GX28-6400-2

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)

X28-6400-2
S 360-20

IBM System/360
Time Sharing System

# TSS/360

## *Quick Guide for Users*

TERMINALS

COMMANDS

PL/I

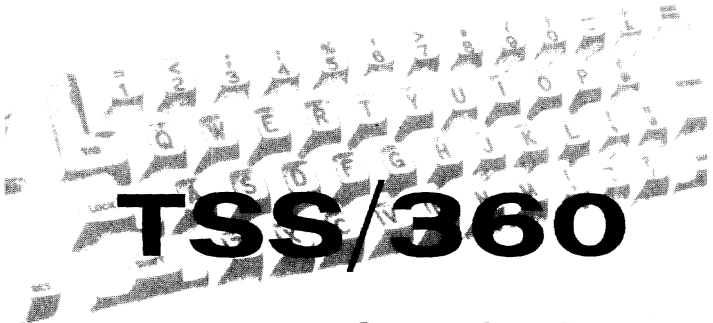
FORTRAN

ASSEMBLER

IBM

Systems Development Division

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:

| *Switch* | *Setting* | *Position* |
|---|---|---|
| SYSTEM | ATTEND | up |
| PRINTER1 | 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 ¬ | ¢ 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[1] | 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[2] | ..... | .... | 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 | \[1] | . | 12-8-1 | ¬[5] | 11-8-7 |
| ! | 5A | ! | ! | 11-8-2 | ! | 11-8-2 |
| \| or ±[10] | 4F | ↑ | , | 0-8-1 | \| | 12-8-7[6] |
| ) | 6E | ) | ) | 0-8-7 | ) | 0-8-6 |
| ¢ | 4A | [2] | ¢ | 11-8-7 | ¢ | 12-8-2 |
| + | 4E | + | + | 12-8-6 | + | 12-8-6 |
| _ | 6D | ]3 | _[4] | 0-8-6 | _ | 0-8-5[7] |
| ? | 6F | ? | ? | 12-8-2 | ? | 0-8-7 |
| ) | 5D | ) | ) | 11-8-5 | ) | 11-8-5 |
| = | 7E | = | = | 8-6 | = | 8-6 |
| ° or ⟨[11] | 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[8] |
| * | 5C | * | * | 11-8-4 | * | 11-8-4 |
| ( | 4D | ( | ( | 12-8-5 | ( | 12-8-5 |
| A | C1 | A | A | 12-0-1 | A[9] | 12-1 |
| B | C2 | B | B | 12-0-2 | B | 12-2 |
| C | C3 | C | C | 12-0-3 | C | 12-3 |
| D | C4 | D | D | 12-0-4 | D | 12-4 |
| E | C5 | E | E | 12-0-5 | E | 12-5 |
| F | C6 | F | F | 12-0-6 | F | 12-6 |
| G | C7 | G | G | 12-0-7 | G | 12-7 |
| H | C8 | H | H | 12-0-8 | H | 12-8 |
| I | C9 | I | I | 12-0-9 | I | 12-9 |
| J | D1 | J | J | 12-11-1 | J | 11-1 |
| K | D2 | K | K | 12-11-2 | K | 11-2 |
| L | D3 | L | L | 12-11-3 | L | 11-3 |
| M | D4 | M | M | 12-11-4 | M | 11-4 |
| N | D5 | N | N | 12-11-5 | N | 11-5 |
| O | D6 | O | O | 12-11-6 | O | 11-6 |
| P | D7 | P | P | 12-11-7 | P | 11-7 |
| Q | D8 | Q | Q | 12-11-8 | Q | 11-8 |
| R | D9 | R | R | 12-11-9 | R | 11-9 |
| S | E2 | S | S | 11-0-2 | S | 0-2 |
| T | E3 | T | T | 11-0-3 | T | 0-3 |
| U | E4 | U | 4 | 11-0-4 | U | 0-4 |
| V | E5 | V | V | 11-0-5 | V | 0-5 |
| W | E6 | W | W | 11-0-6 | W | 0-6 |
| X | E7 | X | X | 11-0-7 | X | 0-7 |
| Y | E8 | Y | Y | 11-0-8 | Y | 0-8 |
| Z | E9 | Z | Z | 11-0-9 | Z | 0-9 |
| blank | 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 ¬ at terminal during input
8. Nothing printed at terminal during input; use multiple-punch 8-1 to print ⟩ 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 ( ± )
11. Degree sign on 2741 correspondence terminal is folded into less-than sign

