify user to system
ZLOGON	user with written identity procedure
BEGIN	logon to MTT application program
TIME	terminate execution after time interval
EXECUTE	initiate nonconversational task
SECURE	reserve private volumes for nonconversational tasks
BACK	change conversational task to nonconversational
CANCEL	stop execution of nonconversational task
ABEND	abnormally terminate task processing and restart
ABENDREG	display register contents following an abend
USAGE	print out user statistics
EXHIBIT	display BWQ activity or user task activity
LOGOFF	terminate task processing

### Data Management

CATALOG	catalog private data set characteristics
CLOSE	close user data sets
EVV	catalog private VAM data sets by volume
DDEF	define data set characteristics to system
RET	change catalog attributes of VAM data set
RELEASE	release private devices
CDD	execute prestored DDEF commands
DELETE	uncatalog private data sets
ERASE	uncatalog and free space of disk data sets
PERMIT	authorize user to share data set
SHARE	share data set belonging to other user
DSS?	present status of cataloged data sets
PC?	present status of cataloged data sets
POD?	describe members of partitioned data set
DDNAME?	list DDNAMES
JOBLIBS	manipulate DDNAMES
VT	high-speed copy, VAM data sets to tape
TV	high-speed restore, tape data sets to VAM
VV	high-speed copy, VAM data sets to VAM
CDS	copy data set

### Character Set Selection

K	input from keyboard
KA	input from keyboard with full character set
KB	input from keyboard with lower-case character folded
C	input from 1056 card reader
CA	input from 1056 card reader with full character set
CB	input from 1056 card reader with folded character set

### Language Processing

ASM	assemble
FTN	FORTRAN compile
PLI	PL/I compile
LNK	link edit modules

GN28-3172 9/15/70 15

**COMMANDS**



**Program Control**

LOAD load module into storage  
 UNLOAD unload module from storage  
 CALL pass parameters and execute module  
 RUN execute module  
 GO resume interrupted-program execution  
 STOP stop module execution  
 BRANCH continue executing at different location of module  
 AT ✓ prepare for dynamic control of executing module  
 REMOVE (X) remove effects of AT  
 IF provide logical control of commands  
 SET change value of data or code  
 DISPLAY display data or code on SYSOUT  
 DUMP put displayed data in data set for subsequent printing  
 QUALIFY identify module name to system

**Command Creation**

PROCDEF define user written command  
 BUILTIN identify module as command processor

**Profile Management**

DEFAULT specify change values of defaults  
 PROFILE change values in user profile  
 SYNONYM change names of commands and operands

**Text Editing**

EDIT prepare system to edit VISAM data sets  
 END end editing process  
 REGION specify data set region to be edited  
 ENABLE stop keeping history of data set changes  
 DISABLE keep history of data set changes  
 POST stop keeping history of data set changes  
 STATE reverse effects of changes using history data  
 CONTEXT replace character string by another  
 LOCATE locate character string  
 CORRECT correct characters within line  
 REVISE delete old lines and insert new lines sequentially  
 UPDATE insert lines anywhere within data set  
 EXCERPT insert lines from another data set  
 EXCISE delete lines  
 INSERT add new lines sequentially  
 NUMBER renumber lines  
 LIST print lines on SYSOUT

**Data Editing**

DATA create VSAM or VISAM data set  
 LINE? print line data sets on SYSOUT  
 MODIFY modify VISAM data set

**Bulk Output**

DMPRST performs a time-shared dump or restore of VAM2 volumes  
 PRINT print data set on high-speed printer  
 WT write tape formatted for high-speed printing  
 PUNCH punch data set into cards

**Message Handling**

EXPLAIN provide explanatory material for messages  
 PRMPT generate, exchange, or change messages

### System Programmer Commands

CPS*	clean up public storage
CVV	catalog public VAM volume
EVV	enter VAM volumes
LPDS*	list public data sets
NEWMSG	new updates for messages
PATCLEAR	performs time-shared initialization of VAM2 disks
PATFIX	fix page assignment table
RPS	create public volume from private volume
UPDTUSER	update user table

### Linkage Editor Statements

TRAITS	
COMBINE	summary of these statements and their formats follow
INCLUDE	command formats
RENAME	
END	

\* for TSS\*\*\*\*\* userid only

## Command Specifications

**Format** — command name followed by at least one blank or tab character, followed by one or more operands delimited by commas or tab characters; operand field may be blank

**Command Statements** — One or series of commands, separated by semicolons, read as one SYSIN record; comments delimited by apostrophes can be placed before, within, or after command statements

### Types of Statements

- Dynamic** — statement containing AT command followed by BRANCH, GO, DISPLAY, DUMP, GO, IF, SET, or STOP
- Immediate** — statement containing no AT command; executed when entered
- Conditional** — statement containing IF command

## Data Set Modifications

User may modify data sets when using MODIFY, DATA, FTN, ASM, LNK and text editing commands.

### MODIFY

	Type
#	system-prompt character
line no., data	modify, correct, or enter new line
D, line no., last line no.	delete line or range of lines
R, line no., last line no.	review lines
%E	end modification

### DATA

	Type
# (for VSAM)*; line no. (for VISAM)	system-prompt character
% line no., data	modify, correct, or enter new line
% D, line no., last line no.	delete line or range of lines
%E	end modification

\*VSAM data sets cannot be modified

### FTN, ASM or LNK

	Type
#	System prompt character
Line no, data	Modify, correct, or enter new line
D, line no, last line no	Delete line or range of lines
Carriage return	End modification

### Text Editing

Text editor can be used to modify, correct, delete, review, and update VISAM data sets (see "Text Editing" commands).

## Program Control Commands (General Information)

### Expressions

Type	Operator	Meaning
Arithmetic	+	addition
	-	subtraction
	*	multiplication
	/	division
Logical	¬	logical inversion or negation
	&	logical intersection
		logical union
Relational	>	greater than
	<	less than
	=	equal to
	>=	greater than or equal to
	<=	less than or equal to
	¬=	not equal to
	¬>	not greater than
	¬<	not less than

## Variables

Variables can be indicated with internal or external symbolic names, hexadecimal locations, register numbers, or dynamic statement counter.

<b>Types</b>	<b>Examples</b>	
<b>External symbolic names</b> — symbols referenced at load or execution time	FORTRAN module name CSECT name PSECT name module entry point FORTRAN blank common	PGM PGM # C PGM # P PGM # E &COM
<b>Internal symbolic names</b> — symbols referenced during single assembly or compilation; they can only be referenced if internal symbol dictionary was requested	FORTRAN statement numbers data names symbols defined by ASM statements unnamed assembler language CSECT FORTRAN blank common subscripted symbols generalized form SYMBOL.(OFFSET,LENGTH) where offset is expression, length is integer	5(1) %CSECT &COM A(I,J)
<b>Hexadecimal locations</b> — hexadecimal address in user's virtual storage enclosed in apostrophes and preceded by L	L'value' L'B000'	
<b>Register numbers</b> — general registers, indicated as nR, where n = 0-15 single-precision floating point registers, indicated as nE, where n = 0,2,4, or 6 double-precision floating point registers, indicated as nD, where n = 0,2,4, or 6	3R 4E 6D	
<b>Dynamic statement counter</b> — % number of times dynamic statement has been executed		

## Constants

Types	Examples
<b>Integers</b> — signed decimal integers	-647 +1066
<b>Character</b> — letters, decimal digits, and special characters, enclosed in apostrophes	'\$3.98' 'HOW'
<b>Hexadecimal</b> — one or more hexadecimal digits enclosed in apostrophes, preceded by X	X'1234' X'9FEC3'
<b>Floating point</b> — signed or unsigned decimal number with or without decimal point or exponents specified as E — single-precision D — double-precision	3.141 3141.59E-1  31.20D+3 31E-5
<b>Address</b> — character A followed by symbol enclosed in apostrophes; may be internal, external, or subscripted	A'symbol'
<b>Definition of Operand Terms</b>	<b>Examples</b>
<b>Data locations</b> — specified as symbols, hexadecimal locations, registers, or dynamic statement counter, in this format: - SYMBOL.(OFFSET,LENGTH), where offset is integer, length is expression	Y.(X'EDC',4)
<b>Data field</b> — contiguous group of storage to be dumped or displayed; indicated as first location:last location	FLDA:FLDB 0:4R 0:5E 6:2D

### Command Instruction Set

<u>Operation</u>	<u>Operands</u>	<u>Comment</u>
ABEND	none	
ABENDREG	none	
ASM	NAME=object module name [,STORED={ Y } { N }] [,MACROLIB=(symbolic ddname, index portion ddname)] [,VERID=version identi- fication] [,ISD={ Y } { N } ]  [,SYMLIST= { Y } { N } ]  [,ASMLIST= { Y } { N } ]  [,CRLIST= { Y } { N } ]  [,STEDIT= { Y } { N } ]  [,ISDLIST= { Y } { N } ]  [,PMDLIST= { Y } { N } ]  [,LISTDS= { Y } { N } ]  [,LINC=(first line number, increment)]	Y-module prestored; N-not prestored  default: only system library used default: listing and mod- ule are time stamped Y-produce internal symbol dictionary N-do not  Y-produce symbolic list- ing; N-do not  Y-produce object pro- gram listing; N-do not Y-produce cross-refer- ence listing; N-do not Y-produce edited sym- bol table; N-do not Y-produce ISD listing; N-do not  Y-produce program module dictionary; N-do not Y-store all requested listings as list data set; N-print all requested listings on SYSOUT defaults: Y if conv'l; N if nonconv'l ignored if STORED=Y default: (100, 100)
AT	instruction location [...]	
BACK	DSNAME=data set name	
BEGIN	application name [,any application defined parameters]	
BRANCH	INSTLOC=instruction location	
BUILTIN	NAME=command name [,EXTNAME=bpkd macro name] [,DSNAME=dsname]	
CALL	[DSNAME=entry point [,module parameters]]	default: last module referenced by system  Note: for PL/I, specify only module name or subroutine name; no procedure names.
CANCEL	BSN=batch sequence number	
C	none	
CA	none	
CATALOG form 1	NAME=current data set name [,STATE= { N } { U } ]	N-new; U-update

(continued)

21

	[,ACC= { R } { U } ]	R-read only access; U-unlimited
	[,NEWNAME=new data set name]	<u>default</u> : no name change
CATALOG form 2	GDG=generation data group name GNO=number of generations	keyword must be specified
	[,ACTION= { A } { O } ]	A-all generations removed; O-only last
	[ERASE= { Y } { N } ]	Y-erase old generations N-save
CB	none	
CDD	DSNAME=data set name	
	[ { data definition name (data definition name) ,... } ]	<u>default</u> : all referenced DDEFs
CDS	DSNAME1=current data set name [(member name)] DSNAME2=new data set name [(member name)]	
	[ERASE= { Y } { N } ]	Y-erase N-save
	[ { BASE=first line no. [,INCR=increment] [,REPLACE= { R } { I } ] } ]	<u>default</u> : no renumbering <u>default</u> : 100
CLOSE	[DSNAME=data set name] [.TYPE=T] [.DDNAME=data definition name]	<u>default</u> : all user data sets, except USERLIB, are closed default: normal close default: DSNAME speci- fication closed
CONTEXT	[N1=starting position] [N2=ending position]	<u>default</u> : if N2 specified, CLP; otherwise, first line <u>default</u> : last line when N1 not specified; otherwise N1
	.STRING1=search string [,STRING2=replacement string]	<u>default</u> : null string
CORRECT	[N1-starting line] [N2=ending line] [,SCOL=first column] [,CORMARK=replacement correction characters]	<u>default</u> : CLP <u>default</u> : N1 <u>default</u> : position 0 *-duplicates above and to right \$-duplicates above; characters on right replace @-duplicates above; characters on replace- ment line replace %-removes above charac- ter #-replaces nonconforming hexadecimal character
	[.CHAR= { C-character { M-mixed { H-hexadecimal } } ]	<u>default</u> : C

CPS*	VOLUME=vqlserno [START= {CONT DSCB address}]	default: beginning of specified volume
CVV*	VOLUME=volserno [START= {CONT DSCB address}]	for VAM data sets default: beginning of specified volume
DATA	DSNAME=data set name [(member name)]  [.RTYPE= { I {  BASE=first line number. } LINE {   INCR=increment   } FTN CARD S }]	I=indexed. default: VSAM default: for BASE, 100; for INCR, 100
DDEF	DDNAME= { data definition name } PCS OUT  [.DSORG= { VI VS VP }]	default: VI
	.DSNAME=data set name [(member name)]	
DDNAME?	[JOB LIB= { Y N }]	default: entire JFCB chain displayed
DEFAULT DELETE	{ operand=[value] } [...] [DSNAME=data set name]	default: individual data sets presented for disposition
DISABLE DISPLAY	none data field name [...]	
DMPRST	FROMDEV= { 2311 2314 2400 }  ,FRVOLID=volume identi- fication  ,TODEV= { 2311 2314 2400 }  [.TOVOLID= { volume identification } PRIVATE ]	if defaulted, command canceled  if defaulted, command canceled  if defaulted, command canceled
	[.NEWVLID=volume identi- fication ]	ignored if TODEV is 2400
	[WRITCHK= { YES NO }]	ignored if TODEV is 2400
	[.LABEL= { RETAIN NO }]	ignored if TODEV is 2400
	[IPL= { RETAIN NO }]	ignored if TODEV is 2400

\* Use restricted to privileged programmers

(continued)

23

GN28-3172 9/15/70



	,RUNMODE= {BACK } {FORE }	ignored if task is nonconversational
DSS?	[ NAMES= { (data set name ) [,...] } ]	<u>default</u> : all of user's data sets
DUMP	data field name [...]	
EDIT	DSNAME=data set name {(member name)}	<u>default</u> : USERLIB <u>default</u> : no member
ENABLE	none	
END	none	
ERASE	DSNAME=data set name {(member name)}	<u>default</u> : individual data sets presented for disposition
EVV	[ DEVICE=datatype { 2311 } { 2314 } ]  ,VOLUME=(volume identi- fication [...]) [,USERID=user identifi- cation]	1-6 decimal digits vol- ume serial number for o-authority pro- grammer only; 8 alphanumeric char- acters, first alpha- betic; pad with * as required on right side <u>default</u> : current user ID
EXCERPT	DSNAME=data set name {(member name)} [,RNAME=region name] [,N1=starting line [,N2=ending line]	
EXCISE	[N1=starting line] [,N2=ending line]	<u>default</u> : CLP <u>default</u> : N1
EXECUTE	DSNAME=data set name	
EXHIBIT	OPTION= { UID [ .TYPE= { ALL CONV BACK UID.userid } ] } { BWQ [ .TYPE= { ALL BSN.number } ] }	
EXPLAIN	{ MSGIDD ORIGIN WORD TEXT RESPONSE } [message identification]	<u>default</u> : preceding mes- sage or explainable words explained
FTN	NAME=module name [,STORED= { Y } { N } ] [,VERID=version identi- fication] [,ISD= { Y } { N } ]  [,SLIST= { Y } { N } ]  [,OBLIST= { Y } { N } ]	Y-prestored; N-not prestored <u>default</u> : module and list- ing time-stamped Y-produce internal sym- bol dictionary; N-do not Y-produce source pro- gram listing; N-do not Y-produce object program listing; N-do not

	[,CRLIST={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-produce cross reference listing; <u>N</u> -do not
	[,STEDIT={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-produce edited symbol table; <u>N</u> -do not
	[,MMAP={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-produce memory map; <u>N</u> -do not
	[,BCD={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-input contains BCD; <u>N</u> -does not
	[,PUBLIC={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-public CSECT attribute; <u>N</u> -not
	[,LISTDS={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-store all requested listings as list data set; <u>N</u> -print all requested listings on SYSOUT defaults: Y if conv'l; N if nonconv'l
	[,LINCR=(first line number, increment)]	ignored if STORED=Y <u>default</u> : (100, 100)
GO	none	
K	none	
KA	none	
KB	none	
IF	condition	
INSERT	[N1=preceding line number] [,INCR=increment]	<u>default</u> : CLP <u>default</u> : 100
JOBLIBS	DDNAME=data definition name	
LINE?	DSNAME=data set name [(member name)]	
	[{line number, (first line number, last line number)} [,...]]	<u>default</u> : entire data set
LIST	[N1=starting position CLP LAST] [,N2=ending position LAST]	<u>default</u> : When N2 is specified, CLP; otherwise first line <u>default</u> : When N2 is specified, N1; otherwise last line
	[,CHAR={C-character, H-hexadecimal, M-mixed}]	<u>default</u> : C
LNK	NAME=module name	
	[,STORED={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-prestored; <u>N</u> -not prestored
	[,LIB=data definition name of library]	<u>default</u> : last mentioned library
	[,VERID=version identification]	<u>default</u> : listing and module time-stamped
	[,ISD={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-produce internal symbol dictionary; <u>N</u> -do not
	[,PMDLIST={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-produce internal symbol dictionary listing; <u>N</u> -do not
	[,LISTDS={ $\begin{matrix} Y \\ N \end{matrix}$ }]	Y-store all requested listings as list data set; <u>N</u> -print all requested listings on SYSOUT defaults: Y if conv'l; N is nonconv'l

(continued)

25

	[,LINCR=(first line number, increment)]	ignored if STORED=Y <u>default:</u> (100, 100)
LOAD	[NAME=entry point name]	<u>default:</u> N <u>default:</u> last module referenced by system
LOCATE	[N1=starting position]	<u>default:</u> When N2 speci- fied, CLP; otherwise first line
	[,N2=ending position]	<u>default:</u> When N1 speci- fied, N1; otherwise last line
	[,STRING=search string]	
LOGOFF	none	
LOGON	user identification	
<i>trailing commas unnecessary</i>	.[password]	can only be defaulted nonconversationally specified as 24 when 24- bit addressing is needed for a task running on 32-bit machine
	.[addressing]	<u>default:</u> on 24-bit ma- chine, 24; on 32-bit machine, 32
	.[charge number]	<u>default:</u> will be found by system
	[ csect packing $\left\{ \begin{matrix} A \\ P \\ O \\ X \\ N \end{matrix} \right\}$ ]	A-all CSECTs and PSECTs P-PSECTs only O-private CSECTs only X-all CSECTs, no PSECTs N-no packing <u>default:</u> N
	.[maximum storage]	1-5 decimal digits; <u>default:</u> lesser of limits assigned at SYSGEN or JOIN time
	.[pristine $\left\{ \begin{matrix} P \\ X \end{matrix} \right\}$ ]	P-USERLIB opened; not used to form profile X-USERLIB not opened; not used to form profile <u>default:</u> USERLIB opened and used to form profile
LPDS*	VOLUME=volserno	
	[,START= $\left\{ \begin{matrix} CONT \\ DSCB \text{ address} \end{matrix} \right\}$ ]	<u>default:</u> beginning of speci- fied volume
MODIFY	SETNAME=data set name [(member name)] [,CONF=R]	R-review <u>default:</u> no review
	[,LRECL=record length ,KEYLEN=key length ,RKP=relative key position ,RECFM= $\left\{ \begin{matrix} V \\ F \end{matrix} \right\}$ ]	<u>default:</u> 132 <u>default:</u> 7 <u>default:</u> 4 if V, 0 if F V-variable; F-fixed
	[,FTN= $\left\{ \begin{matrix} Y \\ N \end{matrix} \right\}$ ]	Y=Fortran translation function required N=no translation required
NEWMSG	none	



PRINT DSNAME=data set name  
 [ ,STARTNO=starting byte position ] default: first byte  
 [ ,ENDNO=ending byte position ] default: last byte  
 [ ,PRTSP= { EDIT } ] When EDIT specified, HEADER, LINES, and PAGE must not be specified  
 [ ,HEADER=H ] default: no header  
 [ ,LINES=lines/page ] default: 54  
 [ ,PAGE=P ] default: no pages numbered  
 [ ,ERASE= { Y } ] default: N; Y=erase; N=save  
 [ ,ERROROPT= { ACCEPT } ] default: END  
 [ ,SKIP ]  
 [ ,END ]  
 [ ,FORM=paper form ] default: installation defined  
 [ ,STATION=station id ] default: terminal id in task common  
 [ ,TAPOPT= { AC - ASCII char } ] default: EC  
 [ ,AD - ASCII dump ]  
 [ ,AE - ASCII edit ]  
 [ ,ED - EBCDIC dump ]  
 [ ,EC - normal proc ]

PRMPT MSGID=message identification  
 [ ,INSERTn=inserted characters [...] ] default: no characters inserted

PROCDEF NAME=procedure name  
 PROFILE [ ,CSW= { Y } ] Y-save command symbols; N-do not  
 [ ,N ]

PUNCH DSNAME=data set name[,]  
 [ ,STARTNO=starting byte position ] default: first byte  
 [ ,ENDNO=last byte position ] default: last byte  
 [ ,STACK= { 1 | 2 | 3 | EDIT } ] default: 1  
 [ ,ERASE= { Y } ] default: N; Y=erase; N=save  
 [ ,N ]  
 [ ,FORM=card form ] default: installation defined

QUALIFY MNAME=[link edited module name.] object module name

REGION [ ,RNAME=region name ] default: null string

RELEASE DDNAME=data definition name  
 [ ,DSNAME=data set name ]  
 [ , { SCRATCH } ]  
 [ , { HOLD } ]

REMOVE statement number[,...]

RET           DSNAME=data set name

.RET= { P } { L } { U }  
          { T } { C } { R }

                  P=permanent } default:  
                  storage:        } P  
                  T=temporary }  
                  storage         } default:  
                  L=erase at     } L, if T  
                  LOGOFF         } specified;  
                  C=erase at     } no value  
                  CLOSE          } if P  
                  U=read-write } default:  
                  access         } U  
                  R=read-only }  
                  access         }

REVISE        [N1=starting line]        default: CLP  
                  [,N2=ending line]      default: N1  
                  [,INCR=increment]      default: 100

RPS\*          [VOL=volume serial no.]    default N/A if MVDS  
  specified

                  [ ,UNIT=type { 2311 }  
  { 2314 }  
  { 9 } ]        default: type set at  
  SYSGEN

                  [ ,OPT= { multivolume tape }  
  ddname                            } tape to public storage  
  multivolume public            } public storage to tape  
  data set - MVDS                }



	[,ACV=volserno]	valid for OPT=MVDS only; <u>default</u> : currently mounted ACV volume
	[,START={ CONT DSCB address fileseqno }]	<u>default</u> : beginning of speci- fied volume
RUN	[LOC=entry point name]	<u>default</u> : performs as GO
SECURE	{ (TA=number of tape devices [,type of tape device]) (DA=number of direct access devices [,type of direct access device]) }	
SET	{data location=value} { [,...]	
SHARE	DSNAME=data set name ,USERID=owner's user identi- fication	
	[,OWNERDS={ owner's dsname *ALL }]	<u>default</u> : *ALL
STET	none	
STOP	none	
SYNONYM	{ term=[value] } { [,...]	<u>default</u> : SYSGEN value
TIME	[MINS=minutes]	
TV	DSNAME1=tape data set name [,DSNAME2=vam data set name]	
UNLOAD	[NAME=entry point name]	<u>default</u> : last module ref- erenced by system
UPDATE	none	
UPDTUSER*	none	
USAGE	[USERID=user identification]	3-8 characters <u>default</u> : current user ID
VT	DSNAME1=vam data set name [,DSNAME2=tape data set name]	
VV	DSNAME1=current data set name [,DSNAME2=new data set name]	
WT	DSNAME=current data set name ,DSNAME2=tape data set name [,VOLUME=tape volume number [,FACTOR=blocking factor] [,STARTNO=starting byte position [,ENDNO=ending byte position]	<u>default</u> : scratch tape <u>default</u> : 30 <u>default</u> : first byte <u>default</u> : last byte

\*Use restricted to privileged programmers

(continued)



$$\left[ \text{,PRTSP} = \left\{ \begin{matrix} \text{EDIT} \\ \left\{ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \right\} \end{matrix} \right\} \right]$$
 when EDIT specified  
 HEADER, LINES and  
 PAGE must not be  
 specified  
 default: no header  
 default: 54  
 default no numbering

$$\left[ \text{,ERASE} = \left\{ \begin{matrix} Y \\ N \end{matrix} \right\} \right]$$
 default: N; Y=erase,  
 N=save

ZLOGON none

**Implicit Operands**

Values of these operands built into user profile can be changed with DEFAULT command; operands control user's operating environment.

<u>Operand</u>	<u>Function</u>	<u>Default Value</u>	<u>Other Values</u>
LIMEN	message severity	W - warning messages	I - information X - serious error T - terminate error
BREVITY	message length	T - standard message, no ID	M - message ID E - extended S - standard message
SYSIN	input source	K - keyboard	C - card reader
SYSINX	PROCDEF, EDIT, DATA, MODIFY and PLI input source	G - terminal	E - source list or terminal
TRANTAB	transaction table	N - inactive	Y - active
LINENO	line number prompting	Y - will prompt	N - will not prompt
ALPHABET	character set	1 - folded mode	2 - full EBCDIC 3 - PTTC/6 4 - PTTC/8
DEPROMPT	prompt during ERASE/DELETE	Y - prompt for disposition	N - will not prompt
LPCXPRSS	express mode	N - not active	Y - active
REGSIZE	maximum region name length	0	1 - 247

DDEF Command or Macro Instruction  
DDEF (for new data sets)

Operands	When Applicable					
	Direct Access				Private	
	Public Storage		Private		BSAM or QSAM	
	VAM	JOBLIB	VAM	JOBLIB	DA	TA
DDNAME <sup>1</sup> =data definition name [.DSORG <sub>i</sub> =data set organization]	X	X	X	X	X	X
.DSNAME=symbol [.DCB={,BUFOFF=absexp,DEN=integer,...}]	VS		VS		PS	PS
	VI	VP	VI	VP		
	VP		VP			
.UNIT= ( { DA { ,2311 } { ,2314 } } { TA ,tape type } { sda-X'cuu' } )	X	X	X	X	X	X
	*		*		*	*
.SPACE= ( { TRK } { CYL } { record } { length } { ,primary } { [,secondary] } { [,HOLD] } )			X	X	X	X
			X	X	X	X
	X	X	X	X	X	X
	X	X	X	X	X	X
.VOLUME= ( { PUBLIC } { PRIVATE } { valseqno } { PRIVATE } { valseqno } { [,...] } )	X	X				
			X	X	X	X
	X	X	X	X	**	**
.LABEL= ( { fileseqno-integer } { labeltype- { NL } { SL } { SUL } { AL } { AUL } } } { [,RETPD=integer] } )					X	X
					X	X
					X	X
					X	X
[.DISP=NEW]	X	X	X	X	X	X
[.OPTION=JOBLIB]	X	X	X	X	X	X
[.RET=codes]	X	X	X	X		

1. Keyword not used for macro instructions. For PCS dumps, use PCS OUT ddname.
2. Keyword not used with DDEF macro instruction.

DDEF (for old, existing data sets)

Operands	When Applicable					
	VAM			BSAM or QSAM		
	Cataloged		Cat.		Uncat.	
	VAM	JOBLIB	DA	TA	DA	TA
DDNAME=data definition name [.DSORG=data set organization]	X	X	X	X	X	X
.DSNAME= {symbol } {*symbol }	X	X	X	X	PS	PS
[.DCB=(.BUFOFF=absexp,DEN= integer,...)]	X	X	X	X	X	X
.UNIT= ( ( DA {,2311 } {,2314 } TA [,tape type] [sda-X'cuu'] ) )	*	*	*	*	*	*
.SPACE= ( ( TRK CYL record length primary [.secondary] [.HOLD] ) )					X	X
.VOLUME= ( ( [PUBLIC ] [PRIVATE ] [volseqno ] ) ) ( ( [PRIVATE ] [volserno ] ) ) )	X	X	X	X		
.LABEL= ( ( [fileseqno-integer] [.labeltype- {NL SL SUL AL AUL } ] ) ) [.RETPD=integer] )	X	X	X	X	X	X
.DISP= { OLD MOD }	X	X	X	X	X	X
.OPTION= { JOBLIB CONC }		X	X	X	X	X
[.RET=codes]	X	X				

\*See DCB macro instruction in ASSEMBLER section for other DCB subparameters.  
\*\*Must be specified here

## Linkage Editor Statements

TRAITS	Redefines control-section attributes, which must be identical when sections are combined.
COMBINE	Combines two or more control sections of one module into one control section.
INCLUDE (form 1)	Obtains program modules from libraries; places them in one output module.
INCLUDE (form 2)	Scans user library or JOBLIBs to locate modules that resolve all unresolved external symbol definitions in output module.
INCLUDE (form 3)	Scans user library or JOBLIBs to locate modules that resolve external symbols that user did not specify as being unresolved.
RENAME	Deletes or renames control section names and entry point names, and renames external references to be included in output module.
END	Indicates end of linkage editor control statements for current run; linkage editor statements are then executed; remaining unresolved references and those resolvable from SYSLIB will be listed in separate diagnostic messages.

## Statement Formats

Operation	Operands
control statement name	one or more operands, delimited by commas; field may be blank.
TRAITS	csname $\left[ \begin{array}{l} ([VARIABLE] [.READONLY] \\ [.PUBLIC] [.PROTO] \\ [.COMMON] [.PRVLGD] \\ [.SYSTEM]) \end{array} \right]$
COMBINE	csname,...
RENAME	$\left\{ \begin{array}{l} \text{extref}_1, (\text{extref}_2) \\ \text{epname}_1 [(\text{epname}_2)] \\ \text{csname}_1 [(\text{csname}_2)] \end{array} \right\} [, \dots]$
INCLUDE (Form 1)	[ddname].(module [,...])
INCLUDE (Form 2)	ddname
INCLUDE (Form 3)	ddname-(extref[,...])
END	always blank

Statements must not exceed 256 characters

### Operands

csname = control section name in first module mentioned in next form-1 INCLUDE statement  
 module = name of object module to be included  
 ddname = symbolic name of library to be searched; specified as SYSLIB, USERLIB or symbolic name of DDEF command (or macro instruction) for JOBLIBs  
 extref = external symbol reference  
 epname = entry point name



## CONTENTS

### PL/I

Preprocessor Statements . . .	37
Functions . . .	37
Formats . . .	38
Statements . . .	39
Functional Groups . . .	39
Formats . . .	41
Format Items . . .	42
Picture Characters . . .	42
ON-Conditions Codes . . .	43
Built-in Functions and Pseudo Variables . . .	44
Functional Groups . . .	44
Formats . . .	47
Keywords . . .	47

### FORTRAN

Statement Meanings . . .	57
Expressions . . .	58
Statement Formats . . .	59
Source Program Characters . . .	63
Mathematical Function Subprograms . . .	64
Service Subprograms . . .	66

### ASSEMBLER

Statements . . .	67
Instructions	
Basic Instruction Formats . . .	70
Standard Instruction Set . . .	71
Extended Mnemonic Instruction Codes . . .	74
User Macro Instructions	
Functional Groups . . .	75
Operand Expression . . .	76
Macro Instruction Formats . . .	77
(Access method macro instructions are under access methods; e.g., BSAM, QSAM.)	



## PREPROCESSOR STATEMENT FUNCTIONS

<b>%ACTIVATE</b>	activates identifier previously defined by <b>%DECLARE</b> that had <b>%DEACTIVATE</b> issued against it; makes identifier eligible for replacement by current preprocessor-assigned values.
<b>assignment statement</b>	used to evaluate preprocessor expressions and assign results to preprocessor variable (i.e., <b>%variable = expression</b> )
<b>%DEACTIVATE</b>	makes identifier ineligible for replacement by preprocessor-assigned values when it appears in subsequent non-preprocessor (source) statements.
<b>%DECLARE</b>	establishes identifier as preprocessor variable or procedure name and activates the identifier for replacement.
<b>%DO</b>	used with <b>%END</b> to delimit a preprocessor DO-group; compiles each iteration of groups procedure or compiles conditionally using <b>%IF</b> , <b>%THEN</b> , and <b>%ELSE</b> .
<b>%end</b>	delimits end of <b>%DO</b> -group or <b>%PROCEDURE</b> procedures
<b>%GO TO</b>	causes compiler scan to be transferred unconditionally to a non-sequential preprocessor statement for subsequent compilation.
<b>%INCLUDE</b>	retrieves source program text from a library (POD) and incorporates it into the PL/I program being compiled.
<b>%NULL</b>	provides transfer points for <b>%GO TO</b> statements; no operation is performed.
<b>%PROCEDURE</b>	used in conjunction with <b>%END</b> to delimit preprocessor procedure; usually to create a functional procedure for compilations.
<b>RETURN</b>	returns a value and control back to point from which preprocessor procedure was invoked; no <b>%</b> sign is necessary.



### PREPROCESSOR STATEMENT FORMATS

```

%[label:] ...    ACTIVATE identifier [,identifier] ...;

%assignment statement
%[label:] ...    preprocessor-variable=preprocessor-expression;

%[label:] ...    DEACTIVATE identifier [,identifier] ...;
%[label:] ...    DECLARE identifier
                  { FIXED | CHARACTER entry-declaration }
                  { identifier { FIXED | CHARACTER
                    entry-declaration } } ...;

                                where entry-declaration is:
                                ENTRY [ ( ( CHARACTER | FIXED
                                    [, ( CHARACTER | FIXED ) ] ... )
                                RETURNS ( CHARACTER | FIXED )
                                [ i=m1 [ { TO m2 [ BY m3 ] } ] ] ]
                                where i=preprocessor variable
                                    m1,m2,m3=preprocessor expressions

%[label:] ...    DO
%[label:] ...    END
%[label:] ...    { GO TO }
%[label:] ...    { GOTO }
%[label:] ...    IF
%[label:] ...    INCLUDE
%[label:] ...    THEN preprocessor-clause1
%[label:] ...    ELSE preprocessor-clause2
%[label:] ...    { ddname1 (member-name1) }
%[label:] ...    { ddname 2 (member-name 2) } } ...;

%null statement
%[label:] ...;
%label: [label:] ... PROCEDURE [(identifier [...]) ...]
[return:] ... RETURN (CHARACTER | FIXED);
[return:] ... RETURN (preprocessor-expression);

```

## PLI STATEMENT FUNCTIONAL GROUPS

### Descriptive statements

DECLARE	explicitly declares attributes of names
FORMAT	specifies format list to control format of data being transmitted
OPEN	allows certain attributes to be specified for a file name in addition to opening that file for processing

### Input/Output statements

#### RECORD I/O transfer statements

READ	transmits record from files opened for INPUT or UPDATE to a variable or buffer
WRITE	makes RECORD transmission of record from a variable in internal storage to OUTPUT or UPDATE file
REWRITE	for UPDATE files, it replaces existing record in data set
LOCATE	for buffered output files; allocates based variable in a buffer; may also transmit previously allocated based variable
DELETE	deletes record from UPDATE file

#### STREAM I/O transfer statements

GET	makes stream transmission of data from external source (i.e., data set, terminal, or SYSIN log) or internal source (i.e., character string variable) to one or more variables
PUT	makes stream transmission of data from one or more internal storage locations to a data set on external medium (i.e., data set, terminal, or SYSPRINT) or to an internal receiving field (i.e., character string variable)

#### I/O control statements

OPEN	opens file by associating name of file with data set; can also complete attribute specifications for file
CLOSE	dissociates file from data set previously associated with by opening current task

#### Terminal I/O statements

DISPLAY	displays message at user's terminal; optionally requests a response
---------	---

#### Data Movement and Computational statements

assignment	assigns a value to one or more variables
GET	with STRING option; values of string are assigned to specified variables
PUT	with STRING option; values of specified variables are concatenated into string and assigned as the value of the specified string

#### Control Statements

GO TO	transfers control to specified location
IF	tests value of an expression, and controls flow of execution based on results
DO	specifies repetitive execution of statements within DO-group
CALL	transfers control to specified procedure

RETURN	terminates execution of procedure containing RETURN; returns to invoking procedure or task
END	terminates blocks and groups
STOP	immediately terminates major task
EXIT	terminates execution of task containing EXIT
WAIT	issued during activation of block to delay further execution until certain specified events are completed
null	causes no action; label of null statement used in conjunction with the CHECK condition

**Exception Control Statements**

ON	specifies action upon exception-condition interruptions as user-defined or standard-system procedure
REVERT	used in block to cancel effect of one or more ON statements previously executed in that block; reestablishes action specifications in force when block was activated
SIGNAL	simulates occurrence of interruption for specified condition; can be used to test current action specification for that condition

**Program Structure Statements**

PROCEDURE	heads a procedure; defines primary entry point, parameters (if any), attributes of any value returned by procedure, and any special characteristics of the procedure
ENTRY	specifies secondary entry point of procedure
BEGIN	heads and identifies a begin block
END	terminates blocks of groups
DO	specifies the statements contained within a DO-group are to be considered as an entity for purposes of flow or control
ALLOCATE	allocates storage for specified controlled or based data, independent of block boundaries
FREE	frees storage allocated for specified based or controlled variables (see ALLOCATE)

## PL/I STATEMENT FORMATS

**ALLOCATE**     option 1  
                 [level] identifier [dimension] [attribute] ...  
                 [,level] identifier [dimension] [attribute] ... ...;  
                 option 2  
                 based-variable-identifier [SET(pointer-variable)]  
                 [IN (area-variable)]  
                 [,based-variable-identifier [SET(pointer-variable)]  
                 [IN (area-variable)]] ...;

**assignment statement**  
                 option 1 (element-assignment)  
                 {element-variable} [...] = element-expression;  
                 {pseudo-variable} [...] = element-expression;  
                 option 2 (array assignment)  
                 {array-variable} [...] = { structure-expression [,BY NAME] }  
                 {pseudo-variable} [...] = { array-expression [,BY NAME] }  
                 { element-expression } ;  
                 option 3 (structure assignment)  
                 structure-variable [...] = { structure-expression [,BY NAME] }  
                 { element-expression } ;

**BEGIN**            [ORDER|REORDER] ;

**CALL**            entry-name [(argument [...] ...)]

**CLOSE**           FILE(file-name) [-FILE(file-name)] ...;

**DECLARE**         { [structure-level] identifier [( [lower-bound:upper bound [...] ...])  
                 [attributes] { 1, ... } [...] ... ] } ;  
                 note: attributes may be data, entry, array, file, scope, or storage type

**DELAY**           (element-expression).

**DELETE**         FILE (file-name)  
                 [KEY (expression)]  
                 [EVENT (event-variable)] ;

**DISPLAY**        option 1  
                 (element-expression);  
                 option 2  
                 (element-expression)  
                 REPLY (character-variable)  
                 [EVENT (event-variable)] ;

**DO**                option 1  
                 ;  
                 option 2  
                 WHILE (element-expression) ;  
                 option 3  
                 { pseudo-variable } = specification [...] ...;  
                 { variable } = specification [...] ...;  
                 where specification is:  
                 expression1 [ { TO expression2 [BY expression3] } ]  
                 [ { BY expression3 [TO expression2] } ]  
                 [WHILE (expression4)]

**END**              [label] ;

**entry statement**  
                 entry-name: [entry-name] ...  
                 ENTRY [(parameter [...] ...)]  
                 [RETURNS (attribute...)] ;

**EXIT** ;

## FORMATS

label:[label:]...FORMAT (format-list):

where format-list items are:

<i>data format items</i>	
character string	A[(field-width)]
binary bit string	B[(field-width)]
complex data item	C(real-format-item [, real-format-item])
decimal floating point	E(field-width, number-of-fractional-digits) [, number-of-significant-digits]
decimal fixed point	F(field-width [, number-of-fractional-digits] [, scaling-factor] )
spacing character	X(field-width)
numeric data or character string picture	P'picture specification'

where picture characters are:

### CHARACTER-STRING SPECIFIERS

X position may contain any character  
A any alphabetic or blank  
9 any digit or blank

### ARITHMETIC DIGIT and POINT SPECIFIERS

9 any decimal digit  
V assumed point and subfield delimiter

### ZERO SUPPRESSION CHARACTERS

Z digit or blank (leading zeros)  
\* digit or \* (leading zeros)  
Y digit or blank (all zeros)

### STATIC or DRIFTING CHARACTERS

(These are also zero suppression characters)

\$ digit, \$, or blank  
S digit, ±, or blank  
+ digit, +, or blank  
- digit, -, or blank

### INSERTION CHARACTERS

if zero suppression and no preceding digit,  
a blank, an asterisk, or a drifting character  
appears

same as comma  
/ same as comma  
B blank

### CREDIT, DEBIT, and OVERPUNCHED SIGNS

CR CR if field < 0  
DB DB if field < 0  
T digit will be overpunched by sign  
I digit will be overpunched by + if field >= 0  
R digit will be overpunched by - if field < 0

### EXPONENT SPECIFIERS

K assumed start of exponent  
E start of exponent (letter E appears)

### SCALING FACTOR

F (optionally signed decimal-integer-constant)  
moves point the specified number of places  
right if +, or left if -

*remote format items*

R(statement-label-designator)

*control format items*

COLUMN (character-position)

LINE(line-number)

PAGE

SKIP[(relative-position-of-next-line)]

FREE

option 1  
controlled-variable[...]; ...;

option 2  
[pointer-qualifier ->]  
based-variable [IN(area-variable)]

[, [pointer-qualifier >]  
based-variable [IN(area-variable)]] ...;

GET

option-list;

where option-list is:

[FILE(filename) { STRING(character-string-name)  
EDIT (data-list)(format-list)  
[data-specification { [(data-list)(format-list)]... } ]  
LIST(data-list)  
DATA(data-list) } ]

[COPY] [SKIP [(expression)]]

GO TO }  
GOTO }

{ label-constant  
element-label-variable } ;

IF

element-expression

THEN unit1

[ELSE unit2];

*note:* unit1 and unit2 may be statements, nested IFs, etc.

```

LOCATE      variable
            FILE(file-name) [SET(pointer-variable)]
            [KEYFROM(expression)];
null statement
ON          [label:] ...;
            condition [SNAP] {SYSTEM; | on-unit}
            where conditions are:
                AREA                original status  changing prefix
                CHECK(name-list)   on           NOCHECK
                CONDITION(identifier) off
                CONVERSION(or CONV) on           NOCONVERSION
                ENDFILE             on
                ENDPAGE             on
                ERROR               on
                FINISH              on
                FIXEDOVERFLOW(or FOFL) on         NOFIXEDOVERFLOW
                KEY                 on
                NAME                on
                OVERFLOW(or FOFL)  on           NOOVERFLOW
                RECORD(file-name)  on
                SIZE                off
                SUBSCRIPTRANGE     off
                STRINGRANGE        off           NOSTRINGRANGE
                TRANSMIT(file-name) on
                UNDEFINEDFILE(file-name) on
                (or)
                UNDERFLOW(or UFL)  on           NOUNDERFLOW
                ZERODIVIDE(or ZDIV) on           NOZERODIVIDE

OPEN       FILE(file-name)[options-group]
            [,FILE(file-name)[options-group] ] ...;
            where option-group is:
            [DIRECT] SEQUENTIAL]
            [BUFFERED] | UNBUFFERED]
            [STREAM] [RECORD]
            [INPUT] [OUTPUT] [UPDATE]
            [KEYED] [EXCLUSIVE]
            [BACKWARDS]
            [TITLE(element-expression)]
            [PRINT]
            [LINESIZE(element-expression)]
            [PAGESIZE(element-expression)]

procedure statement
entry-name:
[entry-name:] ...

PROCEDURE[(parameter[... ] ...)]
[OPTIONS(option-list)]
[RECURSIVE] [RETURNS(attribute...)]
[ORDER 1 REORDER]

PUT        FILE(file-name)
            STRING(character-string-variable)
            [data-specification { EDIT(data-list)(format-list)
                                [(data-list)(format-list)] ... }
                                LIST(data-list)
                                DATA[(data-list)]
            ]
            [COPY]
            [PAGE[LINE(element-expression)]
            SKIP [(element-expression)]
            LINE(element-expression)];

READ       option-list;
            where option-list is:
            FILE(file-name)
            [ [INTO(variable) [ {KEY(expression)
                                [KEYTO(character-string-variable)} ] [EVENT
                                (event-variable)]
            ]
            [SET(pointer-variable) [KEY(expression)
                                [KEYTO(character-string-variable) ]
            ]
            [IGNORE(expression) [EVENT(event-variable) ]
            ]
            ]
            ]
RETURN     option 1
            ;
            option 2
            (expression);

REVERT     condition;
            note: see ON statement for condition options

REWRITE    FILE(file-name)
            [FROM(variable)]
            [KEY(element-expression)]
            [EVENT(event-variable)];

SIGNAL     condition;
            note: see ON statement for condition options

STOP;

WAIT       (event-name [,event-name] ...)
            [(element-expression)];

WRITE      FILE(file-name) FROM(variable)
            [KEYFROM(element-expression)]
            [EVENT(event-variable)];

```

## BUILT-IN FUNCTIONS AND PSEUDO VARIABLES

### Computational

BIT	convert specified value to bit string
BOOL	perform Boolean operation on two bit strings
CHAR	convert given value to character string
HIGH	forms character string of specified length from highest character in collating sequence
INDEX	search specified string for specified bit or character string configuration
LENGTH	find string length of given value
LOW	form character string of specified length from lowest character of collating sequence
REPEAT	concatenate specified string with itself a specified number of times to form new string
STRING	concatenates all elements in an aggregate variable into single string element
SUBSTR	extract a substring of specified length from a given string (also a pseudo variable)
UNSPEC	return bit string that is the internal coded representation of specified value (also pseudo variable)

### Arithmetic

ABS	return absolute value of specified value
ADD	return sum of two numbers with specified precision
BINARY	convert value to binary equivalent
CEIL	get smallest integer greater than or equal to specified value
COMPLEX	form complex number from two given real values (also pseudo variable)
CONJG	find conjugate of complex number (i.e., sign of imaginary portion reversed)
DECIMAL	convert given value to decimal base of specified precision
DIVIDE	divide two specified numbers to specified precision
FIXED	convert value to fixed-point scale with specified precision
FLOAT	convert number to floating-point scale with specified precision
FLOOR	determine largest integer that does not exceed specified value
IMAG	search specified string for specified bit or character string configuration (also pseudo variable)
MAX	extract value of highest valued expression from set of two or more expressions
MIN	extract value of lowest valued expression from set of two or more expressions
MOD	perform division (ignoring signs) and extract remainder resulting from division of one real number by another
MULTIPLY	multiply two numbers to specified precision
PRECISION	convert value to specified precision
REAL	extract real part of complex number (also pseudo variable)
ROUND	round given expression at specified digit
SIGN	determine if value is positive, negative, or zero
TRUNC	truncate value to an integer

### Mathematical

ATAN	find arctangent of X or X/Y and express result in radians
ATAND	find arctangent of X or X/Y and return result in degrees

ATANH	find inverse hyperbolic tangent of given value
COS	find cosine of an angle expressed in radians
COSD	find cosine of specified real value in degrees
COSH	find hyperbolic cosine of specified value
ERF	find error function of given real value
ERFC	find complement of error function for specified real value
EXP	raise 'e' (the base of natural logarithm system) to given value
LOG	return natural logarithm (i.e., base e) of specified value
LOG10	find base 10 common logarithm of specified value
LOG2	find binary logarithm of specified value
SIN	return sine of specified value expressed in radians
SIND	find sine of given real value in degrees
SINH	find hyperbolic sine of specified value
TAN	return tangent of specified value in radians
TAND	return tangent of given real value, expressed in degrees
TANH	return hyperbolic tangent of specified value

#### Array Manipulation

ALL	tests all bits in bit string array to determine if corresponding bits of given array elements are all 1's
ANY	tests the bits in bit-string array to determine if at least one of the corresponding bits of the given array elements is set to 1
DIM	find current extent for specified dimension of given array
HBOUND	find current upperbound for specified dimension of given array
LBOUND	find current lower bound for specified dimension of given array
POLY	form polynomial from two given arguments and return value of that polynomial
PROD	return product of all the elements of a given array
SUM	find sum of all elements in given array

#### Condition Indications

DATAFIELD	extract contents of datafield that caused NAME condition to be raised
ONCHAR	extract character that raised a CONVERSION condition (also pseudo variable)
ONCODE	determine type of interruption that caused on-unit to become active
ONCOUNT	used in any on-unit entered due to abnormal completion of input/output event to determine number of interruptions (including current one) remaining to be handled
ONFILE	determine name of file for which input/output or CONVERSION condition was raised
ONKEY	extract value of key for record that caused an input/output or CONVERSION condition to be raised
ONLOC	determine entry point to procedure in which ON condition is raised
ONSOURCE	extract contents of field being processed when CONVERSION condition was raised (also pseudo variable)

#### Based Storage Functions

ADDR	determine location of specified variable
EMPTY	clear an area of storage (defined by an area variable) by effectively freeing all allocations within the area
NULL	return a null pointer value that cannot identify any allocation



---

NULLO return a null offset value to indicate that an offset variable does not currently identify an allocation.