## Functional Character Sets

| Function characters | EBCDIC equivalent | Punch codes 029/1057 |
|---|---|---|
| **IBM 2741/1052** | | |
| TAB | 05 | |
| SHIFT (up)[1] | 36 | |
| SHIFT (down)[1] | 06 | |
| BACKSPACE[2] | 16 | |
| RETURN (new line) | 15 | |
| LINE FEED | 25 | |
| BYPASS (stop printer)*[3] | 24 | |
| RESTORE (start printer)*[3] | 14 | |
| EOA (end of address)*[3,4] | 7B | |
| EOB (end of block)*[3,5] | 26 | |
| EOT (end of transmission)[3,8] | 37 | |
| PREFIX*[3,6] | 27 | |
| CANCEL*[3,7] | none | |
| RDR STOP*[3] | 35 | |
| **Teletypewriter** | | |
| 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 | |
| ←(backspace) | 16 | |
| **IBM 1056 Card Reader** | | |
| 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 ALTN 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 |

**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 | 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<br>COMBINE<br>INCLUDE<br>RENAME<br>END | summary of these statements and their formats follow command formats |

* 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.

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

## Constants

| Types | Examples |
|---|---|
| Integers — signed decimal integers | -647<br>+1066 |
| **Character** — letters, decimal digits, and special characters, enclosed in apostrophes | '$3.98'<br>'HOW' |
| **Hexadecimal** — one or more hexadecimal digits enclosed in apostrophes, preceded by X | X'1234'<br>X'9FEC3' |
| **Floating point** — signed or unsigned decimal number with or without decimal point or exponents specified as | 3.141<br>3141.59E-1 |
| E — single-precision | 31.20D+3 |
| D — double-precision | 31E-5 |
| **Address** — character A followed by symbol enclosed in apostrophes; may be internal, external, or subscripted | A'symbol' |

| Definition of Operand Terms | Examples |
|---|---|
| **Data locations** — specified as symbols, hexdecimal locations, registers, or dynamic statement counter, in this format: – SYMBOL.(OFFSET,LENGTH), where offset is integer, length is expression | Y.(X'EDC',4) |
| **Data field** — contiguous group of storage to be dumped or displayed; indicated as first location:last location | FLDA:FLDB<br>0:4R<br>0:5E<br>6:2D |

| Operation | Operands | Comment |
|---|---|---|
| ABEND | none | |
| ABENDREG | none | |
| ASM | NAME=object module name | |
| | [,STORED={Y / N}] | Y-module prestored; N-not prestored |
| | [,MACROLIB=(symbolic ddname, index portion ddname)] | default: only system library used |
| | [,VERID=version identification] | default: listing and module are time stamped |
| | [,ISD={Y / N}] | Y-produce internal symbol dictionary N-do not |
| | [,SYMLIST={Y / N}] | Y-produce symbolic listing; N-do not |
| | [,ASMLIST={Y / N}] | Y-produce object program listing; N-do not |
| | [,CRLIST={Y / N}] | Y-produce cross-reference listing; N-do not |
| | [,STEDIT={Y / N}] | Y-produce edited symbol table; N-do not |
| | [,ISDLIST={Y / N}] | Y-produce ISD listing; N-do not |
| | [,PMDLIST={Y / N}] | Y-produce program module dictionary; N-do not |
| | [,LISTDS={Y / N}] | Y-store all requested listings as list data set; N-print all requested listings on SYSOUT<br>defaults: Y if conv'l; N if nonconv'l |
| | [,LINCR=(first line number, increment)] | ignored if STORED=Y<br>default: (100, 100) |
| AT | instruction location [,...] | |
| BACK | DSNAME=data set name | |
| BEGIN | application name<br>[,any application defined parameters] | |
| BRANCH | INSTLOC=instruction location | |
| BUILTIN | NAME=command name<br>[,EXTNAME=bpkd macro name] [,DSNAME=dsname] | |
| CALL | [DSNAME=entry point<br>[,module parameters]] | default: last module referenced by system<br>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)*