Miscellaneous Functions

ALLOCATION determine if storage is allocated for specified controlled variable

COUNT determine number of data items transmitted during last GET or PUT operation on specified file

DATE determine years, month, and day of current date

LINENO find current line number for file having PRINT attribute

TIME determine current time in hours, minutes, seconds, and milliseconds

## PL/I KEYWORDS AND THEIR USES (includes Built-in Function Formats)

### Keywords

ABS(x)  
 %ACTIVATE or %ACT  
 ADD(x,y,p[,q])  
 ADDR(x)  
 ALIGNED  
 ALL(x)  
 ALLOCATE  
 ALLOCATION(x)  
 ANY(x)  
 AREA  
 AREA[size]  
 ATAN(x[,y])  
 ATAND(x[,y])  
 ATANH(x)  
 AUTOMATIC or AUTO  
 BACKWARDS  
  
 BASED(pointer-variable)  
 BEGIN  
 BINARY or BIN  
 BINARY or BIN (x[,p[,q]])  
 BIT(length) [VARYING]  
 BIT(expression [,size])  
 BOOL(x,y,w)  
 BUFFERED or BUF  
 BUFFERS(n)  
  
 BUILTIN  
 BY  
 BY NAME  
  
 CALL entry-name  
  
 CEIL(x)  
 CHAR(expression [,size])  
 CHARACTER or CHAR  
 (length)[VARYING]  
 CHECK(name-list)  
 CLOSE  
 COBOL  
  
 COLUMN or COL(w)  
 COMPLETION(event-name)  
 COMPLEX(or CPLX)  
 COMPLEX or CPLX(a,b)  
 CONDITION(name)  
 CONJG(x)  
 CONSECUTIVE  
  
 CONTROLLED or CTL  
 (pointer-variable)  
 CONVERSION or CONV  
 COPY  
 COS(x)  
 COSD(x)  
 COSH(x)  
 COUNT(file-name)

### Use of Keywords

built-in function  
 preprocessor statement  
 built-in function  
 built-in function  
 attribute(data)  
 built-in function  
 statement  
 built-in function  
 built-in function  
 condition  
 attribute (data, program control)  
 built-in function  
 built-in function  
 built-in function  
 attribute (storage)  
 attribute(file), option of OPEN  
 statement  
 attribute(storage)  
 statement  
 attribute (data, arithmetic)  
 built-in function  
 attribute(data, string)  
 built-in function  
 built-in function  
 attribute(file)  
 option of ENVIRONMENT attri-  
 bute  
 attribute(entry)  
 clause of DO statement  
 option of the assignment state-  
 ment  
 statement or option of INITIAL  
 attribute  
 built-in function  
 built-in function  
  
 attribute(data, string)  
 condition  
 statement  
 option of ENVIRONMENT attri-  
 bute  
 format item  
 built-in function, pseudo-variable  
 attribute (data, arithmetic)  
 built-in function, pseudo-variable  
 condition  
 built-in function  
 option of ENVIRONMENT attri-  
 bute  
  
 attribute(storage)  
 condition  
 option of GET statement  
 built-in function  
 built-in function  
 built-in function  
 built-in function

CTLASA	option of ENVIRONMENT Attribute
CTL360	option of ENVIRONMENT attribute
DATA	STREAM I/O transmission mode
DATAFIELD	built-in function
DATE	built-in function
%DEACTIVATE or %DEACT	preprocessor statement
DECIMAL or DEC	attribute (data,arithmetic)
DECIMAL or DEC(x[,p[,q] ])	built-in function
DECLARE or DCL	statement
%DECLARE or %DCL	preprocessor statement
DEFINED or DEF base-identifier $\left\{ \begin{array}{l} \text{subscript-list} \\ \text{POSITION} \end{array} \right\}$ $\left\{ \begin{array}{l} * \end{array} \right\}$	attribute(data)
DELAY(n)	statement
DELETE	statement
DIM(x,n)	built-in function
DIRECT	attribute(file)
DISPLAY	statement
DIVIDE(x,y,p[,q])	built-in function
DO	statement
%DO	preprocessor statement
EDIT	STREAM I/O transmission mode
ELSE	clause of IF statement
%ELSE	clause of %IF statement
EMPTY	built-in function
END	statement
%END	preprocessor statement
ENDFILE(file-name)	condition
ENDPAGE(file-name)	condition
ENTRY[(parameter-attribute- list)] [...]	attribute(entry) or statment
ENVIRONMENT or ENV	attribute(file)
ERF(x)	built-in function
ERFC(x)	built-in function
ERROR	condition
EVENT <sup>1</sup>	option of CALL, READ, WRITE, REWRITE, and DELETE statements, attribute (data, program control)
EXCLUSIVE	attribute(file)
EXIT	statement
EXP(x)	built-in function
EXTERNAL or EXT	attribute (data,scope)
F(block-size[,record-size])	option of ENVIRONMENT attribute
FILE	attribute(file)
FILE(file-name)	option of GET and PUT statements, specification of RECORD I/O statements
FINISH	condition
FIXED	attribute (data, arithmetic)
FIXED(x[,p[,q] ])	built-in function
FIXEDOVERFLOW or FOFL	condition
FLOAT	attribute(arithmetic)
FLOAT(x[,p[,q] ])	built-in function
FLOOR(x)	built-in function
FORMAT(format-list)	statement
FREE	statement
FROM(variable)	option of WRITE or REWRITE statements

G <sup>3</sup> (max-message-size)	option of ENVIRONMENT attribute
GENERIC entry-name-declaration[,...]	attribute(entry)
GENKEY <sup>2</sup>	option of ENVIRONMENT attribute
GET	statement
GO TO or GOTO	statement
%GO TO or %GOTO	preprocessor statement
HBOUND(x,h)	built-in function
HIGH(i)	built-in function
IF	statement
%IF	preprocessor statement
IGNORE(n)	option of READ statement
IMAG(x)	built-in function, pseudo-variable
IN	option of ALLOCATE and FREE statements
%INCLUDE	preprocessor statement
INDEX(string,configuration)	built-in function
INDEXAREA <sup>2</sup>	option of ENVIRONMENT attribute
INDEXED	option of ENVIRONMENT attribute
INITIAL or INIT(item[,item] ...)	attribute(data)
INITIAL CALL	
entry name[argument-list]	attribute(data)
INPUT	attribute(file), option of the OPEN statement
INTERNAL or INT	attribute(scope,data)
INTO(variable)	option of READ statement
KEY(file-name)	condition
KEY(x)	option of READ, DELETE, and REWRITE statements
KEYED	attribute(file), option of OPEN statement
KEYFROM(x)	option of WRITE statement
KEYTO(variable)	option of READ statement
LABEL[statement-label-constant[,...]]	attribute(data, program control)
LENGTH(string)	built-in function
LBOUND(x,n)	built-in function
LEAVE	option of ENVIRONMENT attribute
LIKE structure-variable	attribute
LINE(w)	format item, option of PUT statement
LINENO(file-name)	built-in function
LINESIZE(w)	option of OPEN statement
LIST	STREAM I/O transmission mode
LOCATE	statement
LOG(x)	built-in function
LOG2(x)	built-in function
LOG10(x)	built-in function
LOW(i)	built-in function
MAIN	option of PROCEDURE statement
MAX(x <sub>1</sub> ,x <sub>2</sub> ,...,x <sub>n</sub> )	built-in function
MIN(x <sub>1</sub> ,x <sub>2</sub> ,...,x <sub>n</sub> )	built-in function
MOD(x <sub>1</sub> ,x <sub>2</sub> )	built-in function
MULTIPLY(x <sub>1</sub> ,x <sub>2</sub> ,p[,q])	built-in function

NAME(file-name)	condition
NCP(n)	option of ENVIRONMENT attribute
NOCHECK	condition prefix identifier (disables CHECK)
NOCONVERSION or NOCONV	condition prefix identifier (disables CONVERSION)
NOFIXEDOVERFLOW or NOFOFL	condition prefix identifier (disables FIXEDOVERFLOW)
NOLOCK <sup>2</sup>	option of READ statement
NOOVERFLOW or NOOFL	condition prefix identifier (disables OVERFLOW)
NOSIZE	condition prefix identifier (disables SIZE)
NOSTRINGRANGE or NOSTRG	condition prefix identifier (disables STRINGRANGE)
NOSUBSCRIPTRANGE or NOSUBRG	condition prefix identifier (disables SUBSCRIPTRANGE)
NOUNDERFLOW or NOUFL	condition prefix identifier (disables UNDERFLOW)
NOWRITE <sup>2</sup>	option of ENVIRONMENT attribute
NOZERODIVIDE or NOZDIV	condition prefix identifier (disables ZERODIVIDE)
NULL	built-in function
NULLO	built-in function
OFFSET(area-name)	attribute(data, program control)
ON	statement
ONCHAR	built-in function, pseudo-variable
ONCOUNT	built-in function
ONCODE	built-in function
ONFILE	built-in function
ONKEY	built-in function
ONLOC	built-in function
ONSOURCE	built-in function, pseudo-variable
OPEN	statement
OPTIONS(list)	option of PROCEDURE statement
ORDER	option of PROCEDURE and BEGIN statements
OUTPUT	attribute(file), option of the OPEN statement
OVERFLOW or OFL	condition
PAGE	format item, option of PUT statement
PAGESIZE (w)	option of OPEN statement
PICTURE or PIC	
{ 'character-picture-specification' }	attribute(data, string)
{ 'numeric-picture-specification' }	
POINTER or PTR (area-variable)	attribute(data, program control)
PENDING <sup>3</sup>	condition
POLY(a,x)	built-in function
POSITION or POS(i) <i>see DEFINED</i>	attribute(data)
PRECISION or PREC(x,p[.q])	built-in function
PRINT	attribute(file), option of OPEN statement
PRIORITY <sup>1</sup> (x)	option of CALL statement
PRIORITY <sup>1</sup> [(task-name)]	built-in function, pseudo-variable

PROCEDURE or PROC	statement
%PROCEDURE or %PROC	preprocessor statement
PROD(x)	built-in function
PUT	statement
R <sup>3</sup> (maximum-record-size)	option of ENVIRONMENT attribute
READ	statement
REAL	attribute(data, arithmetic)
REAL(x)	built-in function, pseudo-variable
RECORD	attribute(file), option of OPEN statement
RECURSIVE	option of PROCEDURE statement
REENTRANT	option of PROCEDURE statement
REFER	option of BASED attribute
REGIONAL <sup>3</sup>	option of ENVIRONMENT attribute
REORDER	option of PROCEDURE and BEGIN statements
REPEAT(string,i)	built-in function
REPLY(c)	option of DISPLAY statement
RETURN	statement
RETURNS(attribute)	attribute(entry), option of PROCEDURE and ENTRY statements
REVERT	statement
REWIND	option of ENVIRONMENT attribute
REWRITE	statement
ROUND(x,n)	built-in function
SEQUENTIAL or SEQL	attribute(file)
SET(pointer-variable)	option of ALLOCATE, LOCATE, and READ statements
SIGN(x)	built-in function
SIGNAL	statement
SIN(x)	built-in function
SIND(x)	built-in function
SINH(x)	built-in function
SIZE	condition
SKIP[(x)]	format item, option of GET and PUT statements
SNAP	option of ON statement
SQRT(x)	built-in function
STATIC(pointer-variable)	attribute(storage)
STATUS(event-name)	built-in function, pseudo-variable
STOP	statement
STREAM	attribute(file), option of OPEN statement
STRING(x)	built-in function, pseudo-variable
STRINGRANGE or STRG	condition
STRING(string-name)	option of GET and PUT statements
iSUB	dummy variable of DEFINED attribute
SUBSCRIPTRANGE or SUBRG	condition
SUBSTR(string,i[,j])	built-in function, pseudo-variable
SUM(x)	built-in function
SYSIN	name of standard system input file
SYSPRINT	name of standard system output file
SYSTEM	option of the ON statement

TAN(x)	built-in function
TAND(x)	built-in function
TANH(x)	built-in function
TASK <sup>1</sup>	attribute (data, program control), option of PROCEDURE statement
TASK <sup>1</sup> [(task-name)]	option of CALL statement
THEN	clause of IF statement
%THEN	clause of %IF statement
TIME	built-in function
TO	clause of DO statement
TITLE(x)	option of OPEN statement
TRANSIENT <sup>3</sup>	attribute(file)
TRANSLATE(s,r[,p])	built-in function
TRANSMIT	condition
TRKOFI	option of ENVIRONMENT attribute
TRUNC(x)	built-in function
U(max-block-size)	option of ENVIRONMENT attribute
UNALIGNED or UNAL	attribute(data)
UNBUFFERED or UNBUF	attribute(file), option of OPEN statement
UNDEFINEDFILE or UNDF(file-name)	condition
UNDERFLOW or UFL	condition
UNLOCK <sup>2</sup>	statement
UNSPEC(x)	built-in function, pseudo-variable
UPDATE	attribute(file), option of OPEN statement
V(max-block-size[,max-record-size])	option of ENVIRONMENT attribute
VARYING or VAR (see BIT)	attribute(data, string)
VBS <sup>4</sup> (max-block-size [,max-record-size])	option of ENVIRONMENT attribute
VERIFY(expression1,expression2)	built-in function
VS <sup>4</sup> (max-block-size [,max-record-size])	option of ENVIRONMENT attribute
WAIT	statement
WHILE	clause of DO statement
WRITE	statement
ZERODIVIDE or ZDIV	condition

1 - should not be used; it causes abnormal termination

2 - is not used; it is ignored

3 - raises the UNDEFINEDFILE condition

4 - treated same as if key word V is specified

TYPE OF OPERATION	OPERATION SYMBOL		OPERATION
	60-Character Set	48-Character Set	
ARITHMETIC	+	+	addition or prefix +
	-	-	subtraction or prefix -
	*	*	multiplication
	/	/	division
COMPARISON	**	**	exponentiation
	>	GT	greater than
	>=	GE	greater than or equal to
	=	=	equal to
	≠	NE	not equal to
	<=	LE	less than or equal to
	<	LT	less than
	≧	NG	not greater than
LOGICAL (bit string)	¬	NOT	not
		OR	or
	&	AND	and
CONCATENATION		CAT	concatenation

OPERATION	PRIORITY*
**	1st
¬	
+ (prefix)	
- (prefix)	
/	2nd
*	
+ (infix)	3rd
- (infix)	
	4th
=	
>=	
<=	
>	
<	
≠	
≧	
&	6th
	7th

\* Parentheses modify priorities; operations enclosed in parentheses are performed first, beginning with the innermost pair of parentheses. The expression  $(-A)^2$  must be written in PL/I notation as:  
 $(-A)**2$   
 so that reversing the sign of A will be performed first. Without parentheses, exponentiation would be performed first and the result would be  $-(A*A)$  or  $-(A^2)$ .



### CHARACTER SETS WITH EBCDIC AND CARD-PUNCH CODES

These card codes are those used by the TSS/360 high-speed card reader. The code variations for use with the IBM Card Reader can be found in *IBM System/360 Time Sharing System, Terminal User's Guide, Form GC28-2017*.

#### 48-CHARACTER SET

Character	Card-Punch	8-Bit Code
blank	no punches	0100 0000
.	12-8-3	0100 1011
(	12-8-5	0100 1101
+	12-8-6	0100 1110
\$	11-8-3	0101 1011
*	11-8-4	0101 1100
)	11-8-5	0101 1101
_	11	0110 0000
/	0-1	0110 0001
'	0-8-3	0110 1011
:	8-5	0111 1101
=	8-6	0111 1110
A	12-1	1100 0001
B	12-2	1100 0010
C	12-3	1100 0011
D	12-4	1100 0100
E	12-5	1100 0101
F	12-6	1100 0110
G	12-7	1100 0111
H	12-8	1100 1000
I	12-9	1100 1001
J	11-1	1101 0001
K	11-2	1101 0010
L	11-3	1101 0011
M	11-4	1101 0100
N	11-5	1101 0101
O	11-6	1101 0110
P	11-7	1101 0111
Q	11-8	1101 1000
R	11-9	1101 1001
S	0-2	1110 0010
T	0-3	1110 0011
U	0-4	1110 0100
V	0-5	1110 0101
W	0-6	1110 0110
X	0-7	1110 0111
Y	0-8	1110 1000
Z	0-9	1110 1001
0	0	1111 0000
1	1	1111 0001
2	2	1111 0010
3	3	1111 0011
4	4	1111 0100
5	5	1111 0101
6	6	1111 0110
7	7	1111 0111
8	8	1111 1000
9	9	1111 1001

#### 60-CHARACTER SET

Character	Card-Punch	8-Bit Code
blank	no punches	0100 0000
<	12-8-3	0100 1011
(	12-8-4	0100 1100
+	12-8-5	0100 1101
&	12-8-6	0100 1110
12	12-8-7	0100 1111
S	12	0101 0000
*	11-8-3	0101 1011
)	11-8-4	0101 1100
:	11-8-5	0101 1101
]	11-8-6	0101 1110
_	11-8-7	0101 1111
11	11	0110 0000
/	0-1	0110 0001
.	0-8-3	0110 1011
%	0-8-4	0110 1100
-	0-8-5	0110 1101
>	0-8-6	0110 1110
?	0-8-7	0110 1111
:	8-2	0111 1010
#	8-3	0111 1011
@	8-4	0111 1100
'	8-5	0111 1101
=	8-6	0111 1110
A	12-1	1100 0001
B	12-2	1100 0010
C	12-3	1100 0011
D	12-4	1100 0100
E	12-5	1100 0101
F	12-6	1100 0110
G	12-7	1100 0111
H	12-8	1100 1000
I	12-9	1100 1001
J	11-1	1101 0001
K	11-2	1101 0010
L	11-3	1101 0011
M	11-4	1101 0100
N	11-5	1101 0101
O	11-6	1101 0110
P	11-7	1101 0111
Q	11-8	1101 1000
R	11-9	1101 1001
S	0-2	1110 0010
T	0-3	1110 0011
U	0-4	1110 0100
V	0-5	1110 0101
W	0-6	1110 0110
X	0-7	1110 0111
Y	0-8	1110 1000
Z	0-9	1110 1001
0	0	1111 0000
1	1	1111 0001
2	2	1111 0010
3	3	1111 0011
4	4	1111 0100
5	5	1111 0101
6	6	1111 0110
7	7	1111 0111
8	8	1111 1000
9	9	1111 1001

<i>Composite Symbols</i>	<i>Card Punch</i>	<i>60-Character Set Equivalent</i>	<i>Composite Symbols</i>	<i>Card-Punch</i>
..	12-8-3, 12-8-3	:	<=	12-8-4, 8-6
LE	11-3, 12-5	<=		12-8-7, 12-8-7
CAT	12-3, 12-1, 0-3		**	11-8-4, 11-8-4
**	11-8-4, 11-8-4	**	] <	11-8-7, 12-8-4
NL	11-5, 11-3	] <	] >	11-8-7, 0-8-6
NG	11-5, 12-7	] >	] =	11-8-7, 8-6
NE	11-5, 12-5	] =	> =	0-8-6, 8-6
//	0-1, 0-1	%	/*	0-1, 11-8-4
..	0-8-3, 12-8-3	:	*/	11-8-4, 0-1
AND	12-1, 11-5, 12-4	&	->	11, 0-8-6
GE	12-7, 12-5	> =		
GT	12-7, 0-3	>		
LT	11-3, 0-3	<		
NOT	11-5, 11-6, 0-3	]		
OR	11-6, 11-9			
/*	0-1, 11-8-4	/*		
*/	11-8-4, 0-1	*/		
PT	11-7, 0-3	->		

*Note:* When using the 48-character set, the following rules should be observed:

1. The two periods that replace the colon must be immediately preceded by a blank if the preceding character is a period.
2. The two slashes that replace the percent symbol must be immediately preceded by a blank if the preceding character is an asterisk, or immediately followed by a blank if the following character is an asterisk.
3. The sequence "comma period" represents a semicolon except when it occurs in a comment or character string, or when it is immediately followed by a digit.



## **FORTRAN Statement Meanings**

### **Assignment Statements**

Arithmetic—assign arithmetic value to variable

Logical—assign logical value to variable

ASSIGN—assign statement number to variable

### **Control Statements**

GO TO (unconditional, computed, and assigned)—transfer control to another statement

IF (arithmetic, logical)—test condition and transfer control

DO—repeatedly execute statements that follow

CONTINUE—dummy statement; usually last in DO loop

PAUSE—temporarily halt execution and write message on SYSOUT

STOP—terminate execution of object program

END—defines end of source program or subprogram to compiler

### **Input Output Statements**

READ—read data

WRITE—write data

FORMAT—define structure of data record

NAMelist—lists variables to be read or written

END FILE—define end of data set

REWIND—position to first data set associated with data reference number

BACKSPACE—backspace one record

### **Specification Statements**

Type Statements

IMPLICIT—specify type and length for variable or array

EXPLICIT specification (integer, real, complex, logical) same as

IMPLICIT

DIMENSION—allocate storage for array

COMMON—allow different programs to share storage

### **Subprograms**

FUNCTION—define function subprogram

SUBROUTINE—define subroutine subprogram

END—define physical end of subprogram to be compiled

RETURN—specify logical end of subprogram

CALL—invoke subroutine subprogram

ENTRY—define entry point in subprogram

EXTERNAL—expand in-line function out-of-line

BLOCK DATA—enter data into labeled common block

### **Other FORTRAN Statements Accepted by FORTRAN IV**

READ—read data from system input data set

PUNCH—write data into data set associated with system output

PRINT—write data into data set associated with system output

DATA—define initial values for variables and arrays

DOUBLE PRECISION—specify variables as double-precision variables

## Arithmetic and Logical Expressions

Consist of arithmetic or logical constants, variables, or subscripted variable separated by arithmetic, relational, or logical operators.

### Arithmetic

<u>Operators</u>	<u>Explanation</u>
**	exponentiation
*	multiplication
/	division
+	addition
-	subtraction

### Relational

<u>Operator</u>	<u>Explanation</u>
.GT.	greater than ( $>$ )
.GE.	greater than or equal to ( $\geq$ )
.LT.	less than ( $<$ )
.LE.	less than or equal to ( $\leq$ )
.EQ.	equal to ( $=$ )
.NE.	not equal to ( $\neq$ )

### Logical

<u>operation</u>	<u>Explanation</u>
.NOT.	.NOT.A—if A is .TRUE. then value .NOT.A is .FALSE.; if A is .FALSE. then value .NOT.A is .TRUE.
.AND.	A.AND.B—if A and B are .TRUE. then value A.AND.B is .TRUE.; if either A or B or both are .FALSE., then value A.AND.B is .FALSE.
.OR.	A.OR.B—if A or B or both are .TRUE. then value A.OR.B is .TRUE.; if A and B are .FALSE. then value A.OR.B is .FALSE.

Two logical operators may appear in sequence only if second logical operator is .NOT.

**Order of computations in expressions** Where parentheses are omitted, or entire expression is enclosed within single pair of parentheses, this is order in which operations are performed:

<u>Operation</u>	<u>Hierarchy</u>
Evaluation of functions	1st (highest)
Exponentiation (**)	2nd
Multiplication and division (* and /)	3rd
Addition and subtraction (+ and -)	4th
.LT., .LE., .EQ., .NE., .GT., .GE.	5th
.NOT.	6th
.AND.	7th
.OR.	8th

**Subscript** Specifies one of index coordinates that identify specific element of array. From 1 to 7 subscripts are used in accordance with dimensionality of array being subscripted; multidimensional subscripts, separated by commas; subscripts, enclosed in parentheses, follow array name. One of seven forms;  $v, c', v+c', v-c', c*v, c*v+c', c*v-c'$ , where  $v$  is unsigned, nonsubscripted, integer variable and  $c$  and  $c'$  are unsigned integer constants. Whatever subscript form is used, evaluated result must be greater than 0.

## FORTRAN Statement Formats

### Arithmetic and logical assignment statement

**a = b**

**a** – subscripted or unsubscripted variable

**b** – arithmetic or logical expression

**a** must be logical variable if, and only if, **b** is logical expression

### ASSIGN and assigned GO TO statements

**ASSIGN i to m**

.