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

| | | |
|---|---|---|
| CPS* | VOLUME=volserno<br>[,START={CONT / DSCB address}] | default: beginning of specified volume |
| CVV* | VOLUME=volserno<br>[,START={CONT / DSCB address}] | for VAM data sets<br>default: beginning of specified volume |
| DATA | DSNAME=data set name<br>[(member name)]<br>[,RTYPE={I / LINE / FTN / CARD / S} {[BASE=first line number,] [INCR=increment]}] | I=indexed, default: VSAM<br>default: for BASE, 100; for INCR, 100 |
| DDEF | DDNAME={data definition name / PCS OUT}<br>[,DSORG={VI / VS / VP}]<br>,DSNAME=data set name<br>[(member name)] | default: VI |
| DDNAME? | [JOBLIB={Y / N}] | default: entire JFCB chain displayed |
| DEFAULT | {operand=[value]} [,...] | |
| DELETE | [DSNAME=data set name] | default: individual data sets presented for disposition |
| DISABLE | none | |
| DISPLAY | data field name [,...] | |
| DMPRST | FROMDEV={2311 / 2314 / 2400} | if defaulted, command canceled |
| | ,FRVOLID=volume identification | if defaulted, command canceled |
| | ,TODEV={2311 / 2314 / 2400} | if defaulted, command canceled |
| | [,TOVOLID={volume identification / PRIVATE}] | |
| | [,NEWVLID=volume identification] | 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)*

| | | |
|---|---|---|
| | ,RUNMODE= {BACK / FORE} | ignored if task is nonconversational |
| DSS? | [ NAMES= { data set name / (data set name [,...]) } ] | default: all of user's data sets |
| DUMP | data field name [,...] | |
| EDIT | DSNAME=data set name [(member name)] | default: USERLIB<br>default: no member |
| ENABLE | none | |
| END | none | |
| ERASE | DSNAME=data set name [(member name)] | default: individual data sets presented for disposition |
| EVV | [ DEVICE=datype {2311 / 2314} ] | |
| | ,VOLUME=(volume identification [,...]) | 1-6 decimal digits volume serial number |
| | [,USERID=user identification] | for o-authority programmer only; 8 alphameric characters, first alphabetic; pad with * as required on right side |
| | | default: current user ID |
| EXCERPT | DSNAME=data set name<br>[(member name)]<br>[,RNAME=region name]<br>[,N1=starting line<br>[,N2=ending line] ] | |
| EXCISE | [N1=starting line]<br>[,N2=ending line] | default: CLP<br>default: 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 / MSGE / MSGS} [,message identification] } | default: preceding message or explainable words explained |
| FTN | NAME=module name | |
| | [,STORED= {Y / N} ] | Y-prestored;<br>N-not prestored |
| | [,VERID=version identification] | default: module and listing time-stamped |
| | [,ISD= {Y / N} ] | Y-produce internal symbol dictionary; N-do not |
| | [,SLIST= {Y / N} ] | Y-produce source program listing; N-do not |
| | [,OBLIST= {Y / N} ] | Y-produce object program listing; N-do not |

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

*(continued)*

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

| | | |
|---|---|---|
| NUMBER | [N1=starting line]<br>[,N2=ending line]<br>[,BASE=base number]<br>[,INCR=increment] | |
| PATCLEAR* | DEVICE={2311 / 2314} | |
| | ,VOLID={volume serial / PRIVATE} | volume must not be in use when command invoked |
| | RUNMODE={FORE / BACK} | |
| PATFIX | VOLDEF{(2311,volid,...) / (2314,volid,...) / PUBLIC} | |
| | [,DEVCOUNT=integer] | default: no. of devices specified in user table |
| | [,FIX={Y / N}] | default: N |
| | [,REPORTDS=data set name] | default: diagnostics written on system printer |
| PC? | [NAMES={data set name / (data set name[,...])}] | default: all data sets in catalog |
| PERMIT | DSNAME={data set name / *ALL} | |
| | [,USERID={(user identification [,...]) / *ALL}] | default: *ALL |
| | [,ACCESS= R \| RW \| U \| R] | R-restricts; RO-read only; RW-read/write; U-unlimited |
| PLI | [NAME=module name]<br>[,PLIOPT=compiler option list]<br>[,PLCOPT=language controller options]<br>[,SOURCEDS=source data set name]<br>[,MERGELST=converter input list] | default: source data set name<br>default: null string assumed |
| | [,MERGEDS=converter input data set] | default: no data set assumed |
| | [,MACRODS=intermediate data set name] | default: data set created or value ignored |
| | [,PRVDS=PRV table's data set name] | |
| POD? | PODNAME=data set name<br>[,DATA=Y]<br>[,ALIAS=Y] | default: not printed<br>default: not listed |
| | [,MODULE={ALL / module name}] | default: no module information printed |
| POST | none | |

*Use restricted to privileged programmers

*(continued)*

| | | |
|---|---|---|
| PRINT | DSNAME=data set name | |
| | [,STARTNO-starting byte position] | default: first byte |
| | [,ENDNO=ending byte position] | default: last byte |
| | [, PRTSP={EDIT / 1 / 2 / 3}<br>[,HEADER=H]<br>[,LINES=lines/page]<br>[,PAGE=P]] | When EDIT specified, HEADER, LINES, and PAGE must not be specified<br>default: no header<br>default: 54<br>default: no pages numbered |
| | [,ERASE={Y / N}] | default: N;Y=erase; N=save |
| | [,ERROROPT={ACCEPT / SKIP / END}] | default: END |
| | [,FORM=paper form] | default: installation defined |
| | [,STATION=station id] | default: terminal id in task common |
| | [,TAPOPT={AC - ASCII char / AD - ASCII dump / AE - ASCII edit / ED - EBCDIC dump / EC - normal proc}] | default: EC |
| PRMPT | MSGID=message identification | |
| | [,INSERTn=inserted characters [,...]] | default: no characters inserted |
| PROCDEF | NAME=procedure name | |
| PROFILE | [CSW={Y / N}] | Y-save command symbols; N-do not |
| 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 / N}] | default: N; Y=erase, N=save |
| | [,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 / T } { L / C } { U / R } | P=permanent storage; T=temporary storage — default: P<br>L=erase at LOGOFF; C=erase at CLOSE — default: L, if T specified; no value if P<br>U=read-write access; R=read-only access — default: U |
| 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 / multivolume public data set–MVDS }] | tape to public storage<br>public storage to tape |