**GO TO m (x<sub>1</sub>,x<sub>2</sub>,x<sub>3</sub>,...,x<sub>n</sub>)**

**i** – executable statement number

**x<sub>1</sub>,x<sub>2</sub>,x<sub>3</sub>,...,x<sub>n</sub>** – executable statement numbers

**m** – unsubscripted integer variable, length 4, equal to  
**x<sub>1</sub>,x<sub>2</sub>,x<sub>3</sub>,...,x<sub>n</sub>**

**BACKSPACE a**

**a** – unsigned integer constant or integer variable, length 4, represents data set reference number

**CALL name (a<sub>1</sub>,a<sub>2</sub>,a<sub>3</sub>,...,a<sub>n</sub>)**

**name** – subroutine's subprogram name or name defined in ENTRY statement in subroutine subprogram

**a<sub>1</sub>,a<sub>2</sub>,a<sub>3</sub>,...,a<sub>n</sub>** – actual arguments supplied to subroutine subprogram; each may be of form &n where n is statement number

**COMMON /r/a (k<sub>1</sub>),b(k<sub>2</sub>),.../r/c(k<sub>3</sub>),d(k<sub>4</sub>),...**

**a,b,...,c,d...** – variable or array names

**k<sub>1</sub>,k<sub>2</sub>,...,k<sub>3</sub>,k<sub>4</sub>...** – optional; each composed of 1 through 7 unsigned integer constants, separated by commas, representing maximum value of each subscript in array

**/r/...** – optional common block names consisting of one through six alphabetic characters; first is alphabetic; names must be embedded in slashes

**CONTINUE** no operands

**DATA y<sub>1</sub>...,y<sub>n</sub>/i<sub>1</sub> \*d<sub>1</sub>...,i<sub>n</sub> \*d<sub>n</sub>/y<sub>n+1</sub>...,y<sub>m</sub>/i<sub>n+1</sub> \*d<sub>n+1</sub>...,i<sub>m</sub> \*d<sub>m</sub>/...**

**y<sub>1</sub>...,y<sub>m</sub>** – variables, subscripted variables (subscripts must be integer constants), or array names

**d<sub>1</sub>...,d<sub>m</sub>** – values of integer, real, complex, logical or literal hexadecimal data constants

**i<sub>1</sub>...,i<sub>m</sub>** – unsigned integer constants indicating number of consecutive variables to be assigned value of **d<sub>1</sub>...,d<sub>m</sub>**

**DO**

$$\underbrace{x}_{\text{end of range}} \underbrace{i}_{\text{DO variable}} = \underbrace{m_1}_{\text{initial value}}, \underbrace{m_2}_{\text{test value}}, \underbrace{m_3}_{\text{increment}}$$

**x** – number of executable statement that follows DO statement

**i** – unsubscripted integer variable

**m<sub>1</sub>,m<sub>2</sub>,m<sub>3</sub>** – either unsigned integer constants greater than 0, or unsigned unsubscripted integer variables greater than 0; sum **m<sub>2</sub>+m<sub>3</sub> + 1** must not exceed size of virtual storage; (**m<sub>3</sub>** optional; if omitted, value assumed to be 1; preceding comma must be omitted)

**DOUBLE PRECISION**  $a, b, c, \dots$

$a, b, c, \dots$  – variable names that may be dimensioned in statement, or function names

**DIMENSION**  $a_1 (k_1), a_2 (k_2), a_3 (k_3), \dots, a_n (k_n)$

$a_1, a_2, a_3, \dots, a_n$

$a_1, a_2, a_3, \dots, a_n$  – array names

$k_1, k_2, k_3, \dots, k_n$  – each composed of 1 through 7 unsigned integer constants, separated by commas, representing maximum value of each subscript in array;  $k_1$  through  $k_n$  may be integer variables, length 4, only when they appear in DIMENSION statement within subprogram

**END** no operands

**END FILE**  $a$

$a$  – unsigned integer constant or integer variable, length 4; represents data set reference number

**ENTRY** name ( $a_1, a_2, a_3, \dots, a_n$ )

name – name of entry point containing from 1 to 6 alphabetic and/or numeric characters; first is alphabetic

$a_1, a_2, a_3, \dots, a_n$  – dummy arguments corresponding to actual argument in CALL statement or in a function reference

**EQUIVALENCE** ( $a, b, c, \dots$ ), ( $d, e, f, \dots$ )

$a, b, c, d, e, f, \dots$  – variables that may be subscripted; subscripts may have two forms single-subscripted variable refers to position of variable in array (i.e., 1st, 25th); multiple-subscripted variable refers to array position as in arithmetic statements

**Explicit specification statements**

**type**\* $s$   $a^*s_1 (k_1)/x_1 /, b^*s_2 (k_2)/x_2 /, \dots, z^*s_n (k_n)/x_n /$

type – INTEGER, REAL, LOGICAL, or COMPLEX

\* $s, s_1, s_2, \dots, s_n$  – optional; each  $s$  represents permissible length specification for associated type

$a, b, \dots, z$  – variable, array, or function names

$(k_1), (k_2), \dots, (k_n)$  – optional; each  $k$  composed of 1 through 7 unsigned integer constants, separated by commas, representing maximum value of each subscript in array; each  $k$  may be unsigned integer variable only if in subprogram's type statement

$/x_1 /, /x_2 /, \dots, /x_n /$  – optional; represent initial data values

**Format statement**

**x** **FORMAT** ( $c_1, c_2, \dots, c_n / c_1', c_2', \dots, c_n' / \dots$ )

$x$  – statement number (1 through 5 digits)

$c_1, c_2, \dots, c_n$  and  $c_1', c_2', \dots, c_n'$  – format codes; may be delimited by one of separators: comma, slash, parenthesis; codes specify length, decimal point, position of data in data set; slash (/) separates FORTRAN records

$aW, s$  – transfer integer, real, complex, or logical data

$aIw$  – transfer integer data

$aFw, d$  – transfer real data not containing decimal exponent

$aEw, d$  – transfer real data containing E decimal exponent

$aDw, d$  – transfer real data containing D decimal exponent

$aZW$  – transfer hexadecimal numbers

$aLw$  – transfer logical data

$aAw$  – transfer alphameric data

$wH$  – transfer literal data

$wX$  – skip data when reading or insert blanks when writing

- a – optional; unsigned integer constant; denotes number of times same format code is repetitively referenced
- w – unsigned integer constant, less than or equal to 255, specifying number of data characters
- s – unsigned integer constant specifying number of significant digits
- d – unsigned integer constant specifying number of decimal places to right of decimal point

**EXTERNAL a,b,c,...**

a,b,c,... – names of subprograms used as arguments in other subprograms

**Function statement**

**type FUNCTION name\*s (a<sub>1</sub>,a<sub>2</sub>,a<sub>3</sub>,...,a<sub>n</sub>)**

type – integer, real, complex, or logical

name – name of FUNCTION subprogram

\*s – optional; represents one of permissible length specifications for associated type

a<sub>1</sub>,a<sub>2</sub>,a<sub>3</sub>,...,a<sub>n</sub> – unsubscripted variable, array, or dummy names of of SUBROUTINE or FUNCTION subprograms; at least one argument must be in argument list (arguments may be enclosed in slashes within commas, equivalent to normal format without slashes)

**Function (arithmetic statement)**

**name (a,b,...,n) = expression**

name – any subprogram name

a,b,...,n – distinct (within same statement) unsubscripted variables

expression – any arithmetic or logical expression not containing subscripted variables; statement functions in this expression must be defined previously

**GO TO (computed statement)**

**GO TO (x<sub>1</sub>,x<sub>2</sub>,x<sub>3</sub>,...,x<sub>n</sub>), i**

x<sub>1</sub>,x<sub>2</sub>,...,x<sub>n</sub>, – executable statement numbers

i – unsubscripted integer variable in range:  $1 \leq i \leq n$

**GO TO (unconditional statement)**

**GO TO X**

X – executable statement number

**IF (arithmetic statement)**

**IF (a) x<sub>1</sub>,x<sub>2</sub>,x<sub>3</sub>**

a – arithmetic expression; not complex

x<sub>1</sub>,x<sub>2</sub>,x<sub>3</sub> – statement numbers

**IF (logical statement)**

**If (a)s**

a – logical expression

s – any statement except specification statement, DO statement, or another logical IF statement

**IMPLICIT type\*s(a<sub>1</sub>,a<sub>2</sub>,...,...),type\*s(a<sub>1</sub>,a<sub>2</sub>,...,...)**

type – integer, real, complex, or logical

\*s – optional; one of permissible length specifications for associated type

a<sub>1</sub>,a<sub>2</sub>,... – single alphabetic characters, separated by commas, or range of characters (in alphabetic sequence) denoted by first and last characters of range separated by hyphen (e.g., A-D)

**NAMelist/x/a,b,...,c/y/d,e,...,f/z/g,h,...,j**

x,y,and z,... – NAMelist names, 1-6 alphameric; first, alphabetic

a,b,c,d,... – variable or array names



**PAUSE**

**PAUSE n**

**PAUSE 'message'**

n – unsigned 1–5-digit integer constant

message – any literal constant

**PRINT b, list**

b – number or array name of FORMAT statement describing data

list – series of variable or array names, separated by commas; may be indexed and incremented; specify number of items written and data storage locations

**PUNCH b, list**

b – number or array name of FORMAT statement describing data

list – series of variable or array names, separated by commas; may be indexed and incremented; specify number of items written and data storage locations

**READ (a, b, END=c, ERR=d) list**

a – unsigned integer constant or integer variable, length 4; data set reference number

b – optional; number or array name of FORMAT statement describing data being read, or NAMELIST name; when NAMELIST name specified, LIST not specified

END=c – optional; statement number to which transfer is made at end of data set

ERR=d – optional; number of statement to which transfer is made if error condition in data transfer

list – optional; series of variable or array names, separated by commas; may be indexed and incremented; specify number of items read and data storage locations

READ statement may take different forms; parameters END=c and ERR=d, optional; parameter list, parameter b, or both, may be omitted

**RETURN Statement**

**RETURN i**

i – optional in subroutine subprogram; not applicable in functional subprogram; integer constant or variable, length 4; value, n, denotes nth statement number in argument list of SUBROUTINE statement

**REWIND a**

a – unsigned integer constant or integer variable, length 4, representing data set reference number

**STOP n**

n – optional; unsigned 1–5-digit integer constant

**SUBROUTINE name (a<sub>1</sub>,a<sub>2</sub>,a<sub>3</sub>,...,a<sub>n</sub>)**

name – subprogram name

a<sub>1</sub>,a<sub>2</sub>,a<sub>3</sub>,...,a<sub>n</sub> – arguments, if any; every argument must be non-subscripted variable or array name, dummy name of another SUBROUTINE or FUNCTION subprogram, or of form \*, where \* is return point specified by statement number in calling program

**WRITE (a, b) list**

a – unsigned integer constant or integer variable, length 4; data set reference number

b – optional; number or array name of FORMAT statement describing data being written, or NAMELIST name

list – optional; series of variable or array names, separated by commas; may be indexed and incremented; specify number of items written and data storage locations

**FORTRAN Source Program Characters**

Alphabetic Characters	EBCDIC or BCDIC Punches	Numeric Characters	EBCDIC or BCDIC Punches	
A	12-1	0	0	
B	12-2	1	1	
C	12-3	2	2	
D	12-4	3	3	
E	12-5	4	4	
F	12-6	5	5	
G	12-7	6	6	
H	12-8	7	7	
I	12-9	8	8	
J	11-1	9	9	
K	11-2			
L	11-3			
M	11-4			
N	11-5			
O	11-6			
P	11-7			
Q	11-8			
R	11-9			
S	0-2			
T	0-3			
U	0-4			
V	0-5			
W	0-6			
X	0-7			
Y	0-8			
Z	0-9			
\$	11-3-8			
(see notes)		(see notes)		
		Special Characters	EBCDIC Punches	BCDIC Punches
		+	12-6-8	12
		-	11	11
		/	0-1	0-1
		=	6-8	3-8
		.	12-3-8	12-3-8
		)	11-5-8	12-4-8
		*	11-4-8	11-4-8
		,	0-3-8	0-3-8
		(	12-5-8	0-4-8
		,	5-8	4-8
		blank	no punch	no punch
		(see notes)		

Source programs are coded in either BCDIC or EBCDIC character codes; mixing two is not allowed.  
 \$ is alphabetic character in EBCDIC only  
 ( is special character in EBCDIC only

Characters listed constitute character set acceptable by FORTRAN. In literal data, any valid code is acceptable.

**Mathematical Function Subprograms**

<b>Function</b>	<b>Name</b>	<b>Definition</b>	<b>I or O*</b>	<b>Arguments (no., type)</b>	<b>Returned Function</b>
Exponential	EXP	$e^{arg}$	O	1 Real *4	Real *4
	DEXT	$e^{arg}$	O	1 Real *8	Real *8
	CEXP	$e^{arg}$	O	1 Complex *8	Complex *8
	CDEXP	$e^{arg}$	O	1 Complex *16	Complex *16
Natural Logarithm	ALOG	$\ln(arg)$	O	1 Real *4	Real *4
	DLOG	$\ln(arg)$	O	1 Real *8	Real *8
	CLOG	$\ln(arg)$	O	1 Complex *8	Complex *8
	CDLOG	$\ln(arg)$	O	1 Complex *16	Complex *16
Common Logarithm	ALOG10	$\log_{10}(arg)$	O	1 Real *4	Real *4
	DLOG10	$\log_{10}(arg)$	O	1 Real *8	Real *8
Arcsine	ARSIN	$\arcsin(arg)$	O	1 Real *4	Real *4
	DARSIN	$\arcsin(arg)$	O	1 Real *8	Real *8
Arccosine	ARCOS	$\arccos(arg)$	O	1 Real *4	Real *4
	DARCOS	$\arccos(arg)$	O	1 Real *8	Real *8
Arctangent	ATAN	$\arctan(arg)$	O	1 Real *4	Real *4
	ATAN2	$\arctan(arg_1 / arg_2)$	O	2 Real *4	Real *4
	DATAN	$\arctan(arg)$	O	1 Real *8	Real *8
	DATAN2	$\arctan(arg_1 / arg_2)$	O	2 Real *8	Real *8
Trigonometric Sine (argument in radians)	SIN	$\sin(arg)$	O	1 Real *4	Real *4
	DSIN	$\sin(arg)$	O	1 Real *8	Real *8
	CSIN	$\sin(arg)$	O	1 Complex *8	Complex *8
	CDSIN	$\sin(arg)$	O	1 Complex *16	Complex *16
Trigonometric Cosine (argument in radians)	COS	$\cos(arg)$	O	1 Real *4	Real *4
	DCOS	$\cos(arg)$	O	1 Real *8	Real *8
	CCOS	$\cos(arg)$	O	1 Complex *8	Complex *8
	CDCOS	$\cos(arg)$	O	1 Complex *16	Complex *16
Trigonometric Tangent	TAN	$\tan(arg)$	O	1 Real *4	Real *4
	DTAN	$\tan(arg)$	O	1 Real *8	Real *8
Trigonometric Cotangent	COTAN	$\cotan(arg)$	O	1 Real *4	Real *4
	DCOTAN	$\cotan(arg)$	O	1 Real *8	Real *8
Square Root	SQRT	$\sqrt{arg}$	O	1 Real *4	Real *4
	DSQRT	$\sqrt{arg}$	O	1 Real *8	Real *8
	CSQRT	$\sqrt{arg}$	O	1 Complex *8	Complex *8
	CDSQRT	$\sqrt{arg}$	O	1 Complex *16	Complex *16
Hyperbolic Sine	SINH	$\sinh(arg)$	O	1 Real *4	Real *4
	DSINH	$\sinh(arg)$	O	1 Real *8	Real *8
Hyperbolic Cosine	COSH	$\cosh(arg)$	O	1 Real *4	Real *4
	DCOSH	$\cosh(arg)$	O	1 Real *8	Real *8
Hyperbolic Tangent	TANH	$\tanh(arg)$	O	1 Real *4	Real *4
	DTANH	$\tanh(arg)$	O	1 Real *8	Real *8
Error Function	ERF	$\frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$	O	1 Real *4	Real *4
	DERF	$\frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$	O	1 Real *8	Real *8
Complemented Error Function	ERFC	$1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$	O	1 Real *4	Real *4
	DERFC	$1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$	O	1 Real *8	Real *8
Gamma	GAMMA	$\int_0^\infty u^{x-1} e^{-u} du$	O	1 Real *4	Real *4
	DGAMMA	$\int_0^\infty u^{x-1} e^{-u} du$	O	1 Real *8	Real *8
Log-gamma	ALGAMA	$\ln \Gamma(arg)$	O	1 Real *4	Real *4
	DLGAMA	$\ln \Gamma(arg)$	O	1 Real *8	Real *8
Remaindering	MOD	$arg_1 \pmod{arg_2}$	I	2 Integer *4	Integer *4
	AMOD	$arg_1 \pmod{arg_2}$	I	2 Real *4	Real *4
	DMOD	$arg_1 \pmod{arg_2}$	I	2 Real *8	Real *8

<u>Function</u>	<u>Name</u>	<u>Definition</u>	<u>I or O*</u>	<u>Arguments (no., type)</u>	<u>Returned Function</u>
Absolute value	IABS	$ \arg $ $\sqrt{a^2 + b^2}$ for $a+bi = \arg$	I	1 Integer *4	Integer *4
	ABS		I	1 Real *4	Real *4
	DABS		I	1 Real *8	Real *8
	CABS		O	1 Complex *8	Real *4
Truncation	INT	Sign of arg times largest integer $\leq  \arg $	I	1 Real *4	Integer *4
	AINT		I	1 Real *4	Real *4
	IDINT		I	1 Real *8	Integer *4
Largest value	AMAX0	$\text{Max}(\arg_1, \arg_2, \dots)$	I	$\geq 2$ Integer *4	Real *4
	AMAX1		I	$\geq 2$ Real *4	Real *4
	MAX0		I	$\geq 2$ Integer *4	Integer *4
	MAX1		I	$\geq 2$ Real *4	Integer *4
	DMAX1		I	$\geq 2$ Real *8	Real *8
Smallest value	AMIN0	$\text{Min}(\arg_1, \arg_2, \dots)$	I	$\geq 2$ Integer *4	Real *4
	AMIN1		I	$\geq 2$ Real *4	Real *4
	MIN0		I	$\geq 2$ Integer *4	Integer *4
	MIN1		I	$\geq 2$ Real *4	Integer *4
	DMIN1		I	$\geq 2$ Real *8	Real *8
Float	FLOAT	Convert from integer to real	I	1 Integer *4	Real *4
	DFLOAT		I	1 Integer *4	Real *8
Fix	IFIX	Convert from real to integer	I	1 Real *4	Integer *4
	HFIX		I	1 Real *4	Integer *2
Transfer of sign	SIGN	Sign of $\arg_2$ times $ \arg_1 $	I	2 Real *4	Real *4
	ISIGN		I	2 Integer *4	Integer *4
	DSIGN		I	2 Real *8	Real *8
Positive difference	DIM	$\arg_1 - \min(\arg_1, \arg_2)$	I	2 Real *4	Real *4
	IDIM		I	2 Integer *4	Integer *4
Obtain most significant part of Real *8 argument	SNGL		I	1 Real *8	Real *4
Obtain real part of complex argument	REAL		I	1 Complex *8	Real *4
Obtain imaginary part of complex argument	AIMAG		I	1 Complex *8	Real *4
Express Real *4 argument in Real *8 form	DBLE		I	1 Real *4	Real *8
Express two real arguments in complex form	CMPLX	$C = \arg_1 + i\arg_2$	I	2 Real *4	Complex *8
	DCMPLX		I	2 Real *8	Complex *16
Obtain conjugate of complex argument	CONJG	$C = X - iY$ for $\arg = X + iY$	I	1 Complex *8	Complex *8
	DCONJG		I	1 Complex *16	Complex *16

\* I=in-line; O=out-of-line

**Service Subprograms**

Function	Name	Operands	I or O*	No. of ARGS.	Storage Estimate		Format
					Hex	Dec	
Turn all sense lights off or one sense light on	SLITE	(i) where i equals 0,1, 2,3,4	O	1	324	804	
Test a sense light, or record its status	SLITET	(i, j) where i equals 0, 1,2,3,4 and j equals 1 or 2	O	2	324	804	
Dump specified storage area and terminate processing	DUMP	(a <sub>1</sub> , b <sub>1</sub> , f <sub>1</sub> , ..., a <sub>n</sub> , b <sub>n</sub> , f <sub>n</sub> ) a=upper limit b=lower limit F=format specification 0 1 2 3 4 5 6 7 8 9	O	3	48	168	hex Logical *1 Logical *4 Integer *2 Integer *4 Real *4 Real *8 Complex *8 Complex *16 literal character
Dump specified storage area and resume processing	PDUMP	(a <sub>1</sub> , b <sub>1</sub> , f <sub>1</sub> , ..., a <sub>n</sub> , b <sub>n</sub> , f <sub>n</sub> ) where a, b and f are as in DUMP	O	3	48	168	
Terminate user programming; pass control to terminal; write message to SYSOUT	EXIT STOP	(issued without CALL)	O	0	1AC	428	
Write message to SYSOUT; continue processing user's program	PAUSE	(issued without CALL)	O	0	1AC	428	
Test and record status of exponent overflow indicators	OVERFL	j, where j =1 for >16 <sup>63</sup> =2 for no over-or-under-flow =3 for <16 <sup>-65</sup>	I				
Test and record status of divide check indicator	DVCHK	(j) J=1 if divide check on =2 if divide check off	I				

## Assembler Instruction Statements

### Symbol Definition

EQU equate symbol

### Data Definition

DC define constant  
 DS define storage space  
 CCW define channel command word

### Program Sectioning and Linking

START start assembly  
 CSECT identify control section  
 PSECT identify prototype control section  
 COM identify blank common control section  
 ENTRY identify entry point symbol  
 EXTRN identify external symbols  
 DSECT identify dummy control section

### Establishing Base Registers

USING use base address register  
 DROP drop base address registers

### Listing Control

TITLE identify assembly output  
 EJECT start new page  
 SPACE space listing  
 PRINT print optional data

### Program Control

ICTL source format control  
 ISEQ input sequence check  
 ORG set location counter  
 LTORG begin literal pool  
 CNOP conditional no-operation  
 COPY copy predefined source coding  
 END end assembly

### ASSEMBLER STATEMENTS

Name	Operation	Operands	Specified as
[symbol]	CCW	command code  ,data address  ,flag values  ,count	1 byte, absolute expression, right justified  absolute, relocatable, or complex expression  absolute expression  absolute expression
		Channel command operation codes and flag representations are summarized under "Appendix"	
	CNOP	byte alignment, word type $\left\{ \begin{array}{l} \text{double} \\ \text{single} \end{array} \right\}$	b=n, where n = 0,2,4, or 6 w=n, where n = 4 or 8
	COM	same as CSECT	default = standard common section
	COPY	symbol name	name of area to be copied

Name	Operation	Operands	Specified as					
[symbol]	CSECT	[public storage] [,read only storage] [,variable section length] [,privileged section] [,section includes SYS entry points]	PUBLIC .READONLY .VARIABLE .PRVLGD .SYSTEM					
[symbol]	DC	[duplication factor] constant-type [ length { bits } ] [scale] [exponent] constant[...]	(see *) L.n (see **) Ln (see **) Sn (see ***) En (see †) (see ††)					
'constant' or (constant)								
*	**	***	†	††	Constant specified as	Implied length	Implied align- ment	Trunca- ting/ padding side
Type	L.n Ln (n=)	Sn	En					
C	1 bit to 256 bytes				'up to 256 char- acters'			right
X					'hex digits'	as needed	byte	left
B					'binary digits'			left
F	1 bit to 8 bytes	-187 to +346	-85 to +75			4	word	left
H					'decimal digits,...'	2	half word	left
E		0 to 2L-2	-85 to +75			4	word	right
D		2				8	double word	right
<sup>1</sup> L is length constant; no negative scaling								
P	1 bit to 16 bytes				decimal digits ,...	as needed	byte	left
Z								
A	1 bit <sup>2</sup> to 4 bytes				(any expres- sion,...)	4	word	left
<sup>2</sup> Bit-length specifications with absolute expressions; relocatable A-type constants, 3 or 4 bytes; relocatable Y-type constants, 2 bytes								
V	3 or 4 bytes				(reloca- table ex- pression ,...)	4	word	
R	1 bit to 4 bytes				(external sym- bol,...)	4	word	
S	2 bytes				<sup>3</sup>	2	half word	
<sup>3</sup> one absolute or relocatable expression, or two absolute expressions								
Y	1 bit to 2 bytes <sup>4</sup>				(any expres- sion)	2	half word	left
<sup>4</sup> one absolute or relocatable expression, or two absolute expressions								

	DROP	reg1 [,reg2,....,reg16]	absolute value
[symbol]	DS	[duplication factor] [constant type] [ length { bits } { bytes } ] [scale] [exponent]	same as DC except as noted below
			1. Specification of 'constant' operand, optional 2. 'Constant' operand reserves space but does not store data 3. Maximum length for C and X constants, 65,535 bytes 4. Duplication factor of 0 forces alignment to assumed alignments in DC
[symbol]	DSFCT	none	
	END	[control transfer point]	relocatable or absolute expression default = first instr of CSECT
	ENTRY	entry point [...]	relocatable symbols
	EJECT	none	
[symbol]	EQU	self-defining term, previously defined symbol, or combining expression	
	EXTRN ICTL	external symbol [...] beginning source column [.ending source column] [.continue column]	relocatable symbols b=decimal digit, range, 1-40 e=decimal digit, range, 41-80; default=71 c=decimal digit, range, 2-40; default=16 if no ICTL is used, b=1, e=71, c=16.
	ISEQ	[sequence field - left col] [sequence field - right col]	decimal digits. default = no sequence check
[symbol]	LTORG	none	
	ORG	[new location counter address]	expression; default=current location counter position +1
	PRINT	printing option { listing } { no listing }  [ code { executable } { all } { none } ]  [ .constants { print full } { const } { print 8 } { bytes or } { less } ]	<u>ON</u> <u>OFF</u>  <u>GEN</u> <u>FULLGEN</u> <u>NOGEN</u>  <u>DATA</u> <u>NODATA</u>
	SPACE	[no. of lines to be spaced]	decimal digits. default = 1
[symbol]	START	[initial location ctr. address]	self-defining term. default = 0
[symbol]	TITLE	'characters'	to 100 characters
		Note: symbol may be from 1 to 4 alphameric characters	
	USING	base value, reg1 [,reg2,....,reg16]	absolute or relocatable value, absolute expression