| | | |
|---|---|---|
| | [,ACV=volserno] | valid for OPT=MVDS only; default: currently mounted ACV volume |
| | [,START={CONT / DSCB address / fileseqno}] | default: beginning of specified volume |
| RUN | [LOC=entry point name] | default: 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<br>,USERID=owner's user identification | |
| | [,OWNERDS={owner's / dsname / *ALL}] | default: *ALL |
| STET | none | |
| STOP | none | |
| SYNONYM | {term=[value]} [,...] | |
| TIME | [MINS=minutes] | default: SYSGEN value |
| TV | DSNAME1=tape data set name<br>[,DSNAME2=vam data set name] | |
| UNLOAD | [NAME=entry point name] | default: last module referenced by system |
| UPDATE | none | |
| UPDTUSER* | none | |
| USAGE | [USERID=user identification] | 3-8 characters<br>default: current user ID |
| VT | DSNAME1=vam data set name<br>[,DSNAME2=tape data set name] | |
| VV | DSNAME1=current data set name<br>[,DSNAME2=new data set name] | |
| WT | DSNAME=current data set name<br>,DSNAME2=tape data set name | |
| | [,VOLUME=tape volume number] | default: scratch tape |
| | [,FACTOR=blocking factor] | default: 30 |
| | [,STARTNO=starting byte position] | default: first byte |
| | [,ENDNO=ending byte position] | default: last byte |

*Use restricted to privileged programmers

*(continued)*

| | | |
|---|---|---|
| [,PRTSP={EDIT \| {1 \| 2 \| 3}}] | | 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 numbering |
| [,ERASE={Y \| N}] | | 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.

| Operand | Function | Default Value | Other Values |
|---|---|---|---|
| LIMEN | message severity | W - warning messages | I - information<br>X - serious error<br>T - terminate error |
| BREVITY | message length | T - standard message, no ID | M - message ID<br>E - extended<br>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<br>3 - PTTC/6<br>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[1]=data definition name | X | X | X | X | X | X |
| [,DSORG[2]=data set organization] | VS<br>VI<br>VP | VP | VS<br>VI<br>VP | VP | PS | PS |
| ,DSNAME=symbol | X | X | X | X | X | X |
| [,DCB=(,BUFOFF=absexp,DEN=integer,...)] | * | | * | | * | * |
| [,UNIT=((DA {,2311 / ,2314}) / TA[,tape type] / sda-X'cuu')] | | | X | X | X | X |
| | | | X | X | X | X |
| | | | | | X | X |
| | | | | | X | |
| [,SPACE=(([{TRK / CYL / record / length}]), | | | | | X | |
| | | | | | X | |
| | | | | | X | |
| | | | | | | |
| ,primary | X | X | X | X | X | |
| [,secondary] | X | X | X | X | X | |
| [,HOLD])] | X | X | X | X | X | |
| [,VOLUME=(([{PUBLIC / PRIVATE}]) / volseqno) | X | X | | | | |
| | | | X | X | X | X |
| | | | X | X | X | X |
| ([{PRIVATE / volserno}][,...]) | X | X | X | X | ** | ** |
| [,LABEL=([fileseqno-integer] | | | | | X | X |
| | | | | | X | X |
| [,labeltype={NL / SL / SUL / AL / AUL}] | | | | | X | X |
| | | | | | X | X |
| | | | | | X | X |
| | | | | | X | X |
| ,RETPD=integer)] | X | X | X | X | X | X |
| [,DISP=NEW] | X | X | X | X | X | X |
| [,OPTION=JOBLIB] | | 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 | X | X | X | X | X | X |
| [,DSORG=data set organization] | | | | | PS | PS |
| ,DSNAME= {symbol / *symbol} | X | X | X | X | X | X |
| | X | X | X | X | X | X |
| [,DCB=(,BUFOFF=absexp,DEN= integer,...)] | * | * | * | * | * | * |
| [ ,UNIT=(( DA {,2311 / ,2314} / TA[,tape type] / sda-X'cuu' )) ] | | | | | X | |
| | | | | | X | |
| | | | | | | X |
| | | | | | | X |
| [ ,SPACE=( [{TRK / CYL / record length}] ,primary [,secondary] [,HOLD] ) ] | | | | | | |
| | X | X | X | | X | |
| [ ,VOLUME=( [{PUBLIC / PRIVATE / volseqno}] ) ] | | | | | | |
| | X | X | X | X | | |
| | | | X | X | | |
| [,{PRIVATE / volserno}[,...]] | X | X | X | X | X | X |
| | X | X | X | X | X | X |
| [ ,LABEL=( {fileseqno-integer} ) ] | | | | | | X |
| [ ,labeltype-{NL / SL / SUL / AL / AUL} ] | | | | | X | X |
| | | | | | X | X |
| | | | | | X | X |
| | | | | | X | X |
| | | | | | X | X |
| [,RETPD=integer | X | X | X | X | X | X |
| [ ,DISP={OLD / MOD} ] | X | X | X | X | 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 [([VARIABLE][,READONLY] [,PUBLIC][,PROTO] [,COMMON][,PRVLGD] [,SYSTEM])] |
| COMBINE | csname,... |
| RENAME | {extref$_1$, (extref$_2$) / epname$_1$ [(epname$_2$)] / csname$_1$ [(csname$_2$)]} [,...] |
| 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