### Basic Instruction Formats

Basic Machine Format		Assembler Operand Field Format	Applicable Instructions														
RR	<table border="1"> <tr> <td>8</td> <td>4</td> <td>4</td> </tr> <tr> <td>Operation Code</td> <td>R1</td> <td>R2</td> </tr> </table>	8	4	4	Operation Code	R1	R2	R1,R2	All RR instructions except SPM and SVC								
	8	4	4														
	Operation Code	R1	R2														
<table border="1"> <tr> <td>8</td> <td>4</td> <td><del>4</del></td> </tr> <tr> <td>Operation Code</td> <td>R1</td> <td><del>R2</del></td> </tr> </table>	8	4	<del>4</del>	Operation Code	R1	<del>R2</del>	R1	SPM									
8	4	<del>4</del>															
Operation Code	R1	<del>R2</del>															
<table border="1"> <tr> <td>8</td> <td>8</td> </tr> <tr> <td>Operation Code</td> <td>I</td> </tr> </table>	8	8	Operation Code	I	I	SVC (See notes 1,6,8)											
8	8																
Operation Code	I																
RX	<table border="1"> <tr> <td>8</td> <td>4</td> <td>4</td> <td>4</td> <td>12</td> </tr> <tr> <td>Operation Code</td> <td>R1</td> <td>X2</td> <td>B2</td> <td>D2</td> </tr> </table>	8	4	4	4	12	Operation Code	R1	X2	B2	D2	R1,D2 (X2,B2) R1,S2 (X2)	All RX instructions (See notes 1-4,7)				
8	4	4	4	12													
Operation Code	R1	X2	B2	D2													
RS	<table border="1"> <tr> <td>8</td> <td>4</td> <td>4</td> <td>4</td> <td>12</td> </tr> <tr> <td>Operation Code</td> <td>R1</td> <td>R3</td> <td>B2</td> <td>D2</td> </tr> </table>	8	4	4	4	12	Operation Code	R1	R3	B2	D2	R1,R3, D2(B2) R1,R3,S2	BXH,BXLE LM,STM				
	8	4	4	4	12												
Operation Code	R1	R3	B2	D2													
<table border="1"> <tr> <td>8</td> <td>4</td> <td><del>4</del></td> <td>4</td> <td>12</td> </tr> <tr> <td>Operation Code</td> <td>R1</td> <td><del>R3</del></td> <td>B2</td> <td>D2</td> </tr> </table>	8	4	<del>4</del>	4	12	Operation Code	R1	<del>R3</del>	B2	D2	R1,D2(B2) R1,S2	All shift instructions (See notes 1-3,7,8)					
8	4	<del>4</del>	4	12													
Operation Code	R1	<del>R3</del>	B2	D2													
SI	<table border="1"> <tr> <td>8</td> <td>8</td> <td>12</td> </tr> <tr> <td>Operation Code</td> <td>I2</td> <td>B1 D1</td> </tr> </table>	8	8	12	Operation Code	I2	B1 D1	D1(B1), I2	All SI instructions except LPSW,SSM, HIO,SIO TIO,TCH,TS								
	8	8	12														
Operation Code	I2	B1 D1															
<table border="1"> <tr> <td>8</td> <td><del>8</del></td> <td>4</td> <td>12</td> </tr> <tr> <td>Operation Code</td> <td><del>I2</del></td> <td>B1</td> <td>D1</td> </tr> </table>	8	<del>8</del>	4	12	Operation Code	<del>I2</del>	B1	D1	D1(B1) S1	LPSW,SSM, HIO,SIO, TIO,TCH,TS (See notes 2,3,6-8)							
8	<del>8</del>	4	12														
Operation Code	<del>I2</del>	B1	D1														
SS	<table border="1"> <tr> <td>8</td> <td>4</td> <td>4</td> <td>4</td> <td>12</td> <td>4</td> <td>12</td> </tr> <tr> <td>Operation Code</td> <td>L1</td> <td>L2</td> <td>B1</td> <td>D1</td> <td>B2</td> <td>D2</td> </tr> </table>	8	4	4	4	12	4	12	Operation Code	L1	L2	B1	D1	B2	D2	D1(L1,B1), D2(L2,B2) S1(L1), S2(L2)	PACK,UNPK, MVO,AP, CP,DP,MP, SP,ZAP
	8	4	4	4	12	4	12										
Operation Code	L1	L2	B1	D1	B2	D2											
<table border="1"> <tr> <td>8</td> <td>8</td> <td>4</td> <td>12</td> <td>4</td> <td>12</td> </tr> <tr> <td>Operation Code</td> <td>L</td> <td>B1</td> <td>D1</td> <td>B2</td> <td>D2</td> </tr> </table>	8	8	4	12	4	12	Operation Code	L	B1	D1	B2	D2	D1(L,B1), D2(B2) S1(L),S2	NC,OC,XC, CLC,MVC, MVN,MVZ, TR,TRT, ED,EDMK (See notes 2,3,5,7)			
8	8	4	12	4	12												
Operation Code	L	B1	D1	B2	D2												

1. R1, R2, and R3—absolute expressions; specify general registers 0 through 15; floating point registers 0, 2, 4, and 6.
2. D1 and D2—absolute expressions; specify displacement values from 0 to 4095.
3. B1 and B2—absolute expressions; specify base registers 0 - 15.
4. X2—absolute expression; specifies index registers 1 - 15. X2 = 0 specifies no indexing. If B2 specified, X2 must be included, or omitted either by specifying 0 or preceding B2 with comma.  
Ex: L 2,48(,5)
5. L, L1, and L2—absolute expressions; specify field lengths: L from 0 to 256; L1 and L2 from 0 to 16. Assembled values will be one less than specified (except that zero length will be assembled if 0 is specified).
6. I and I2—absolute expressions; provide immediate data; values from 0 to 255.
7. S1 and S2—absolute or relocatable expressions; specify address.
8. RR, RS and SI—fields shown crossed out with X's; not examined during execution; not written in symbolic operand, but assembled as binary 0s.

### Standard Instruction Set

Mnemonic Code	Instruction	Operation Code	Basic Machine Format	Operand Field Format
A	Add	5A	RX	R1.D2(X2.B2)
AD	Add normalized, long	6A	RX	R1.D2(X2.B2)
ADD <sup>4</sup>	Add normalized, double	66	RX	R1.D2(X2.B2),
ADDR <sup>4</sup>	Add normalized, double	26	RR	R1,R2
ADR	Add normalized, long	2A	RR	R1,R2
AE	Add normalized, short	7A	RX	R1.D2(X2.B2)
AER	Add normalized, short	3A	RR	R1,R2
AH	Add half-word	4A	RX	R1.D2(X2.B2)
AL	Add logical	5E	RX	R1.D2(X2.B2)
ALR	Add logical	1E	RR	R1,R2
AP	Add decimal	FA	SS	D1(L1.B1),D2(L2.B2)
AR	Add	1A	RR	R1,R2
AU	Add unnormalized, short	7E	RX	R1.D2(X2.B2)
AUR	Add unnormalized, short	3E	RR	R1,R2
AW	Add unnormalized, long	6E	RX	R1.D2(X2.B2)
AWR	Add unnormalized, long	2E	RR	R1,R2
AX <sup>4</sup>	Add normalized, mixed	76	RX	R1.D2(X2.B2)
BAL <sup>1</sup>	Branch and link	45	RX	R1.D2(X2.B2)
BALR <sup>1</sup>	Branch and link	05	RR	R1,R2
BAS	Branch and store	4D	RX	R1.D2(X2.B2)
BASR	Branch and store	0D	RR	R1,R2
BC	Branch on condition	47	RX	M1.D2(X2.B2)
BCR	Branch on condition	07	RR	M1,R2
BCT <sup>1</sup>	Branch on count	46	RX	R1.D2(X2.B2)
BCTR <sup>1</sup>	Branch on count	06	RR	R1,R2
BXH <sup>1</sup>	Branch on index high	86	RS	R1.R3.D2(B2)
BXLE <sup>1</sup>	Branch on index low or equal	87	RS	R1.R3.D2(B2)
C	Compare algebraic	59	RX	R1.D2(X2.B2)
CD	Compare, long	69	RX	R1.D2(X2.B2)
CDR	Compare, long	29	RR	R1,R2
CE	Compare,short	79	RX	R1.D2(X2.B2)
CER	Compare,short	39	RR	R1,R2
CH	Compare half-word	49	RX	R1.D2(X2.B2)
CL	Compare logical	55	RX	R1.D2(X2.B2)
CLC	Compare logical	D5	SS	D1(L1.B1),D2(B2)
CLI	Compare logical immediate	95	SI	D1(B1),I2
CLR	Compare logical	15	RR	R1,R2
CP	Compare decimal	F9	SS	D1(L1.B1),D2(L2.B2)
CR	Compare algebraic	19	RR	R1,R2
CVB	Convert to binary	4F	RX	R1.D2(X2.B2)
CVD	Convert to decimal	4E	RX	R1.D2(X2.B2)

Mnemonic Code	Instruction	Operation Code	Basic Machine Format	Operand Field Format
D	Divide	5D	RX	R1,D2(X2,B2)
DD	Divide, long	6D	RX	R1,D2(X2,B2)
DDR	Divide, long	2D	RR	R1,R2
DE	Divide, short	7D	RX	R1,D2(X2,B2)
DER	Divide, short	3D	RR	R1,R2
DP	Divide decimal	FD	SS	D1(L1,B1),D2(L2,B2)
DR	Divide	1D	RR	R1,R2
ED	Edit	DE	SS	D1(L,B1),D2(B2)
EDMK	Edit and mark	DF	SS	D1(L,B1),D2(B2)
EX	Execute	44	RX	R1,D2(X2,B2)
HDR	Halve, long	24	RR	R1,R2
HER	Halve, short	34	RR	R1,R2
HIO	Halt I/O	9E	SI	D1(B1)
IC	Insert character	43	RX	R1,D2(X2,B2)
ISK <sup>2</sup>	Insert storage key	09	RR	R1,R2
L	Load	58	RX	R1,D2(X2,B2)
LA <sup>3</sup>	Load address	41	RX	R1,D2(X2,B2)
LCDR	Load complement, long	23	RR	R1,R2
LCER	Load complement, short	33	RR	R1,R2
LCR	Load complement	13	RR	R1,R2
LD	Load, long	68	RX	R1,D2(X2,B2)
LDR	Load, long	28	RR	R1,R2
LE	Load, short	78	RX	R1,D2(X2,B2)
LER	Load, short	38	RR	R1,R2
LH	Load half-word	48	RX	R1,D2(X2,B2)
LM	Load multiple	98	RS	R1,R3,D2(B2)
LMC <sup>2</sup>	Load multiple control	B8	RS	R1,R3,D2(B2)
LNDR	Load negative, long	21	RR	R1,R2
LNER	Load negative, short	31	RR	R1,R2
LNR	Load negative	11	RR	R1,R2
LPDR	Load positive, long	20	RR	R1,R2
LPER	Load positive, short	30	RR	R1,R2
LPR	Load positive	10	RR	R1,R2
LPSW <sup>2</sup>	Load PSW	82	SI	D1(B1)
LR	Load	18	RR	R1,R2
LRA <sup>2</sup>	Load real address	B1	RX	R1,D2(X2,B2)
LTDR	Load and test, long	22	RR	R1,R2
LTER	Load and test, short	32	RR	R1,R2
LTR	Load and test	12	RR	R1,R2
LX <sup>4</sup>	Load mixed	74	RX	R1,D2(X2,B2)
M	Multiply	5C	RX	R1,D2(X2,B2)
MD	Multiply, long	6C	RX	R1,D2(X2,B2)
MDD <sup>4</sup>	Multiply normalized, double	65	RX	R1,D2(X2,B2)
MDDR <sup>4</sup>	Multiply normalized, double	25	RR	R1,R2
MDR	Multiply, long	2C	RR	R1,R2
ME	Multiply, short	7C	RX	R1,D2(X2,B2)
MER	Multiply, short	3C	RR	R1,R2
MH	Multiply half-word	4C	RX	R1,D2(X2,B2)
MP	Multiply decimal	FC	SS	D1(L1,B1),D2(L2,B2)
MR	Multiply	1C	RR	R1,R2
MVC	Move characters	D2	SS	D1(L,B1),D2(B2)
MVI	Move immediate	92	SI	D1(B1),I2
MVN	Move numerics	D1	SS	D1(L,B1),D2(B2)
MVO	Move with offset	F1	SS	D1(L1,B1),D2(L2,B2)
MVZ	Move zones	D3	SS	D1(L,B1),D2(B2)
N	AND logical	54	RX	R1,D2(X2,B2)
NC	AND logical	D4	SS	D1(L,B1),D2(B2)
NI	AND logical immediate	94	SI	D1(B1),I2
NR	AND logical	14	RR	R1,R2
O	OR logical	56	RX	R1,D2(X2,B2)
OC	OR logical	D6	SS	D1(L,B1),D2(B2)
OI	OR logical immediate	96	SI	D1(B1),I2
OR	OR logical	16	RR	R1,R2

Mnemonic Code	Instruction	Operation Code	Basic Machine Format	Operand Field Format
PACK	Pack	F2	SS	D1(L1,B1),D2(L2,B2)
RDD <sup>2</sup>	Read direct	85	SI	D1(B1),I2
S	Subtract	5B	RX	R1,D2(X2,B2)
SD	Subtract normalized, long	6B	RX	R1,D2(X2,B2)
SDD <sup>4</sup>	Subtract normalized, double	67	RX	R1,D2(X2,B2)
SDDR <sup>4</sup>	Subtract normalized, double	27	RR	R1,R2
SDR	Subtract normalized, long	2B	RR	R1,R2
SE	Subtract normalized, short	7B	RX	R1,D2(X2,B2)
SER	Subtract normalized, short	3B	RR	R1,R2
SH	Subtract half-word	4B	RX	R1D2(X2,B2)
SIO <sup>2</sup>	Start I/O	9C	SI	D1(B1)
SL	Subtract logical	5F	RX	R1,D2(X2,B2)
SLA	Shift left, single algebraic	8B	RS	R1,D2(B2)
SLDA	Shift left, double algebraic	8F	RS	R1,D2(B2)
SLDC	Shift left, double logical	8D	RS	R1,D2(B2)
SLL	Shift left, single logical	89	RS	R1,D2(B2)
SLR	Subtract logical	1F	RR	R1,R2
SLT <sup>4</sup>	Search list	A2	RS	M1,D2(L2,B2)
SP	Subtract decimal	FB	SS	D1(L1,B1),D2(L2,B2)
SPM	Set program mask	04	RR	R1
SR	Subtract	1B	RR	R1,R2
SRA	Shift right, single algebraic	8A	RS	R1,D2(B2)
SRDA	Shift right, double algebraic	8F	RS	R1,D2(B2)
SRDL	Shift right, double logical	8C	RS	R1,D2(B2)
SRL	Shift right, single logical	88	RS	R1,D2(B2)
SSK <sup>2</sup>	Set system key	08	R	R1,R2
SSM <sup>2</sup>	Set system mask	80	SI	D1(B1)
ST	Store	50	RX	R1,D2(X2,B2)
STC	Store character	42	RX	R1,D2(X2,B2)
STD	Store long	60	RX	R1,D2(X2,B2)
STE	Store short	70	RX	R1,D2(X2,B2)
STH	Store half-word	40	RX	R1,D2(X2,B2)
STM	Store multiple	90	RS	R1,R3,D2(B2)
STM <sup>2</sup>	Store multiple control	B0	RS	R1,R3,D2(B2)
STRD <sup>4</sup>	Store rounded, long	61	RX	R1,D2(X2,B2)
STRE <sup>4</sup>	Store rounded, short	71	RX	R1,D2(X2,B2)
SU	Subtract unnormalized, short	7F	RX	R1,D2(X2,B2)
SUR	Subtract unnormalized, short	3F	RR	R1,R2
SVC	Supervisor call	0A	RR	I
SW	Subtract unnormalized, long	6F	RX	R1,D2(X2,B2)
SWR	Subtract unnormalized, long	2F	RR	R1,R2
SX <sup>4</sup>	Subtract normalized, mixed	77	RX	R1,D2(X2,B2)
TCH <sup>2</sup>	Test channel	9F	SI	D1(B1)
TIO <sup>2</sup>	Test I/O	9D	SI	D1(B1)
TM	Test under mask	91	SI	D1(B1),I2
TR	Translate	DC	SS	D1(L1,B1),D2(B2)
TRT	Translate and test	DD	SS	D1(L1,B1),D2(B2)
TS	Test and set	93	SI	D1(B1)
UNPK	Unpack	F3	SS	D1(L1,B1),D2(L2,B2)
WRD <sup>2</sup>	Write direct	84	SI	D1(B1),I2
X	Exclusive OR	57	RX	R1,D2(X2,B2)
XC	Exclusive OR	D7	SS	D1(L1,B1),D2(B2)
XI	Exclusive OR, immediate	97	SI	D1(B1),I2
XR	Exclusive OR	17	RR	R1,R2
ZAP	Zero and add decimal	F8	SS	D1(L1,B1),D2(L2,B2)

<sup>1</sup> Limited to 24-bit addressing capability; exercise caution when using in 32-bit mode.

<sup>2</sup> Privileged instructions.

<sup>3</sup> Users in 24-bit mode must not exceed 24-bit limitation when loading address or using LA adding technique.

<sup>4</sup> RPQ (request price quote) instructions. All users are restricted from using these mnemonic codes as macro instruction names.

### Extended Mnemonic Instruction Codes

Extended Code	Instruction	Machine Instruction
<b>B</b> D2(X2,B2)	Branch unconditional	BC 15,D2(X2,B2)
<b>BR</b> R2	Branch unconditional (RR format)	BCR 15,R2
<b>NOP</b> D2(X2,B2)	No operation	BC 0,D2(X2,B2)
<b>NOPR</b> R2	No operation (RR format)	BCR 0,R2
<i>Used After Compare Instructions</i>		
<b>BH</b> D2(X2,B2)	Branch on high	BC 2,D2(X2,B2)
<b>BL</b> D2(X2,B2)	Branch on low	BC 4,D2(X2,B2)
<b>BE</b> D2(X2,B2)	Branch on equal	BC 8,D2(X2,B2)
<b>BNH</b> D2(X2,B2)	Branch on not high	BC 13,D2(X2,B2)
<b>BNL</b> D2(X2,B2)	Branch on not low	BC 11,D2(X2,B2)
<b>BNE</b> D2(X2,B2)	Branch on not equal	BC 7,D2(X2,B2)
<i>Used After Arithmetic Instructions</i>		
<b>BO</b> D2(X2,B2)	Branch on overflow	BC 1,D2(X2,B2)
<b>BP</b> D2(X2,B2)	Branch on plus	BC 2,D2(X2,B2)
<b>BM</b> D2(X2,B2)	Branch on minus	BC 4,D2(X2,B2)
<b>BZ</b> D2(X2,B2)	Branch on 0	BC 8,D2(X2,B2)
<i>Used After Test Under Mask Instructions</i>		
<b>BO</b> D2(X2,B2)	Branch if 1s	BC 1,D2(X2,B2)
<b>BM</b> D2(X2,B2)	Branch if mixed	BC 4,D2(X2,B2)
<b>BZ</b> D2(X2,B2)	Branch if 0s	BC 8,D2(X2,B2)

Assembler User Macro Instructions			
DATA SET MANAGEMENT		PROGRAM CONTROL MANAGEMENT	
<b>Define Data Set</b>		<b>Virtual Storage Management</b>	
DDEF	CDD	GETMAIN DCLASS*	
DCB	DEL	FREEMAIN RSPRV*	
CAT	DCBD	CSTORE LSCHP*	
FINDDS*	FINDJFCB*	CKCLS*	
<b>Connect Data Set to System</b>		<b>Program Linking and Loading</b>	
OPEN	DUOPEN (VAM only)	ADCOND DELETE	
<b>Access Data Set</b>		ADCON SAVE	
<u>VSAM</u>	<u>VISAM</u>	VPAM	ARM RETURN
GET	GET	FIND	CALL DELETE*
PUT	PUT	STOW	LOAD DLINK*
PUTX	READ	<u>IOREQ</u>	ENTER* RESUME*
SETL	WRITE	IOREQ	LIBESRCH* STORE*
<u>BSAM</u>	SETL	IOREQ	<b>Interruption Handling</b>
READ	ESLTL	CHECK	SIR SIEC
WRITE	DELREC	VCCW	SPEC DIR
CHECK	RELEX		SSFC SAI
DQDFCB	<u>QSAM</u>		SEEC RAE
GETPOOL	GET		SAFC INTINO
GETBUF	PUT		STEC CLATI
FREEBUF	PUTX		USATT AITD
FREEPOOL	RELEASE		ATPOL* IIT*
BSP	TRUNC		PTI*
CNTRL	CNTRL		<b>Transfer to Command Mode</b>
FEOV	SETL		PAUSE ABEND
POINT			COMMAND OBEY
NOTE			EXIT RTRN*
			CLIC* CLIP*
<b>Manipulate Entire Data Set</b>		<b>Communications with SYSIN/SYSOUT</b>	
<i>Copy Data Set</i>	<i>Both Output</i>	GATRD GTWRC MCAST	
CDS	PR	GATWR GTWSR	
	PU	GTWAR SYSIN	
	WT	MSGWR PRMPT	
<b>Catalog Data Set Attributes</b>		<b>Communications with Operator and Log</b>	
CAT		WTO WTL	
DEL		WTOR WTOA	
<b>Disconnect Data Set from System</b>		<b>Task/Timing Maintenance</b>	
CLOSE	DUPCLOSE (VAM only)	STIMER EBCDIME USAGE	
<b>Remove Data Set from System</b>		TTIMER REDTIM*	
ERASE		<b>Command Creation</b>	
REL		BPKD GDV	
<b>Language Processing</b>		<b>System-Oriented Macro Instructions</b>	
LPCEDIT		AUXPG* VSEND*	
LPCINIT		AWAIT* XTRSYS*	
		PIREC* XTRTM*	
		PULSE* XTRCT*	
		PRESENT*	
		RDI*	
		TSEND*	
		TWAIT*	
		VSEND*	
		XTRXTS*	

\*Available to nonprivileged users, but intended primarily for system programmers.

## Operand Expressions

Operands of macro instructions may be expressed in several ways:

alphabetic	A - Z \$ @ #
alphanumeric	A - Z \$ @ # 0-9
special characters	+ - , = . * ( ) / & and blank
special symbols	1-6 alphanumeric characters and + - . * ' / &
relocatable expression	changes when program is located
absolute expression	value remains constant when program is re-located
explicit address	displacement (index register, base register)
implicit address	symbolic address [(index register)]
register notation	absolute expression enclosed in parentheses that evaluates from 0-15.
text	character string enclosed in apostrophes; embedded blanks and special characters permitted; two apostrophes or two ampersands represent one apostrophe or one ampersand in character string; text operand limited to 255 characters, including enclosing apostrophes
characters	character string not enclosed in apostrophes; no embedded commas or blanks; two apostrophes or two ampersands represent one apostrophe or one ampersand
oplist	series of operands or list in text format or coded somewhere else and addressed with addr expressions; when operand list is coded elsewhere, operand string ends with end-of-message code, hexadecimal 27

## Expression Abbreviations

Abbreviations used when more than one operand expression represents one operand value:

addr	relocatable expression, register notation
addrx	register notation, explicit address, implied address
addx	explicit address, implied address
value	register notation, absolute expression
integer	decimal integer, absolute expression

Assembler Macro Instruction Formats

Name	Operation	Operands
symbol	ABEND	compcode } value { 1 } }          { 2 } }          { 3 } (0) } ,message { text } } addrx } } (1) }
symbol	ADCON	type code { CALL } LOAD } DELETE } [.EP=symbol] } IMPLICIT } } INTERNAL }
		[.LDERR= { pass return codes - CODE } } take error exit - ERR }
		[.DELOPT= { specified module only - SMO } } all associated modules - SDM }
		[.HSHTAB= { XPDS } } NORM }
	ADCOND	none
symbol	AETD	[ ( { ep-symbol, sa-symbol } ... ) ]
{symbol}	ARM	loc-addrx,extref-addrx  return data: reg 15 = address of initialized or altered adcon group
symbol	BPKD	ep-symbol [ ,(param-relexp,...) ]

**BSAM (basic sequential access method)**

{symbol}	READ	decb-symbol,type { SF SB } ,dcb-addr ,area-addr[,length { 'S' value } ]
{symbol}	WRITE	decb-symbol,type-SF,dcb-addr,area-addr [,length { 'S' value } ]
{symbol}	CHECK	decb { addrx } } (1) }
{symbol}	DQDECB	decb { addrx } } (1) }
		return data: reg 0 = number of unchecked DECBs in queue reg 1 = pointer to list of unchecked DECBs
{symbol}	GETPOOL	dcb { addrx } , { number-value,length-value } } (1) }                                    } (0) }
{symbol}	GETBUF	dcb { addrx } ,register-absexp } (1) }
{symbol}	FREEBUF	dcb { addrx } ,register-absexp } (1) }
{symbol}	FREEPOOL	dcb { addrx } } (1) }





[symbol]	CDD	<p>oplist { text }           { addr }</p> <p>where oplist equals: dsname-name[ ,DDNAME=ddname-symbol,...]</p> <p>return data: reg 15 = code</p> <table border="0"> <tr> <td style="text-align: right;">(hex)</td> <td style="text-align: left;"><u>significance</u></td> </tr> <tr> <td style="text-align: right;">00</td> <td>normal</td> </tr> <tr> <td style="text-align: right;">04</td> <td>invalid dsname</td> </tr> <tr> <td style="text-align: right;">08</td> <td>invalid ddname</td> </tr> <tr> <td style="text-align: right;">0C</td> <td>ddname not in data set</td> </tr> <tr> <td style="text-align: right;">10</td> <td>error return from DDEF</td> </tr> <tr> <td style="text-align: right;">14</td> <td>not line data set</td> </tr> </table>	(hex)	<u>significance</u>	00	normal	04	invalid dsname	08	invalid ddname	0C	ddname not in data set	10	error return from DDEF	14	not line data set														
(hex)	<u>significance</u>																													
00	normal																													
04	invalid dsname																													
08	invalid ddname																													
0C	ddname not in data set																													
10	error return from DDEF																													
14	not line data set																													
[symbol]	CDS	<p>oplist { text }           { addr }</p> <p>where optlist equals: dsname1-symbol[(member name,...)] ,dsname2-symbol[(member name,...)]</p> <p>[ ,ERASE= { Y }           { N } ]</p> <p>[ ,line-integer ] [ ,increment-integer ]</p> <p>[ ,REPLACE= { I }               { R } ]</p> <p>return data: reg 15 = code</p> <table border="0"> <tr> <td style="text-align: right;">(hex)</td> <td style="text-align: left;"><u>significance</u></td> </tr> <tr> <td style="text-align: right;">00</td> <td>normal</td> </tr> <tr> <td style="text-align: right;">04</td> <td>invalid input parameters</td> </tr> <tr> <td style="text-align: right;">08</td> <td>original data set name not in catalog or task definition table (TDT)</td> </tr> <tr> <td style="text-align: right;">0C</td> <td>data set not in catalog, no DDEF macro instruc- tion or command executed</td> </tr> <tr> <td style="text-align: right;">10</td> <td>JFCB for original data set not consistent with JFCB for new data set</td> </tr> <tr> <td style="text-align: right;">14</td> <td>member name not given for partitioned data set</td> </tr> <tr> <td style="text-align: right;">18</td> <td>user does not have write access for new data set</td> </tr> <tr> <td style="text-align: right;">1C</td> <td>original data set not VAM or SAM</td> </tr> <tr> <td style="text-align: right;">20</td> <td>data set not of direct- access; command ignored</td> </tr> <tr> <td style="text-align: right;">24</td> <td>new data set member name already in POD</td> </tr> <tr> <td style="text-align: right;">28</td> <td>data set copied, old data set not erased; user does not have proper access</td> </tr> <tr> <td style="text-align: right;">2C</td> <td>data set copied, new data set not renumbered, not a line data set</td> </tr> <tr> <td style="text-align: right;">30</td> <td>data set copied and re- numbered, old data set not erased, user does not have proper access</td> </tr> </table>	(hex)	<u>significance</u>	00	normal	04	invalid input parameters	08	original data set name not in catalog or task definition table (TDT)	0C	data set not in catalog, no DDEF macro instruc- tion or command executed	10	JFCB for original data set not consistent with JFCB for new data set	14	member name not given for partitioned data set	18	user does not have write access for new data set	1C	original data set not VAM or SAM	20	data set not of direct- access; command ignored	24	new data set member name already in POD	28	data set copied, old data set not erased; user does not have proper access	2C	data set copied, new data set not renumbered, not a line data set	30	data set copied and re- numbered, old data set not erased, user does not have proper access
(hex)	<u>significance</u>																													
00	normal																													
04	invalid input parameters																													
08	original data set name not in catalog or task definition table (TDT)																													
0C	data set not in catalog, no DDEF macro instruc- tion or command executed																													
10	JFCB for original data set not consistent with JFCB for new data set																													
14	member name not given for partitioned data set																													
18	user does not have write access for new data set																													
1C	original data set not VAM or SAM																													
20	data set not of direct- access; command ignored																													
24	new data set member name already in POD																													
28	data set copied, old data set not erased; user does not have proper access																													
2C	data set copied, new data set not renumbered, not a line data set																													
30	data set copied and re- numbered, old data set not erased, user does not have proper access																													

(continued)

Name	Operation	Operand																																																																																																																																																																																																																															
		34 data set copied and original erased, new data set not renumbered, not a line data set 38 data set copied, new data set not renumbered, old data set not erased																																																																																																																																																																																																																															
[symbol]	CLATT	none																																																																																																																																																																																																																															
[symbol]	CLOSE	( { dcb-addr, [opt]{ REREAD } } { LEAVE } } , ... ), TYPE=T ]																																																																																																																																																																																																																															
[symbol]	COMMAND	message { text } { addrx } (1)																																																																																																																																																																																																																															
[symbol]	CSTORE	module-symbol, epname-symbol, address-addr, length-value, attribute-value  return data: reg 15 = Code Significance 00 normal return 04 module name or entry point name already in use																																																																																																																																																																																																																															
	DCB	<table border="1"> <thead> <tr> <th rowspan="2"></th> <th colspan="7">Applicable Access Methods</th> </tr> <tr> <th>VSAM</th> <th>VISAM</th> <th>VPAM</th> <th>BSAM</th> <th>QSAM</th> <th>IOREQ</th> <th></th> </tr> </thead> <tbody> <tr> <td>[.DDNAME=symbol]</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> </tr> <tr> <td>[.DSORG=code]</td> <td>VS VSP</td> <td>VI VIP</td> <td>VP</td> <td>PS</td> <td>PS</td> <td></td> <td>RX</td> </tr> <tr> <td>[.RECFM=code]</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> </tr> <tr> <td>[.LRECL=absexp]</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td></td> <td></td> </tr> <tr> <td>[.EODAD=symbol]</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td></td> <td></td> </tr> <tr> <td>[.OPTCD=code]</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td></td> <td></td> </tr> <tr> <td>[.SYNAD=symbol]</td> <td></td> <td>X</td> <td>X*</td> <td>X</td> <td>X</td> <td></td> <td>X</td> </tr> <tr> <td>[.PAD=absexp]</td> <td></td> <td>X</td> <td>X*</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>[.RKP=absexp]</td> <td></td> <td>X</td> <td>X*</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>[.DEV=code]</td> <td></td> <td>1</td> <td>1</td> <td>X</td> <td>X</td> <td></td> <td>X</td> </tr> <tr> <td>[.KEYLEN=absexp]</td> <td></td> <td>X</td> <td>X*</td> <td>X</td> <td></td> <td></td> <td>X</td> </tr> <tr> <td>[.TRTCH=code]</td> <td></td> <td></td> <td></td> <td>X</td> <td>X</td> <td></td> <td>X</td> </tr> <tr> <td>[.PRTSP=absexp]</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>X</td> </tr> <tr> <td>[.MODE=code]</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>X</td> </tr> <tr> <td>[.STACK=absexp]</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>X</td> </tr> <tr> <td>[.MACRF=code]</td> <td></td> <td></td> <td></td> <td>X</td> <td>X</td> <td></td> <td></td> </tr> <tr> <td>[.BLKSIZE=absexp]</td> <td></td> <td></td> <td></td> <td>X</td> <td>X</td> <td></td> <td></td> </tr> <tr> <td>[.IMSK=code]</td> <td></td> <td></td> <td></td> <td>X</td> <td>X</td> <td></td> <td></td> </tr> <tr> <td>[.EXLST=symbol]</td> <td></td> <td></td> <td></td> <td>X</td> <td>X</td> <td></td> <td></td> </tr> <tr> <td>[.NCP=absexp]</td> <td></td> <td></td> <td></td> <td>X</td> <td></td> <td></td> <td>X</td> </tr> <tr> <td>[.BUFNO=absexp]</td> <td></td> <td></td> <td></td> <td>X</td> <td></td> <td></td> <td></td> </tr> <tr> <td>[.BFALN=code]</td> <td></td> <td></td> <td></td> <td>X</td> <td></td> <td></td> <td></td> </tr> <tr> <td>[.BUFL=absexp]</td> <td></td> <td></td> <td></td> <td>X</td> <td></td> <td></td> <td></td> </tr> <tr> <td>[.BFTEK=code]</td> <td></td> <td></td> <td></td> <td>X</td> <td></td> <td></td> <td></td> </tr> <tr> <td>[.BUFCB=addr]</td> <td></td> <td></td> <td></td> <td>X</td> <td></td> <td></td> <td></td> </tr> <tr> <td>[.EROPT=code]</td> <td></td> <td></td> <td></td> <td></td> <td>X</td> <td></td> <td></td> </tr> </tbody> </table> <p>* = VISAM members of partitioned data set            1. = some established value assumed by system</p>		Applicable Access Methods							VSAM	VISAM	VPAM	BSAM	QSAM	IOREQ		[.DDNAME=symbol]	X	X	X	X	X	X	X	[.DSORG=code]	VS VSP	VI VIP	VP	PS	PS		RX	[.RECFM=code]	X	X	X	X	X	X	X	[.LRECL=absexp]	X	X	X	X	X			[.EODAD=symbol]	X	X	X	X	X			[.OPTCD=code]	X	X	X	X	X			[.SYNAD=symbol]		X	X*	X	X		X	[.PAD=absexp]		X	X*					[.RKP=absexp]		X	X*					[.DEV=code]		1	1	X	X		X	[.KEYLEN=absexp]		X	X*	X			X	[.TRTCH=code]				X	X		X	[.PRTSP=absexp]							X	[.MODE=code]							X	[.STACK=absexp]							X	[.MACRF=code]				X	X			[.BLKSIZE=absexp]				X	X			[.IMSK=code]				X	X			[.EXLST=symbol]				X	X			[.NCP=absexp]				X			X	[.BUFNO=absexp]				X				[.BFALN=code]				X				[.BUFL=absexp]				X				[.BFTEK=code]				X				[.BUFCB=addr]				X				[.EROPT=code]					X		
	Applicable Access Methods																																																																																																																																																																																																																																
	VSAM	VISAM	VPAM	BSAM	QSAM	IOREQ																																																																																																																																																																																																																											
[.DDNAME=symbol]	X	X	X	X	X	X	X																																																																																																																																																																																																																										
[.DSORG=code]	VS VSP	VI VIP	VP	PS	PS		RX																																																																																																																																																																																																																										
[.RECFM=code]	X	X	X	X	X	X	X																																																																																																																																																																																																																										
[.LRECL=absexp]	X	X	X	X	X																																																																																																																																																																																																																												
[.EODAD=symbol]	X	X	X	X	X																																																																																																																																																																																																																												
[.OPTCD=code]	X	X	X	X	X																																																																																																																																																																																																																												
[.SYNAD=symbol]		X	X*	X	X		X																																																																																																																																																																																																																										
[.PAD=absexp]		X	X*																																																																																																																																																																																																																														
[.RKP=absexp]		X	X*																																																																																																																																																																																																																														
[.DEV=code]		1	1	X	X		X																																																																																																																																																																																																																										
[.KEYLEN=absexp]		X	X*	X			X																																																																																																																																																																																																																										
[.TRTCH=code]				X	X		X																																																																																																																																																																																																																										
[.PRTSP=absexp]							X																																																																																																																																																																																																																										
[.MODE=code]							X																																																																																																																																																																																																																										
[.STACK=absexp]							X																																																																																																																																																																																																																										
[.MACRF=code]				X	X																																																																																																																																																																																																																												
[.BLKSIZE=absexp]				X	X																																																																																																																																																																																																																												
[.IMSK=code]				X	X																																																																																																																																																																																																																												
[.EXLST=symbol]				X	X																																																																																																																																																																																																																												
[.NCP=absexp]				X			X																																																																																																																																																																																																																										
[.BUFNO=absexp]				X																																																																																																																																																																																																																													
[.BFALN=code]				X																																																																																																																																																																																																																													
[.BUFL=absexp]				X																																																																																																																																																																																																																													
[.BFTEK=code]				X																																																																																																																																																																																																																													
[.BUFCB=addr]				X																																																																																																																																																																																																																													
[.EROPT=code]					X																																																																																																																																																																																																																												

	DCBD	none																						
[symbol]	DDEF	<p>oplist { text }           { addr }</p> <p>where oplist equals:</p> <p>{ ddname-symbol } [ .dsorg { VI } ]   { PCSOUT }      { VS }</p> <p>.DSNAME=data set name [.DISP= {OLD NEW} ]</p> <p>See DDEF command for additional operands return data: reg 15 = code</p> <table> <thead> <tr> <th>(hex)</th> <th>significance</th> </tr> </thead> <tbody> <tr><td>00</td><td>no error</td></tr> <tr><td>04</td><td>data set name undefined</td></tr> <tr><td>08</td><td>data set name not unique</td></tr> <tr><td>0C</td><td>attention interruption</td></tr> <tr><td>10</td><td>DSORG inconsistent</td></tr> <tr><td>14</td><td>nonexistent generation name specified</td></tr> <tr><td>18</td><td>DSNAME not fully qualified</td></tr> <tr><td>20</td><td>space not available</td></tr> <tr><td>40</td><td>ddname not unique</td></tr> <tr><td>80</td><td>other</td></tr> </tbody> </table> <p>Note: When reg 15 contains a nonzero code, reg 1 contains the identification of the diagnostic message that explains the error.</p>	(hex)	significance	00	no error	04	data set name undefined	08	data set name not unique	0C	attention interruption	10	DSORG inconsistent	14	nonexistent generation name specified	18	DSNAME not fully qualified	20	space not available	40	ddname not unique	80	other
(hex)	significance																							
00	no error																							
04	data set name undefined																							
08	data set name not unique																							
0C	attention interruption																							
10	DSORG inconsistent																							
14	nonexistent generation name specified																							
18	DSNAME not fully qualified																							
20	space not available																							
40	ddname not unique																							
80	other																							
[symbol]	DEL	<p>dsname { name }           { addr }</p> <p>return data: reg 15 = code</p> <table> <thead> <tr> <th>(hex)</th> <th>significance</th> </tr> </thead> <tbody> <tr><td>00</td><td>valid return code</td></tr> <tr><td>08</td><td>invalid return from NEXTPAR</td></tr> <tr><td>0C</td><td>invalid dsname (input preceded by left parenthesis) for NEXTPAR</td></tr> <tr><td>10</td><td>no dsname supplied after verb</td></tr> <tr><td>14</td><td>return code from CHECKDS not divisible by 4</td></tr> <tr><td>24</td><td>data set not cataloged</td></tr> <tr><td>28</td><td>data set on public volume</td></tr> <tr><td>2C</td><td>data set name member of partitioned data set</td></tr> <tr><td>34</td><td>sharer does not have unlimited access to data set</td></tr> <tr><td>40</td><td>attention interruption</td></tr> </tbody> </table>	(hex)	significance	00	valid return code	08	invalid return from NEXTPAR	0C	invalid dsname (input preceded by left parenthesis) for NEXTPAR	10	no dsname supplied after verb	14	return code from CHECKDS not divisible by 4	24	data set not cataloged	28	data set on public volume	2C	data set name member of partitioned data set	34	sharer does not have unlimited access to data set	40	attention interruption
(hex)	significance																							
00	valid return code																							
08	invalid return from NEXTPAR																							
0C	invalid dsname (input preceded by left parenthesis) for NEXTPAR																							
10	no dsname supplied after verb																							
14	return code from CHECKDS not divisible by 4																							
24	data set not cataloged																							
28	data set on public volume																							
2C	data set name member of partitioned data set																							
34	sharer does not have unlimited access to data set																							
40	attention interruption																							
[symbol]	DELETE	<p>{ EP=symbol } { EPLOC= { addrx } }           { (1) }</p>																						
[symbol]	DIR	<p>(icb-addr...)</p> <p>return data: reg 1 = address of any invalid ICB reg 15 = code</p> <table> <thead> <tr> <th>code</th> <th>condition</th> </tr> </thead> <tbody> <tr><td>04</td><td>ICB contains invalid DCB (for input/output and asynchronous ICBs only), or invalid time interval or clock number was specified</td></tr> <tr><td>08</td><td>no routine specified</td></tr> <tr><td>0C</td><td>interrupt servicing routine is active (no further interrupts will be presented to interrupt routine until current servicing action completed)</td></tr> <tr><td>10</td><td>invalid parameter (invalid length specified, or nonprivileged user attempted to DIR privileged routine)</td></tr> </tbody> </table>	code	condition	04	ICB contains invalid DCB (for input/output and asynchronous ICBs only), or invalid time interval or clock number was specified	08	no routine specified	0C	interrupt servicing routine is active (no further interrupts will be presented to interrupt routine until current servicing action completed)	10	invalid parameter (invalid length specified, or nonprivileged user attempted to DIR privileged routine)												
code	condition																							
04	ICB contains invalid DCB (for input/output and asynchronous ICBs only), or invalid time interval or clock number was specified																							
08	no routine specified																							
0C	interrupt servicing routine is active (no further interrupts will be presented to interrupt routine until current servicing action completed)																							
10	invalid parameter (invalid length specified, or nonprivileged user attempted to DIR privileged routine)																							

Name	Operation	Operands
[symbol]	DQDECB	decbl {addrx} { (1) }
		return data: reg 0 = number of unchecked DECBS in the queue reg 1 = pointer to list of unchecked DECBS
	DUPCLOSE	decbl -addr, decb2 -addr
	DUPOPEN	decbl -addr, decb2 -addr[, opt-code]
		For valid codes, see OPEN
[symbol]	EBCDIME	oplist { text } { addr } [, time-addr] [, L= { integer } { register } ]
[symbol]	ERASE	dsname { text } { addr }
		return data: fourth byte of general register 15 set to: code      significance 00      no error detected 04      not class D or batch monitor entry 08      invalid return code 0C      invalid delimiters in data set name 10      no data set name supplied 14      invalid return code from CHEKDS module 18      data set name not in catalog or TDT 1C      partitioned data set not fully qualified name 20      member of partitioned data set not found in POD 24      data set not cataloged 28      data set on public volume 2C      data set is member of partitioned data set 30      user does not own data set in ERASE batch monitor entry 34      sharing/access conflicts prevent processing 38      no catalog entry for ERASE batch monitor entry 3C      data set name undefined; return code from DDEF 44      data set not on direct access 48      volume not found 4C      system JFCB 58      data set in use 5C      resources exceeded
[symbol]	EXIT	message { text } { addrx } [..NOMSG] { (1) }
[symbol]	FREEMAIN	{ PAGE[, VAR] } R      { value } { (0) } , A= { addrx } { (1) }
		return data: reg 15 = code (hex)      significance 00      normal return 04      unsuccessful, no storage freed

[symbol]	GATRD	msg-addr,length-addr[,SIC]
		return data: length-addr=set to actual length reg 15 = code (bits 16-23) <u>significance</u> 0   input record contains no continuation code, record complete 1   input record contains continuation code; issue GATRD for next portion of record 2   record truncated; exceeded maximum length specified by user. (bits 24-31) 0   SYSIN is VSAM fixed-length records (nonconversational) 2   SYSIN is VSAM variable-length records (nonconversational) 4   SYSIN is VISAM (nonconversational)



		8 attention interruption occurred during input or output; record unpredictable 10 SYSIN/SYSOUT received from terminal keyboard 20 SYSIN received from card reader at terminal
[symbol]	GATWR	msg-addr,length-addr return data: see GATRD codes 8 and 10
[symbol]	GDV	$\left[ \text{com-} \begin{array}{l} \text{addrx} \\ \text{text} \\ (1) \end{array} \right]$
[symbol]	GETMAIN	$\left\{ \text{PAGE}[\text{VAR}] \right\} \text{,LV=} \left\{ \text{value}[\text{PR=integer}] \right. \\ \left. \text{R} \right. \\ \left. \left[ \text{,PACK=0} \right] \right\} \\ \left[ \text{.EXIT=RETURN} \right]$ <p>return data:          reg 1 = address of allocated virtual storage          reg 15 = code          (hex) significance          00 normal return          04 request cannot be satisfied          08 bytes specified with invalid protection class</p>
[symbol]	GTWAR	msgout-addr,lengthout-addr, msgin-addr,lengthin-addr[,SIC] return data: see GATRD
	GTWRC	msg-addr,length-addr return data: see GATRD codes 8 and 10
[symbol]	GTWSR	msgout-addr,lengthout-addr, msgin-addr,lengthin-addr[,SIC] return data: see GATRD
[symbol]	INTINQ	$\text{icb-addr} \left[ \text{,MODE=} \begin{array}{l} \text{R} \\ \text{W} \\ \text{CLEAR} \\ \text{(C, branch-addr.)} \\ \text{TYP-code} \end{array} \right]$ <p>return data:          reg 15 = 00 normal          04 undefined routine specified (modes C, W, CLEAR)          08 bad parameter list, conditions specified cannot be met (modes W, C)</p>

**IOREQ (input/output request facility)**

[symbol]	IOREQ	decb-symbol, type- { NB } ,dcb-addr, vccw-addr,length-value,sio-value ,IMSK=code return data: reg 15 = code (dec) significance 0 I/O initiated 4 NCP value in data control block exceeded (I/O not initiated), or DECB "active" or in "wait" state 8 I/O not initiated; VCCW list contains error; rule for forming VCCW list violated (refer to the IOREQ: VCCW macro instruction)
----------	-------	---



Name	Operation	Operand
	IOREQ	(Continued) 12 I/O not initiated, area needed for IOREQ too large; reduce or change VCCW list
[symbol]	CHECK	decb- { addrx } (1)
[symbol]	VCCW	command- { 'symbol' code } absolute expression data-relexp count-absexp[,flag- { CD CC NCC SCC IOC } ,[SIL] [,SKP] )

[symbol]	LOAD	{ EP=symbol EPLOC= { addrx } (1) }
[symbol]	MCAST	[CTT=addrx] [,.EOB=addrx] [.CONT=addrx] [.CLP=addrx] .TRP=addrx] [,DIV=addrx] [.SSM=addrx] [,USM=addrx] [,PL=addrx] [.CP=addrx] [,KC=addrx] [,RS=addrx]
[symbol]	MSGWR	msgcode-addr,[varinf-addr] [,rarea-addr,rlength-addr]  return codes rlength = actual byte length of response reg 15 = code    significance 0            no attention interrupt: no error in response length (if applicable) 4            response too long for area specified 8            attention interrupt occurred; response unpredictable.
[symbol]	OBEY	[ com- { addrx } { text } (1) ]
[symbol]	OPEN	{ { decb-addr, [(opt <sub>1</sub> -code opt <sub>2</sub> -code)] } } ... )  where opt <sub>1</sub> code can be: INPUT    input data set; value is assumed if opt is omitted OUTPUT   output data set INOUT    data control block opened as if INPUT opt was specified; both input and output opera- tions allowed OUTIN    data control block opened as if OUTPUT opt was specified; input and output operations allowed RDBACK   positions an input data set to be read backward (magnetic tape only) UPDAT    allows updating of existing data set (direct-access devices only)
[symbol]	PAUSE	[ message- { text } { addrx } (1) ]

[symbol]	PR	<p>oplist- { text }           { addr }</p> <p>where oplist equals: dsname-name, [startno-integer] [endno-integer],</p> <p>[ spacing- { EDIT           { 1           { 2           { 3 } } } , [H], [lines-integer], [P] ] ]</p> <p>[ .ERASE= { Y }           { N } ] , [error- { ACCEPT                               { SKIP                               { END } } ] [ ,form-specsym ]</p> <p>return data: reg 1 = batch sequence number reg 15 = code   significance           0     PR request accepted.           All    register 15 contains two-byte           other message number.           codes</p>
[symbol]	PRMPT	<p>msgid- { addr } , resp opt { N           { text }            { P                               { U } } [ ,user resp-addr ]</p> <p>[ ( { param { addr } } { text } { ... } ) ]</p> <p>return codes: reg 15 = code   significance           00    no errors; normal return           04    I/O error           08    system error           0C    message could not be found           10    message filtered out by user due                   to choice of message display,                   level or length in user profile           14    insufficient output buffer space                   available; message truncated           18    explanation not in message file           1C    matching response not in prede-                   fined responses of message file           20    invalid response option specified:                   not N, P or U.           24    message continued           28    attention interrupt during I/O                   operation           2C    message not issued; search for                   message text terminated after                   fifteenth reference pointer found           30    invalid response code found for                   a predictable response</p>
[symbol]	PU	<p>oplist- { text }           { addr }</p> <p>where oplist equals: dsname-name, [startno-integer], [endno-integer],</p> <p>[select- { 1           { 2           { 3           { EDIT } } } ] , [ .ERASE= { Y }                                       { N } ] [ ,form-specsym ]</p> <p>return data: reg 1 = batch sequence number reg 15 = Code   Significance           0     PU request was accepted           All    register 15 contains two-byte           other message number           codes</p>

QSAM (queued sequential access method)

Name	Operation	Operand
[symbol]	GET	dcb- { addrx } (1) [ .area- { addrx } (0) ]
[symbol]	PUT	dcb- { addrx } (1) [ .area- { addrx } (0) ]
[symbol]	PUTX	dcbout- { addrx } (1) . [ dcbin- { addrx } (0) ]
[symbol]	RELEASE	dcb- { addrx } (1)
[symbol]	TRUNC	dcb- { addrx } (1)
[symbol]	SETL	dcb- { addrx } (1) ,type { C R B E P } [ ,limit- { addrx } (0) ]