## FORTRAN



## ASSEMBLER

## PREPROCESSOR STATEMENT FUNCTIONS

| | |
|---|---|
| %ACTIVATE | activates identifier previously defined by %DECLARE that had %DEACTIVATE issued against it; makes identifier eligible for replacement by current preprocessor-assigned values. |
| assignment statement | used to evaluate preprocessor expressions and assign results to preprocessor variable (i.e., %variable = expression) |
| %DEACTIVATE | makes identifier ineligible for replacement by preprocessor-assigned values when it appears in subsequent non-preprocessor (source) statements. |
| %DECLARE | establishes identifier as preprocessor variable or procedure name and activates the identifier for replacement. |
| %DO | used with %END to delimit a preprocessor DO-group; compiles each iteration of groups procedure or compiles conditionally using %IF, %THEN, and %ELSE. |
| %end | delimits end of %DO-group or %PROCEDURE procedures |
| %GO TO | causes compiler scan to be transferred unconditionally to a non-sequential preprocessor statement for subsequent compilation. |
| %INCLUDE | retrieves source program text from a library (POD) and incorporates it into the PL/I program being compiled. |
| %NULL | provides transfer points for %GO TO statements; no operation is performed. |
| %PROCEDURE | used in conjunction with %END to delimit preprocessor procedure; usually to create a functional procedure for compilations. |
| RETURN | returns a value and control back to point from which preprocessor procedure was invoked; no % 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)
%[label:] ...    DO       [i=m1[{TO m2 [BY m3]}]]
                          [    {BY m3 [TO m2]}]
                          where i=preprocessor variable
                            m1,m2,m3=preprocessor expressions
%[label:] ...    END      [label];
%[label:] ...    {GO TO}  label;
                 {GOTO }
%[label:] ...    IF       preprocessor-expression
                          %THEN preprocessor-clause1
                          [%ELSE preprocessor-clause2]
%[label:] ...    INCLUDE    {ddname1 (member-name1)}
                            {member-name1          }

                          [,{ddname 2 (member-name 2)}] ...;
                            {member-name 2           }
%null statement
  %[label:] ...;
%label: [label:] ... PROCEDURE [(identifier [,...] ...)]
                     'RETURNS (CHARACTER|FIXED)';
[label:] ...     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 / psuedo-variable} [,...] =element-expression;

option 2 (array assignment)

{array-variable / psuedo-variable} [,...] = {structure-expression [,BY NAME] / 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]{ [,{...}] ...] ...;

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 / 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 (attrubute...)];

EXIT;

## FORMATS

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

where format-list items are:

*data format items*

| | |
|---|---|
| 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)]
[data-specification { EDIT (data-list)(format-list) [(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**

```
[label:]...;
```

**ON**

```
condition [SNAP] {SYSTEM; | on-unit}
```

where conditions are:

| | *original status* | *changing prefix* |
|---|---|---|
| AREA | on | |
| CHECK(name-list) | off | NOCHECK |
| CONDITION(identifier) | on | |
| 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 | NOSIZE |
| SUBSCRIPTRANGE | off | NOSUBSCRIPTRANGE |
| STRINGRANGE | off | NOSTRINGRANGE |
| TRANSMIT(file-name) | on | |
| UNDEFINEDFILE(file-name) (or) | on | |
| 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 | 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)          }] [EVENT                 }]
[{                {KEYTO(character-string-variable)}]  (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 if 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 | Use of Keywords |
|---|---|
| ABS(x) | built-in function |
| %ACTIVATE or %ACT | preprocessor statement |
| ADD(x,y,p[,q]) | built-in function |
| ADDR(x) | built-in function |
| ALIGNED | attribute(data) |
| ALL(x) | built-in function |
| ALLOCATE | statement |
| ALLOCATION(x) | built-in function |
| ANY(x) | built-in function |
| AREA | condition |
| AREA[(size)] | attribute (data, program control) |
| ATAN(x[,y]) | built-in function |
| ATAND(x[,y]) | built-in function |
| ATANH(x) | built-in function |
| AUTOMATIC or AUTO | attribute (storage) |
| BACKWARDS | attribute(file), option of OPEN statement |
| BASED(pointer-variable) | attribute(storage) |
| BEGIN | statement |
| BINARY or BIN | attribute (data, arithmetic) |
| BINARY or BIN (x[,p[,q]]) | built-in function |
| BIT(length) [VARYING] | attribute(data, string) |
| BIT(expression [,size]) | built-in function |
| BOOL(x,y,w) | built-in function |
| BUFFERED or BUF | attribute(file) |
| BUFFERS(n) | option of ENVIRONMENT attribute |
| BUILTIN | attribute(entry) |
| BY | clause of DO statement |
| BY NAME | option of the assignment statement |
| CALL entry-name | statement or option of INITIAL attribute |
| CEIL(x) | built-in function |
| CHAR(expression [,size]) | built-in function |
| CHARACTER or CHAR (length)[VARYING] | attribute(data, string) |
| CHECK(name-list) | condition |
| CLOSE | statement |
| COBOL | option of ENVIRONMENT attribute |
| COLUMN or COL(w) | format item |
| COMPLETION(event-name) | built-in function, pseudo-variable |
| COMPLEX(or CPLX) | attribute (data, arithmetic) |
| COMPLEX or CPLX(a,b) | built-in function, pseudo-variable |
| CONDITION(name) | condition |
| CONJG(x) | built-in function |
| CONSECUTIVE | option of ENVIRONMENT attribute |
| CONTROLLED or CTL (pointer-variable) | attribute(storage) |
| CONVERSION or CONV | condition |
| COPY | option of GET statement |
| COS(x) | built-in function |
| COSD(x) | built-in function |
| COSH(x) | built-in function |
| COUNT(file-name) | 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 [{subscript-list / POSITION / *}] | 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[1] | 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[3](max-message-size) | option of ENVIRONMENT attribute |
| GENERIC entry-name-declaration[,...]...) | attribute(entry) |
| GENKEY[2] | 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[2] | option of ENVIRONMENT attribute |
| INDEXED | option of ENVIRONMENT attribute |
| INITIAL or INIT(item[,item]...) | attribute(data) |
| INITIAL CALL entry name[argument-list] | atribute(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($x_1$,$x_2$,...,$x_n$) | built-in function |
| MIN($x_1$,$x_2$,...,$x_n$) | built-in function |
| MOD($x_1$,$x_2$) | built-in function |
| MULTIPLY($x_1$,$x_2$,p[,q]) | built-in function |