[symbol]	RAE	[area-addr]
[symbol]	REL	oplist- { text } addr where oplist equals: ddname-symbol[,dsname-name] return data: reg 15 = code (hex) significance 00 normal 04 defaulted, invalid ddname 08 attention interrupt 0C reserved ddname specified i0 undefined ddname 14 uncataloged on public storage 18 undefined dsname 20 spurious input 24 unable to unload all modules loaded from library
[symbol]	RETURN	[(reg <sub>1</sub> -integer[,reg <sub>2</sub> -integer])][,T] [ ,RC= { absexp } (15) ]
[symbol]	SAEC	[ EP= { symbol } 0 [ ,DCB=addr ] [ ,COMAREA=addr ] [ { INTTYP } = { (I A S R) { ,code } ... } { ATTNTYP } = { NULL SAVE RESTORE } ] [ ,MF=L  (E,icb- { addrx } (1) ) ]

[symbol]	SAI	[area-addr]
[symbol]	SAVE	(reg <sub>1</sub> -integer[,reg <sub>2</sub> -integer]),[T] [,entry point ID- $\left\{ \begin{array}{l} \text{characters} \\ * \end{array} \right\}$ ] where * denotes name field symbol
[symbol]	SEEC	[EP= $\left\{ \begin{array}{l} \text{symbol} \\ 0 \end{array} \right\}$ ] [,COMAREA=addr] [,INTTYP=integer] [,MSGAREA=addr] [,MSGLTH=integer] [,MF=L (E,icb- $\left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\}$ )]
[symbol]	SIEC	[EP= $\left\{ \begin{array}{l} \text{symbol} \\ 0 \end{array} \right\}$ ] [,DCB=addr] [,COMAREA=addr][,MF=L (E,icb- $\left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\}$ )]
[symbol]	SIR	(icb-addr,...)[,PRTY=integer] [,INHIBIT= {YES NO} ]  return data: reg 1 = address of invalid ICB reg 15 = <u>code</u> <u>condition</u> 04 ICB contains invalid DCB (for input/output, asynchronous ICBs only), or invalid time or clock number specified 08 ICB specified previously by another SIR macro instruction 0C invalid parameter (ICB, length invalid, nonprivileged user requested privileged priority or has privileged flag on in parameter area) 10 total user time will exceed 7.5 hours; time not set
[symbol]	SPEC	[EP=symbol] [,COMAREA=addr]  [INTTYP= $\left\{ \begin{array}{l} \left( \left( \left( \begin{array}{l} A \\ S \\ R \end{array} \right) \right) \left\{ \begin{array}{l} \text{code-integer} \\ \text{integer-integer} \\ \text{integer-range} \end{array} \right\} \dots \right) \end{array} \right\}$ ALL NULL SAVE RESTORE ]  [,MF=L (E,icb- $\left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\}$ )]
[symbol]	SSEC	[EP= $\left\{ \begin{array}{l} \text{symbol} \\ 0 \end{array} \right\}$ ] [,COMAREA=addr]  [,INTTYP=integer]  [,MF=L (E,icb- $\left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\}$ )]
[symbol]	STEC	[EP= $\left\{ \begin{array}{l} \text{symbol} \\ 0 \end{array} \right\}$ ] [,COMAREA=addr]  [INTTYP= $\left\{ \begin{array}{l} \text{TASKn,} \left\{ \begin{array}{l} \text{DINTVL=addr} \\ \text{BINTVL=addr} \end{array} \right\} \\ \text{REALn} \left\{ \begin{array}{l} \text{DINTVL=addr} \\ \text{BINTVL=addr} \\ \text{TOD=addr} \\ \text{DO} \left\{ \text{W M Y} \right\} =\text{addr} \end{array} \right\} \end{array} \right\}$ ]  [,MF=L (E,icb- $\left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\}$ )]

Name	Operation	Operand
[symbol]	STIMER	$\left\{ \begin{array}{l} \text{TASK} [\text{.exit-symbol}] \left\{ \begin{array}{l} \text{DINTVL}=\text{addr} \\ \text{BINTVL}=\text{addr} \end{array} \right\} \\ \left\{ \begin{array}{l} \text{REAL} [\text{.exit-symbol}] \\ \text{WAIT} \end{array} \right\} \left\{ \begin{array}{l} \text{DINTVL}=\text{addr} \\ \text{BINTVL}=\text{addr} \\ \text{TOD}=\text{addr} \\ \text{DO} \left\{ \text{W}   \text{M}   \text{Y} \right\} =\text{addr} \end{array} \right\} \end{array} \right\}$ $[ \text{.TNO} = \left\{ 0   \text{integer} \right\} ]$ return data: reg 15 = $\frac{\text{code}}{\text{significance}}$ 00 normal return 04 invalid time interval or invalid clock number specified 08 total user task time specified exceeds 7.5 hours, timer was not set
[symbol]	SYSIN	msg-addr.length-addr[.source-code] [,prmt-addr] [,exit-addr] return data: reg 15 = $\frac{\text{code}}{\text{significance}}$ 00 normal return 04 user's input line truncated 08 attention interruption 0C immediate command detected and executed 10 input line in keyboard format, normal return 14 input line in keyboard format, line truncated 20 card reader format, normal return 24 card reader format, line truncated.
[symbol]	TTIMER	$\left\{ \begin{array}{l} \text{TASK} \\ \text{REAL} \end{array} \right\} [ \text{.CANCEL} ] [ \text{.TNO} = \left\{ 0   \text{integer} \right\} ]$ return data: reg 15 = $\frac{\text{code}}{\text{significance}}$ 00 normal return 04 invalid clock number specified
	USAGE	area-addr[,userid-addr]
[symbol]	USATT	none

#### VSAM (Virtual sequential access method)

[symbol]	GET	$\text{dcb-} \left\{ \begin{array}{l} \text{addr} \\ (1) \end{array} \right\} \left[ \text{.area-} \left\{ \begin{array}{l} \text{addrx} \\ (0) \end{array} \right\} \right]$ return data: reg 1 = in locate mode, address of next sequential record
[symbol]	PUT	$\text{dcb-} \left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\} \left[ \text{.area-} \left\{ \begin{array}{l} \text{addrx} \\ (0) \end{array} \right\} \right]$ return data: reg 1 = in locate mode, address of output buffer for next sequential output record
[symbol]	PUTX	$\text{dcb-} \left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\}$
[symbol]	SETL	$\text{dcb-} \left\{ \begin{array}{l} \text{addrx} \\ (1) \end{array} \right\} \text{.type-} \left\{ \begin{array}{l} \text{R} \\ \text{B} \\ \text{E} \\ \text{P} \end{array} \right\} [ \text{.llimit-} \left\{ \begin{array}{l} \text{addrx} \\ (0) \end{array} \right\} ]$

VISAM (virtual indexed sequential access method)

Name	Operation	Operand
[symbol]	GET	dcb- { addr } (1) { [ .area- { addrx } (0) ] }
[symbol]	PUT	dcb- { addrx } (1) { [ .area- { addrx } (0) ] }
[symbol]	READ	dcb-symbol,type- { KY   KZ   KX } .dcb-addr .area-addr, key-addr
[symbol]	WRITE	dcb-symbol,type- { KR   KS   KT } .dcb-addr .area-addr, key-addr
[symbol]	SETL	dcb- { addrx } (1) ,type- { R B E P   K I N } [ .llimits- { addrx } (0) ]
[symbol]	ESETL	dcb- { addrx } (1)
[symbol]	DELREC	dcb- { addrx } (1) ,type- { K } R { .llimit- { addrx } (0) }
[symbol]	RELEX	dcb- { addrx } (1)

VPAM (virtual partitioned access method)

[symbol]	FIND	dcb-addr,name-addr[,area-addr,length-value]  return data: reg 0 = length of user data in POD reg 1 = points to parameter list reg 15 = code  (hex) significance 00 successful completion of FIND 04 Member or alias not located by FIND 08 data control block, indicated in macro instruction, in use to create member, execution of STOW must be complete before FIND 10 length specified in macro instruc- tion not large enough for user data 14 member to be located already open for this data control block, due to previous FIND
[symbol]	STOW	dcb { addrx } (1) { . [ area- { addrx } (0) ] } .type { N   NA   NAR   R   U   D   DA   C   CA } return data bits 24-31 reg 15: code (hex) significance 00 successful completion of STOW 04 new name or alias already in use (N, NA, C, or CA) 08 member name not in POD (U, D, DA, or CA) 10 old member name not in POD (C), alias not in POD (DA), old alias not in POD (CA) 14 invalid STOW type requested 18 user data exceeds maximum length of 510 bytes

Name	Operation	Operand
[symbol]	WT	oplist- $\left\{ \begin{array}{l} \text{text} \\ \text{addr} \end{array} \right\}$  where oplist equals: dsname <sub>1</sub> -name,dsname <sub>2</sub> -name,[volume-alphnum] , [factor-integer] ,[startno-integer] ,[endno-integer]  $\left[ \begin{array}{l} \text{EDIT} \\ \text{,spacing-} \left\{ \begin{array}{l} 1 \\ 2 \\ 3 \end{array} \right\} \text{,[H] ,[lines-integer] ,[P]} \end{array} \right]$  $\left[ \text{,ERASE} = \left\{ \begin{array}{l} Y \\ N \end{array} \right\} \right]$
		return data: reg 1=address of batch sequence number reg 15 = code      significance 0      WT request accepted. All      register 15 contains two-byte other      message number codes
[symbol]	WTL	message-text  return data: reg 15 = code (hex)      significance 0      successful 4      attention interruption C      invalid message length, no message sent
[symbol]	WTO	message-text  return data: reg 15 = code (hex)      significance 0      successful 4      attention interruption C      invalid message length, no message sent
[symbol]	WTOA	message-text  return data: reg 15=code (hex)      significance 0      successful 4      attention interruption C      invalid message length; no message sent
[symbol]	WTOR	message-text,reply-addr,length-value  return data: reg 15 = code (hex)      significance 0      successful 4      attention interruption C      invalid message length, no message sent 10      reply length greater than maximum, reply was received but only maximum number of characters in reply area

## CONTENTS

Permanent Storage Assignment . . .	92
Hexadecimal and Decimal Conversion . . .	92
Hex-Binary Power Table . . .	93
Channel Address Word (CAW) . . .	93
Channel Status Word (CSW) . . .	93
Channel Command Word (CCW) . . .	94
Direct-Access Storage-Device Command Codes . . .	94
Channel Command Codes . . .	95
Virtual Program Status Word (VPSW) . . .	97
Extended Program Status Word (XPSW) . . .	97
TSS/360 Interruption Codes . . .	99
Extended Program Interruption Codes . . .	100
Printer Carriage-Control Codes . . .	102
Machine Codes . . .	102
Extended ANSI FORTRAN Codes . . .	102
Terminal Session Examples	
Example 1 . . .	104
Example 2 . . .	108

GN28-3172 9/15/70 91

**APPENDIX**



**Permanent Storage Assignment**

Address					
Dec	Hex	Binary	Length	Purpose	
0	0	0000 0000	double word	IPL PSW	} overlaid after IPL
8	8	0000 1000	double word	IPL CCW1	
16	10	0001 0000	double word	IPL CCW2	
24	18	0001 1000	double word	external old PSW	
32	20	0010 0000	double word	supervisor call old PSW	
40	28	0010 1000	double word	program old PSW	
48	30	0011 0000	double word	machine-check old PSW	
56	38	0011 1000	double word	input/output old PSW	
64	40	0100 0000	double word	channel status word	
72	48	0100 1000	word	channel status word	
76	4C	0100 1100	word	unused	
80	50	0101 0000	word	timer (uses bytes 50, 51, 52)	
84	54	0101 0100	word	unused	
88	58	0101 1000	double word	external new PSW	
96	60	0110 0000	double word	supervisor call new PSW	
104	68	0110 1000	double word	program new PSW	
112	70	0111 0000	double word	machine-check new PSW	
120	78	0111 1000	double word	input/output new PSW	
128	80	1000 0000	*	diagnostic scan-out area	

\*Maximum size is 256 bytes.

Old and new PSWs, in permanent storage, are always in extended PSW format; interrupt codes will overlay low-core 14-23 as indicated below:

14	E	0000 1110	2 bytes	External interrupt mask (for extended PSW)
16	10	0001 0000	2 bytes	SVC
18	12	0001 0010	2 bytes	Program interrupt code
20	14	0001 0100	2 bytes	Machine check code
22-23	16	0001 0000	2 bytes	I/O interrupt code

**Hexadecimal and Decimal Conversions**

To find a decimal number: Locate each digit of the hex number and its decimal equivalent for each position (columns 6-to-1). The decimal number is the sum of the equivalents.

To find a hex equivalent of a decimal number: (A) Locate the decimal number that is next-lower than that number; this is the first hex digit; (b) subtract the lower decimal number from the original decimal number; (c) repeat step A for the decimal difference, to get the next hex digit; then continue the process until the entire original decimal number has been converted.

BYTE				BYTE				BYTE																
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
6				5				4				3				2				1				

**Hex-Binary Power Table**

POWERS OF 16					POWERS OF 2			
16 <sup>n</sup>				n	2 <sup>n</sup>		n	
			1	0	512	9		
			16	1	1 024	10		
			256	2	2 048	11		
		4	096	3	4 096	12		
		65	536	4	8 192	13		
		1 048	576	5	16 384	14		
		16 777	216	6	32 768	15		
		268	435 456	7	65 536	16		
	4	294	967 296	8	131 072	17		
	68	719	476 736	9	262 144	18		
	1 099	511 627	776	10	524 288	19		
	17	592 186 044	416	11	1 048 576	20		
	281	474 976 710	656	12	2 097 152	21		
4	503	599 627 370	496	13	4 194 304	22		
72	057	594 037 927	936	14	8 383 608	23		
1 152	921	504 606 846	976	15	16 777 216	24		

**Channel Address Word**

<b>Key</b>	<b>0 0 0 0</b>	<b>Channel command word address</b>
0 3 4	7 8	31

**Bits Meaning**

- 0-3 storage protection key = 0-15
- 4-7 must be all 0s
- 8-31 address of first CCW in main storage associated with start-I/O instruction

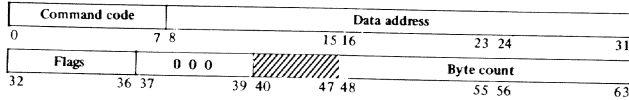
**Channel Status Word**

<b>Key</b>	<b>0 0 0 0</b>	<b>Channel command word address</b>
0 3 4	7 8	31
<b>Status</b>		<b>Byte count</b>
32	47 48	63

**Bit Meaning**

- 0-3 storage protection key used by I/O operation initiated by last start-I/O instruction - = 0-15
- 4-7 must be all 0s
- 8-31 address of last CCW used + 8
- 32-47 conditions that caused CSW to be stored
- 32 attention
- 33 status modifier
- 34 control unit end
- 35 busy
- 36 channel end
- 37 device end
- 38 unit check
- 39 unit exception
- 40 program-controlled interruption
- 41 incorrect length
- 42 program check
- 43 protection check
- 44 channel data check
- 45 channel control check
- 46 interface control check
- 47 chaining check
- 48-63 count; contains residual count (bytes not transmitted) of last CCW used; when reading or writing variable-length records

**Channel Command Word**



- 0-7 command codes by device as indicated below
- 8-31 address of storage area to be operated on by CCW
- 32-36 flags for CCW operations
- 32 causes address portion of next CCW to be used
- 33 causes command code and data address in next CCW to be used
- 34 causes possible incorrect length indication to be suppressed
- 35 suppresses transfer of information to main storage
- 36 causes interruption, as program control interrupt
- 37-39 0s for all CCWs other than TIC
- 40-47 not used
- 48-63 number of bytes in area addressed by bits 8-31

**Direct-Access Storage-Device Command Codes**

Command for CCW		Count	Multiple-track (M-T) Off		M-T on†		
			8-bit code 0123 4567	Hex Dec	Hex Dec	Hex Dec	
Control	No op	X	0000 0001	03 03			
	Seek	6	0000 0111	07 07			
	Seek cylinder	6	0000 1011	0B 11			
	Seek head	6	0001 1011	1B 27			
	Set file mask	1	0001 1111	1F 31			
	Space count	X	0000 1111	0F 15			
	Transfer in channel	X	XXXX 1000	X8			
	Recalibrate (2311 only)		0001 0011	13 19			
	Restore (2321 only)	X	0001 0111	17 23			
	Sense	Sense I/O	6	0000 0110	04 04		
		Release device	X	1001 0100	94 148		
	Switching	Reserve device	X	1011 0100	B4 180		
	Search†	Home address EQ	4 (usually)	0011 1001	39 57	B9	185
		Identifier EQ	5 (usually)	0011 0001	31 49	B1	177
Identifier HI		5 (usually)	0101 0001	51 81	D1	209	
Identifier EQ or HI		5 (usually)	0111 0001	71 113	F1	241	
Key EQ		1 to 255	0010 1001	29 41	A9	169	
Key HI		1 to 255	0100 1001	49 73	C9	201	
Key EQ or HI		1 to 255	0110 1001	69 105	E9	233	
Key & data EQ*		}	0010 1101	2D 45	AD	173	
Key & data HI*			0100 1101	4D 77	CD	205	
Key & data EQ or HI*			0110 1101	6D 109	ED	237	
Continue scan EQ*			Note 1	0010 0101	25 37		
HI*			0100 0101	45 69			
EQ or HI* no compare*			0110 0101	65 101			
EQ or HI* set compare*			0101 0101	55 85			
		0111 0101	75 117				
Read‡	Home address	5	0001 1010	1A 26	9A	154	
	Count	8	0001 0010	12 18	92	146	
	Record R0		0001 0110	16 22	96	150	
	Data	} Number of bytes transferred	0000 0110	06 06	86	134	
	Key & data		0000 1110	0E 14	8E	142	
	Count, key & data		0001 1110	1E 30	9E	158	
Initial program load (IPL)		0000 0010	02 02				
Write	Home address	5 (usually)	0001 1001	19 25			
	Record R0	8+KL+DL of R0	0001 0101	15 21			
	Count key & data	8+KL+DL	0001 1101	1D 29			
	Special count key & data*	8+KL+DL	0000 0001	01 01			
	Data Key & data	DL KL & DL	0000 0101	05 05			
			0000 1101	0D 13			

\* Special Feature Note 1. Includes mask bytes in search argument.  
 † M-T On = M-T Off except, during Search and Read bit 0 = 1 in M-t On.  
 X = not significant; KL = Key Length DL = Data Length; EQ = Equal; HI = High



Device	Command for CCW	8-bit code							Hex	Dec																																																																																																												
		0	1	2	3	4	5	6			7																																																																																																											
1403 or 1443	Write, no space	0	0	0	0	0	0	0	1	01	01																																																																																																											
	Write, space 1 after print	0	0	0	0	1	0	0	1	09	09																																																																																																											
	Write, space 2 after print	0	0	0	1	0	0	0	1	11	17																																																																																																											
	Write, space 3 after print	0	0	0	1	1	0	0	1	19	25																																																																																																											
	Write, skip to channel N after print	1	C	H	A	N	0	0	1																																																																																																													
	Diagnostic read (1403)	0	0	0	0	0	0	1	0	02	02																																																																																																											
	Diagnostic read (1443)	0	0	0	0	0	1	1	0	06	06																																																																																																											
	Test I/O	0	0	0	0	0	0	0	0	00	00																																																																																																											
	Sense	0	0	0	0	0	1	0	0	04	04																																																																																																											
Carriage Control	Space 1 line immediately	0	0	0	0	1	0	1	1	0B	11																																																																																																											
	Space 2 line immediately	0	0	0	1	0	0	1	1	13	19																																																																																																											
	Space 3 line immediately	0	0	0	1	1	0	1	1	1B	27																																																																																																											
	Skip to channel N immediately	1	C	H	A	N	0	1	1																																																																																																													
	No op	0	0	0	0	0	0	1	1	03	03																																																																																																											
		C	H	A	N	Channel	C	H	A	N	Channel																																																																																																											
		0	0	0	1	1	0	1	1	1	7																																																																																																											
		0	0	1	0	2	1	0	0	0	8																																																																																																											
		0	0	1	1	3	1	0	0	1	9																																																																																																											
		0	1	0	0	4	1	0	1	0	10																																																																																																											
	0	1	0	1	5	1	0	1	1	11																																																																																																												
	0	1	1	0	6	1	1	0	0	12																																																																																																												
2400 Tape *	Transfer in channel	0	0	0	0	1	0	0	0	08	08																																																																																																											
	Sense	0	0	0	0	0	1	0	0	04	04																																																																																																											
	Read backward**	0	0	0	0	1	1	0	0	0C	12																																																																																																											
	Write	0	0	0	0	0	0	0	1	01	01																																																																																																											
	Read	0	0	0	0	0	0	1	0	02	02																																																																																																											
	Control	0	0	C	C	C	1	1	1																																																																																																													
	Mode set	D	D	M	M	M	0	1	1																																																																																																													
* 9-track op. forces 800 BPI and odd parity; also, it overrides 7-track but does not reset 7-track. Load/sys reset forces 7-track to 800 BPI, odd parity, data converter on, translator off.																																																																																																																						
<table border="1"> <thead> <tr> <th>C</th> <th>C</th> <th>C</th> <th>Control codes</th> <th>Hex</th> <th>Dec</th> <th>D</th> <th>D</th> <th>7-Track density</th> </tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>REW</td><td>7</td><td>7</td><td>0</td><td>0</td><td>200</td> </tr> <tr> <td>0</td><td>0</td><td>1</td><td>RUN</td><td>0F</td><td>15</td><td>0</td><td>1</td><td>556</td> </tr> <tr> <td>0</td><td>1</td><td>0</td><td>ERG</td><td>17</td><td>23</td><td>1</td><td>0</td><td>800</td> </tr> <tr> <td>0</td><td>1</td><td>1</td><td>WTM</td><td>1F</td><td>31</td><td>1</td><td>1</td><td>800</td> </tr> <tr> <td>1</td><td>0</td><td>0</td><td>BSR</td><td>27</td><td>39</td><td></td><td></td><td></td> </tr> <tr> <td>1</td><td>0</td><td>1</td><td>BSF</td><td>2F</td><td>47</td><td></td><td></td><td></td> </tr> <tr> <td>1</td><td>1</td><td>0</td><td>FSR</td><td>37</td><td>55</td><td></td><td></td><td></td> </tr> <tr> <td>1</td><td>1</td><td>1</td><td>FSF</td><td>3F</td><td>63</td><td></td><td></td><td></td> </tr> </tbody> </table>											C	C	C	Control codes	Hex	Dec	D	D	7-Track density	0	0	0	REW	7	7	0	0	200	0	0	1	RUN	0F	15	0	1	556	0	1	0	ERG	17	23	1	0	800	0	1	1	WTM	1F	31	1	1	800	1	0	0	BSR	27	39				1	0	1	BSF	2F	47				1	1	0	FSR	37	55				1	1	1	FSF	3F	63																														
C	C	C	Control codes	Hex	Dec	D	D	7-Track density																																																																																																														
0	0	0	REW	7	7	0	0	200																																																																																																														
0	0	1	RUN	0F	15	0	1	556																																																																																																														
0	1	0	ERG	17	23	1	0	800																																																																																																														
0	1	1	WTM	1F	31	1	1	800																																																																																																														
1	0	0	BSR	27	39																																																																																																																	
1	0	1	BSF	2F	47																																																																																																																	
1	1	0	FSR	37	55																																																																																																																	
1	1	1	FSF	3F	63																																																																																																																	
**Overrides data Converter On																																																																																																																						
<table border="1"> <thead> <tr> <th>M</th> <th>M</th> <th>M</th> <th>(Mode Modifiers)</th> <th>Set Density</th> <th>Set Odd Parity</th> <th>Set Even Parity</th> <th>Data Converter On</th> <th>Data Converter Off</th> <th>Translator On</th> <th>Translator Off</th> <th>Request TIE (Track in Error)</th> </tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>No Op</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>0</td><td>0</td><td>1</td><td>Not used</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>0</td><td>1</td><td>0</td><td>Reset Condition</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td><td>X</td><td></td> </tr> <tr> <td>0</td><td>1</td><td>1</td><td>Nine-track only</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>X</td> </tr> <tr> <td>1</td><td>0</td><td>0</td><td></td><td>X</td><td></td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td> </tr> <tr> <td>1</td><td>0</td><td>1</td><td></td><td>X</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td> </tr> <tr> <td>1</td><td>1</td><td>0</td><td>Reset Condition</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td> </tr> <tr> <td>1</td><td>1</td><td>1</td><td></td><td>X</td><td>X</td><td></td><td>X</td><td>X</td><td></td><td></td><td></td> </tr> </tbody> </table>											M	M	M	(Mode Modifiers)	Set Density	Set Odd Parity	Set Even Parity	Data Converter On	Data Converter Off	Translator On	Translator Off	Request TIE (Track in Error)	0	0	0	No Op									0	0	1	Not used									0	1	0	Reset Condition	X	X		X			X		0	1	1	Nine-track only								X	1	0	0		X		X	X		X			1	0	1		X		X	X	X				1	1	0	Reset Condition	X	X		X		X			1	1	1		X	X		X	X			
M	M	M	(Mode Modifiers)	Set Density	Set Odd Parity	Set Even Parity	Data Converter On	Data Converter Off	Translator On	Translator Off	Request TIE (Track in Error)																																																																																																											
0	0	0	No Op																																																																																																																			
0	0	1	Not used																																																																																																																			
0	1	0	Reset Condition	X	X		X			X																																																																																																												
0	1	1	Nine-track only								X																																																																																																											
1	0	0		X		X	X		X																																																																																																													
1	0	1		X		X	X	X																																																																																																														
1	1	0	Reset Condition	X	X		X		X																																																																																																													
1	1	1		X	X		X	X																																																																																																														



- 6 I/O mask (summary) { 1=interrupts allowed }  
 { 0=interrupts not allowed }  
 (used with control register 4, extended mask register)
- 7 external mask (summary) { 1=external interrupts allowed }  
 { 0=external interrupts not allowed }  
 (used with control register 6, bit 24-31)
- 12 character mode { ASCII=1 }  
 { EBCDIC=0 }
- 13 machine check mask; operates with register -6 machine-check control

Control register 6			
PSW	Bit 0	Bit 1	Remarks
0	-	-	all machine checks masked-off by PSW bit 13
1	0	0	only CPU machine checks recognized
1	0	1	CPU and channel controller 1 machine checks recognized
1	1	0	CPU and channel controller 0 machine checks recognized
1	1*	1*	All machine checks recognized

\* Bits set to 1 by system reset

- 14 wait state = 1  
running state = 0
- 15 problem state = 1  
supervisor state = 0
- 16-17 instruction length code

ILC bits	Instruction length	Op-code bits
01	1 half word	00
10	2 half words	01 or 10
11	3 half words	11

- 18-19 condition code
- | Bits | Meaning   | Branch or condition masks |
|------|-----------|---------------------------|
| 00   | 0-balance | equal 1000                |
| 01   | negative  | low 0100                  |
| 10   | positive  | high 0010<br>0001         |
| 11   | overflow  | combination 1101          |

- 20-23 program mask
- 20 fixed-point overflow mask
- 21 decimal overflow mask
- 22 exponent underflow mask
- 23 significance mask
- 24-31 spare
- 32-63 instruction address

Note: Real storage dumps use Extended PSW

Time Sharing System/360—Interrupt Codes

Interrupt	Interrupt code (PSW Bits 16-31)	Dec.	Hex.	I.L.C.	Instruction execution is
Machine check (CPU)	00000000 00000000	0	00	u	Terminated
External machine check Mod. 2067-2 (see notes 2, 3)	10uuuuuu uuuuuuuu			u	Completed
Program					
Operation	00000000 00000001	1	0001	1, 2, 3	Suppressed
Privileged operation	00000000 00000010	2	0002	1, 2	Suppressed
Execute	00000000 00000011	3	0003	2	Suppressed
Protection	00000000 00000100	4	0004	0, 2, 3	Suppressed/ terminated
Addressing	00000000 00000101	5	0005	1, 2, 3	Suppressed/ terminated
Specification	00000000 00000110	6	0006	1, 2, 3	Suppressed
Data	00000000 00000111	7	0007	2, 3	Suppressed/ terminated
Fixed-point overflow	00000000 00001000	8	0008	1, 2	Completed
Fixed-point divide	00000000 00001001	9	0009	1, 2	Suppressed/ terminated
Decimal overflow	00000000 00001010	10	000A	3	Completed
Decimal divide	00000000 00001011	11	000B	3	Suppressed
Exponent overflow	00000000 00001100	12	000C	1, 2	Terminated
Exponent underflow	00000000 00001101	13	000D	1, 2	Completed
Significance	00000000 00001110	14	000E	1, 2	Completed
Floating-point divide	00000000 00001111	15	000F	1, 2	Suppressed
Segment relocation	00000000 00010000	16	0010	u	Suppressed
Page relocation	00000000 00010001	17	0011	u	Suppressed
Supervisor Call	00000000 rrrrrrrr				1 Suppressed
External					
External signal 7	00000000 uuuuuuu1			u	Completed
External signal 6	00000000 uuuuuu1u			u	Completed
External signal 5	00000000 uuuuu1uu			u	Completed
External signal 4	00000000 uuuu1uuu			u	Completed
External signal 3	00000000 uuuluuuu			u	Completed
External signal 2	00000000 uu1uuuuu			u	Completed
INTERRUPT pushbutton	00000000 u1uuuuuu			u	Completed
Timer	00000000 1uuuuuuu			u	Completed
I/O Mod. 2067-1					
Multiplexor channel	00000000 aaaaaaaa			u	Completed
Selector channel 1	00000001 aaaaaaaa			u	Completed
Selector channel 2	00000010 aaaaaaaa			u	Completed
Selector channel 3	00000011 aaaaaaaa			u	Completed
Selector channel 4	00000100 aaaaaaaa			u	Completed
Selector channel 5	00000101 aaaaaaaa			u	Completed
Selector channel 6	00000110 aaaaaaaa			u	Completed
I/O Mod. 2067-2 (see note 5)					
CCU 1	00000bbb aaaaaaaa			u	Completed
CCU 2	00001bbb aaaaaaaa			u	Completed
CCU 3	00010bbb aaaaaaaa			u	Completed
CCU 4	00011bbb aaaaaaaa			u	Completed

- Abbreviations:  
 u - unpredictable  
 r - R1 and R2 fields in supervisor call instruction  
 a - I/O device address  
 b - I/O channel address  
 masked; if PSW (13) = 1, machine check interrupts are taken. External machine check interrupts are treated according to associated mask bits in CPU control register 4.
- When interruption occurs in standard PSW mode, interruption code is stored in bits 16-31 of old PSW; when in extended PSW mode, code is stored in locations 00E-017.
- In extended PSW mode, bit 13 is overall machine check interrupt mask; includes external machine check interrupts. If PSW (13) = 0, machine check interrupts are
- Extended PSW bit 7 is overall external interrupt mask. If PSW (7) = 0, external interrupts are masked; if PSW (7) = 1, external interruptions are treated according to associated mask bits in CPU control register 6.
- Extended PSW bit 6 is overall I/O interruption mask; if PSW (6) = 0, all I/O channels are masked; if PSW (6) = 1, I/O interruptions are treated according to associated mask bits in CPU control registers 4 and 5.



## TSS/360 Extended Program Interruption Codes

Codes 1 thru 17 are used by the hardware. We reserve codes 18 thru 31 for future hardware interrupt expansion. This leaves codes 0 and 32 to 65535 for specifying software program interrupt errors. Further, codes 65280 thru 65535 are reserved for those errors which are temporary in nature. The currently defined codes are:

CODE (DEC)	CODE (HEX)	ISSUING MODULE	ERROR DESCRIPTION
00	00	CEAA8	DRAM request to nonexistent drum address
01	01		As in preceding "TSS/360 Interruption Codes"
			thru
31	1F		
32	20		(not assigned)
33	21	CEAA1	nonprivileged module not permitted to use PGOUT
34	22	CEAA0	record length exceeds 32,768 bytes
		CEAA1	record length exceeds 32,768 bytes
35	23	CEAA0	buffer page is not assigned to a VM address
		CEAA1	buffer page is not assigned to a VM address
36	24	CEAA0	task has no devices assigned
		CEAA1	task has no devices assigned
37	25	CEAA0	IOCAL parameter list (IORCB) is too short
38	26	CEAR3	Schedule Table entry value specified is invalid
39	27	CEAAF	task has allowed its program interrupts pending counter to overflow
40	28	CEAAF	task has allowed its service calls pending counter to overflow
41	29	CEAAF	task has allowed its external interrupts pending counter to overflow
42	2A	CEAAF	task has allowed its asynchronous interrupts pending counter to overflow
43	2B	CEAAF	task has allowed its timer interrupts pending counter to overflow
44	2C	CEAAF	task has allowed its I/O interrupts pending counter to overflow
45	2D	CEAAF	unclassified task interrupt
46	2E	CEAA0	IOCAL parameter list (IORCB) is too large
47	2F	CEAA0	IOCAL parameter list (IORCB) crosses a page boundary
		CEAA1	PGOUT parameter list (IORCB) crosses a page boundary
48	30	CEAA0	task does not have requested device assigned
		CEAA1	task does not have requested device assigned
49	31	CEAND	task attempted to delete an IVM page
50	32	CEAA0	SVC page is not assigned to a VM address
		CEAA1	SVC page is not assigned to a VM address
51	33	CEAA0	SVC page is not in core
		CEAA1	SVC page is not in core
52	34	CEHAS	nonprivileged user issued TSSS SVC
53	35	CEAND	task attempted to delete a page in an unassigned segment
54	36	CEAND	task attempted to delete an unassigned page
56	38	CEAND	task attempted to reassign an IVM page
61	3D	CEAQ6	the shared segment table request overflows available space
		CEAQ7	the shared segment table request overflows available space
72	48	CEAH2	illegal code given to SETUP/XTRCT SVC processor
73	49	CEAP7	AWAIT SVC not executed remotely or else not on the last half word of an ECB
74	4A	CEAQ6	invalid shared-page table number given to ADSPG SVC processor
75	4B	CEAQ5	VSEND SVC not executed remotely

76	4C	CEAQ5	VSEND parameter list (MCB) exceeds 239 bytes or crosses a page boundary
77	4D	CEAIR	software detected hardware failure
79	4F	CEAHQ	invalid SVC code issued by task
80	50	CEAHQ	user's task not of sufficient priority to issue SVC
81	51	CEAH7	SETXP SVC not on a word boundary
82	52	CEAH7	count of external addresses is zero
83	53	CEAH7	parameter list crosses a page boundary or page not in caller's page table
84	54	CEAH7	count of external addresses exceeds 1022
85	55	CEAH7	a specified page is unassigned
86	56	CEAH7	external device error
87	57	CEAP4	VPSW does not begin on a double word boundary
88	58	CEAQ8	shared page table number supplied by caller is invalid
89	59	CEAHQ	ADDPG call specified an invalid protection key
90	5A	CEAQ7	segment specified is unassigned
91	5B	CEAP0	from page is not assigned to the task
		CEAS7	real time interrupt not deleted, does not exist
92	5C	CEAP0	to page is not assigned to the task
93	5D	CEAS2	illegal code given to SETSYS/XTRSYS SVC processor
94	5E	CEAS4	illegal code given to SETXTS/XTRXTS SVC processor
95	5F	CEAP0	the from or to address is that of a shared page
96	60	CEAHQ	ADDPG request overflows available VM space
		CEAJT	Enter SVC issued to interrupt table type routine while Type III linkage in effect and P1 flag on
97	61	CEAJT	Enter SVC issued with invalid enter code (higher than 255 <sub>10</sub> or unassigned)
98	62	CEAJT	SVC issued on nonprivileged state and no interrupt routine specified
99	63	CEAJT	No asynchronous error routine defined for device with error
100	64	CEAJT	Asynchronous interrupt received; no DE available for device
101	65	CEAJT	SETTR not accepted; system limit reached in table
102	66	CEAJT	Program interrupt received while in Type III linkage
103	67	CEAJT	SVC interrupt received while in Type III linkage
104	68	CEAQ2	task attempted to set timer beyond 55,364,812 milliseconds
105	69	CEAAC	illegal symbolic device address detected
108	6C	CEAA1	PGOUT record length is zero
109	6D	CEAQ6	shared pages to be added exceed 256 maximum
112	70	CEAAK	a SETAE was issued against a device not assigned to task
113	71	CEAAK	a SETAE was issued specifying a non-existent TSI
123	7B	CEAQ6	ADSPG for public share page table not accepted
124	7C	CEAA0	IOCAL SVC CCW list can not be relocated
125	7D	CEAA0	DRAM CCW list can not be relocated
144	90	CEAAQ	relocation read: no path available
145	91	CEAAQ	relocation read: device error on a permanent volume
146	92	CEAAQ	relocation read: device error on a moveable volume
147	93	CEAAQ	relocation read: surface error
148	94	CEAAQ	relocation read: SIO failure
149	95	CEAAQ	IOCAL read: no path available
150	96	CEAAQ	IOCAL read: device error on a permanent volume
151	97	CEAAQ	IOCAL read: device error on a moveable volume

152	98	CEAAQ	IOCAL read: surface error
153	99	CEAAQ	IOCAL read: SIO failure
154	9A	CEAAQ	PGOUT read: no path available
155	9B	CEAAQ	PGOUT read: device error on a permanent volume
156	9C	CEAAQ	PGOUT read: device error on a moveable volume
157	9D	CEAAQ	PGOUT read: surface error
158	9E	CEAAQ	PGOUT read: SIO failure
159	9F	CEAAQ	TWAIT read: no path available
160	A0	CEAAQ	TWAIT read: device error on a permanent volume
161	A1	CEAAQ	TWAIT read: device error on a moveable volume
162	A2	CEAAQ	TWAIT read: surface error
163	A3	CEAAQ	TWAIT read: SIO failure
200	C8	CEAHQ	task has exceeded its TSEND SVC maximum
202	CA	CEAIS	task issued a major VM SYSER
224	E0	CEATC	Interruption from CIP but device not connected to 2701/2/3
225	E1	CEATC	Interruption received from unknown device type; not a 1050, 2741, or TTY35
226	E2	CEATC	VM address of buffer or TCT page not convertible to a real core address
227	E3	CEATC	Active count in TCT or buffer page not zero
228	E4	CEATC	Release of path could not be issued
229	E5	CEATC	No work specified in flag byte
230	E6	CEATC	TCT or buffer page pointer is to IVM
253	FD	CEABZ	Cannot locate VM page containing ERR simulation data
254	FE	CEABZ	Page containing ERR simulation data not in real core
255	FF	CEABZ	ERR simulation table crosses page boundary

## Printer Carriage-Control Codes

### Machine Codes

Function	Byte value (hexadecimal)
Write (no automatic space)	01
Write and space 1 line after printing	09
Write and space 2 lines after printing	11
Write and space 3 lines after printing	19
Write and skip to channel 1 after printing	89
Write and skip to channel 2 after printing	91
Write and skip to channel 3 after printing	99
Write and skip to channel 4 after printing	A1
Write and skip to channel 5 after printing	A9
Write and skip to channel 6 after printing	B1
Write and skip to channel 7 after printing	B9
Write and skip to channel 8 after printing	C1
Write and skip to channel 9 after printing	C9
Write and skip to channel 10 after printing	D1
Write and skip to channel 11 after printing	D9
Write and skip to channel 12 after printing	E1

To obtain corresponding carriage-control operations (space or skip to channel N) without printing, increase value of low-order digit by hexadecimal 2; example:

space two lines      13  
skip to channel 5    AB  
skip to channel 9    CB

### Extended ANSI FORTRAN codes

Function	Character
Skip no line before printing	+
Skip 1 line before printing	blank
Skip 2 lines before printing	0
Skip 3 lines before printing	-
Skip to channel 1 before printing	1
Skip to channel 2 before printing	2
Skip to channel 3 before printing	3
Skip to channel 4 before printing	4
Skip to channel 5 before printing	5
Skip to channel 6 before printing	6
Skip to channel 7 before printing	7
Skip to channel 8 before printing	8
Skip to channel 9 before printing	9
Skip to channel 10 before printing	A
Skip to channel 11 before printing	B
Skip to channel 12 before printing	C



## EXAMPLES

GN28-3172 9/15/70 103

---

## EXAMPLE 1 – EXPLANATION

1. When your terminal is activated, you enter the LOGON command specifying your user identification code and, if you have one, a password.
2. Once the system verifies your user-id and password, the system responds with a message indicating your LOGON was successful and presents the task identification number assigned to your terminal session.
3. The PC? (present catalog) command causes display of your data set catalog.
4. The system presents your data set names, access level (read-only, read-write, or unlimited), and the data set's owner if you are sharing it.
5. Define a virtual partitioned data set named LIB1 as a job library.
6. Invoke the text editor to create a source data set for a FORTRAN program.
7. Enter the break character (underscore) and the INSERT command to insert an omitted line after line 500.
8. Use the break character and INSERT command again to add lines to the end of the data set.
9. Use the break character and the LIST command to display the lines of the data set.
10. The END command terminates use of the text editor; a semicolon separates the END command from the FTN command, which invokes the FORTRAN compiler.
11. Issue the CALL command to invoke the newly compiled program.
12. The program is executed; however, it contains a bug.
13. The QUALIFY and DISPLAY commands are combined in a command statement to display the variables PRIN and YRS.
14. IYRS is displayed.
15. You unload the program, edit the source data set, and recompile it.

## EXAMPLE 1

```

1. login smith,john
2. TSS:360 LEVEL:8-C
   TASKID=0640 LOGON AT 14:02 ON 09/11/70
3. pc?
4. DATA SETS IN CATALOG WITH QUALIFIER SMITH***
   SMITH***.USERLIB, ACCESS: RW
   SMITH***.SOURCE.SORT, ACCESS: RW
   SMITH***.LIST.SORT.G0000V00, ACCESS: RW
   SMITH***.LIST.SORT.G0001V00, ACCESS: RW
   SMITH***.ASMOX.G0000V00, ACCESS: RW, OWNER: TSS*****
   SMITH***.ASMOX.G0000V00, ACCESS: RW, OWNER: TSS*****
5. cdef lib,vb,lib,option=joblib
6. edit source,invest
   000100 c investment return calculation
   000200 1 rate=.06
   000300 2 prin=500.00
   000400 3 annual=100.00
   000500 4 yrs=10
   000600 prin=prin*(1+rate)*annual
   000700 5 print 16,1,prin
7. 000800 insert 500.10
   000900 do 100 i=1,yrs
8. 001000 insert 700
   001100 100 continue
   001200 16 format (1x,16,f12.2)
   001300 stop
   001400 end
9. 001500 list
   001600 c INVESTMENT RETURN CALCULATION
   001700 1 RATE= .06
   001800 2 PRIN=500.00
   001900 3 ANNUAL=100.00
   002000 4 YRS=10
   002100 DO 100 I=1,YRS
   002200 PRIN=PRIN*(1+RATE)*ANNUAL
   002300 5 PRINT 16,1,PRIN
   002400 100 CONTINUE
   002500 16 FORMAT (1X,16,F12.2)
   002600 STOP
   002700 END
10. end; ftn name=invest,stored;y
11. call invest
12. CHCWA20 TERMINATED: STOP
13. qualify invest; display prin,yrs
   PRIN= 62999951E+04
   YRS= 10000000E+02
14. display yrs
   YRS=0
15. unload invest; edit source,invest
   001200 context 30,1,yrs,yrs; list 500
   000500 4 YRS=10
end; ftn name=invest,stored;y
D164 DEFAULT TO REPLACE PREVIOUS MODULE

```



- 
16. Your debugged program runs correctly.
  17. Issue the DSS? command to learn the status of data set SOURCE.INVEST.
  18. Issue the POD? command to display the organization of a partitioned data set—in this case, LIB1, the job library where the program INVEST is stored.
  19. Issue the PERMIT command to let others share your source data set.
  20. Issue the LOGOFF command to allow TSS/360 to disconnect you in an orderly fashion.
  21. When you receive the LOGOFF message (B007), either reenter the LOGON command to start another task, or turn off your terminal.

16. call invest  
1 5300.00  
2 7677.99  
3 9138.66  
4 10686.38  
5 12328.19  
6 14067.97  
7 15911.93  
8 17866.64  
9 19938.63  
10 22134.93  
CHCR200 TERMINATED: STOP  
17. dss? source.invest  
SMTH\*\*\* SOURCE.INVEST  
VOLUME: F38229 (2314)  
ORGANIZATION: VI PAGES: 0000001  
REFERENCE DATE: 254/70 CHANGE DATE: 254/70  
RECORD FORMAT: 2 RECORD LENGTH: 0000132  
KEY LENGTH: 0020007 KEY POSITION: 0000004  
18. pdc? list  
INVEST  
19. pcrnt: source.invest,n.ro,\*all  
20. logoff?  
21. B007 LOGOFF AT 15:18 ON 09/11/70  
TERMINAL LOGICALLY DISCONNECTED, RECONNECT OR HANG UP.

GN28-3172 9/15/70 107

## EXAMPLE 2 - EXPLANATION

1. When your terminal is activated, you enter the LOGON command specifying your user identification code and, if you have one, a password.
2. Once the system verifies your user-id and password, it responds with a message indicating your LOGON was successful and presents the task identification number assigned to your terminal session.
3. You enter the EDIT command to create a FORTRAN program to solve a problem: calculate the number of combinations of n items, if taken r at a time:

$${}^n C_r = \frac{n!}{r!(n-r)!}$$

4. The system prompts with line numbers and you enter your source statements; then,
5. you want to compile your program. The FTN command compiles the program.
6. Since no diagnostics were issued by the FTN compiler, you decide to execute the program, which asks you to enter the n and r variables, in the form aab, where aa=n and b=r.
7. The results are printed out.
8. You realize 7 items taken 4 at a time cannot produce the answer printed at 7. So you use PCS to display intermediate results during program execution to determine where the error is.
9. Issue an AT command to implant a DISPLAY command at a critical point in the program, to be executed each time statement 100 (in your program) is reached.
10. Execute your program again; enter input upon request, and get displays of important variables each time your DO loop is executed.
11. Incorrect results again. Review the variable data returned by the DISPLAY command. You note that maa is incorrect in the second pass through the DO loop; it should have been +210.

## EXAMPLE 2

```

1. login sm:tn, john
2. %S/260 LEVEL 8.0
   TASKID=024E LOGON AT 14:19 ON 09/11/70
3. edit source.binn
4. 000100      write (1,10)
   000200 10   format (1x, enter input in the form aab')
   000300      read (2,11)aa,i
   000400 11   format (12,11)
   000500 20   k=1
   000600 30   do 100 j=1,i
   000700 40   k=k*j
   000800 50   maa=maa*(maa-1)
   000900 100  continue
   001000 110  summaa/k
   001100      write (1,120)num
   001200 120  format (110)
   001300      stop
   001400      end
   001500 end
5. ftn binn, stored=y
6. call binem
   ENTER INPUT IN THE FORM AAB
   074
7. -21306501      TERMINATED: STOP
   CHR4400
8. qualify binn
9. at 100, display maa,i,k,j
   0000:
10. call binem
   ENTER INPUT IN THE FORM AAB
   074
   00001
   MAA=42
   I=4
   K=1
   J=1
   00001
   MAA=1722
   I=4
   K=2
   J=2
   00001
   MAA=2963562
   I=4
   K=6
   J=3
   00001
   MAA=51135038
   I=4
   K=24
   J=4
11. -21306501      TERMINATED: STOP
   CHR4400

```

- 
12. Remove the PCS DISPLAY command from the program. Using the UPDATE command, enter several new source statements that you feel will correct the bug in your program. Issue FTN to recompile the program.
  13. The compilation is successful. Execute your program, again. Enter test data to verify the program is working correctly.
  14. The program returns an answer of 35; you know that is correct.
  15. Request a listing of your corrected source statements before issuing LOGOFF.

```

12. unload binom
CZAMT010 DECEEDING(PCS): ALL AT STATEMENTS REMOVED BY UNLOAD COMMAND IN BINOM
edi: source.binom
000150C --update
550      msave=maa
800 30      if(j-1) 50,100,60
830 50      maa=maa*(msave-j+1)
end
ftn binom.stored=y
0164      DEFAULT TO REPLACE PREVIOUS MODULE
13. call binom
ENTER INPUT IN THE FORM AAB
074
14.      35      TERMINATED: STOP
15. lire? source.binom
000C100 WRITE (1,10)
000C200 10  FORMAT (1X,'ENTER INPUT IN THE FORM AAB')
000C300 READ (2,11)MAA,1
000C400 11  FORMAT (12,11)
000E500 20  <<1
000E530      MSAVE=MAA
000E600 30  DO 100 J=1,1
000E700 40  K=K*J
000E800 50      if(j-1) 60,100,60
000E850 60      MAA=MAA*(MSAVE-J+1)
000E900 100 CONTINUE
0001000 110 NUM=MAA*K
0001100 WRITE (1,120)NUM
0001200 120  FORMAT (110)
0001300 STOP
0001400 END
logoff
E007      LOGOFF AT 14:43 ON 09/11/70
TERMINAL LOGICALLY DISCONNECTED, RECONNECT OR HANG UP.

```

GN28-3172 9/15/70 111



# IBM Technical Newsletter

File Number S360-20  
Re: Order No. GX28-6400-2  
Newsletter No. GN28-3172  
Date: September 15, 1970

IBM System/360  
Time Sharing System  
Quick Guide for Users

©IBM Corp. 1969, 1970

This technical newsletter is a part of Version 8, Modification 0, of the IBM System/360 Time Sharing System. Replacement pages to be inserted in the publication are noted below.

15-18  
21-32  
71-72  
75-78  
81-82  
85-86  
89-94  
99-112

## Summary of Amendments

The changes in this TNL affect the COMMANDS, ASSEMBLER, APPENDIX, and EXAMPLES sections of the Quick Guide for Users. The new commands ABENDREG, EXHIBIT, NEWMSG, and PATFIX have been added. The Assembler User Macro Instructions LPCEDIT, LPCINIT, AUXPG, and PIREC have been added. Corrections and additions have been made to the Appendix and Examples.

Please file this cover letter in the back of the Quick Guide to provide a record of changes.

IBM Corporation, Dept. 643, Neighborhood Road, Kingston, N. Y. 12401

PRINTED IN U.S.A.



How to insert these pages into your  
"Quick Guide"