| | |
|---|---|
| NAME(file-name) | condition |
| NCP(n) | option of ENVIRONMENT attribute |
| NOCHECK | condition prefix identifier (disable CHECK) |
| NOCONVERSION or NOCONV | condition prefix identifier (disables CONVERSION) |
| NOFIXEDOVERFLOW or NOFOFL | condition prefix identifier (disables FIXEDOVERFLOW) |
| NOLOCK[2] | 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[2] | 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' / 'numeric-picture-specification' } | attribute(data, string) |
| POINTER or PTR (area-variable) | attribute(data, program control) |
| PENDING[3] | condition |
| POLY(a,x) | built-in function |
| POSITION or POS(i) *see DEFINED* | attribute(data) |
| PRECISION or PREC(x,p[,q]) | built-in function |
| PRINT | attribute(file), option of OPEN statement |
| PRIORITY[1](x) | option of CALL statement |
| PRIORITY[1][(task-name)] | built-in function, pseudo-variable |

| | |
|---|---|
| PROCEDURE or PROC | statement |
| %PROCEDURE or %PROC | preprocessor statement |
| PROD(x) | built-in function |
| PUT | statement |
| R[3](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[3] | 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[1] | attribute (data, program control), option of PROCEDURE statement |
| TASK[1] [(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[3] | attribute(file) |
| TRANSLATE(s,r[,p]) | built-in function |
| TRANSMIT | condition |
| TRKOFL | 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[2] | 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[4](max-block-size [,max-record-size]) | option of ENVIRONMENT attribute |
| VERIFY(expression1,expression2) | built-in function |
| VS[4](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 condtion
4 - treated same as if keyword V is specified

| TYPE OF OPERATION | OPERATION SYMBOL | | OPERATION |
|---|---|---|---|
| | 60-Character Set | 48-Character Set | |
| ARITHMETIC | + | + | addition or prefix + |
| | – | – | subtraction or prefix – |
| | * | * | multiplication |
| | / | / | division |
| | ** | ** | exponentiation |
| COMPARISON | > | 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 |
| | ¬< | NL | not less than |
| LOGICAL (bit string) | ¬ | NOT | not |
| | \| | OR | or |
| | & | AND | and |
| CONCATE-NATION | \|\| | CAT | concatenation |

| OPERATION | PRIORITY* |
|---|---|
| ** | 1st |
| ¬ | |
| + (prefix) | |
| – (prefix) | |
| / | 2nd |
| * | |
| + (infix) | 3rd |
| – (infix) | |
| \|\| | 4th |
| = | 5th |
| >= | |
| <= | |
| > | |
| < | |
| ¬= | |
| ¬> | |
| ¬< | |
| & | 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 –(A2).

## CHARACTER SETS WITH EBCDIC AND CARD–PUNCH CODES

These card codes are those used by the TSS/360 high-speed care 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 |
| | 12-8-7 | 0100 1111 |
| & | 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 | 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 |

| *Composite Symbols* | *Card Punch* | *60-Character Set Equivalent* | *Composite Symbols* | *Card-Punch* |
|---|---|---|---|---|
| .. | 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 Operators | Explanation |
|---|---|
| ** | exponentiation |
| * | multiplication |
| / | division |
| + | addition |
| - | subtraction |

| Relational Operator | Explanation |
|---|---|
| .GT. | greater than ( $>$ ) |
| .GE. | greater than or equal to ( $\geqslant$ ) |
| .LT. | less than ( $<$ ) |
| .LE. | less than or equal to ( $\leqslant$ ) |
| .EQ. | equal to (=) |
| .NE. | not equal to ( $\neq$ ) |

| Logical operation | Explanation |
|---|---|
| .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:

| Operation | Hierarchy |
|---|---|
| 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 nonsubscripted 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 ($x_1$,$x_2$,$x_3$,...,$x_n$)**

i – executable statement number
$x_1$,$x_2$,$x_3$,...,$x_n$ – executable statement numbers
m – nonsubscripted integer variable, length 4, equal to $x_1$,$x_2$,$x_3$,...,$x_n$

**BACKSPACE a**

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

**CALL name ($a_1$,$a_2$,$a_3$,...,$a_n$)**

name – subroutine's subprogram name or name defined in ENTRY statement in subroutine subprogram
$a_1$,$a_2$,$a_3$,...,$a_n$ – actual arguments supplied to subroutine subprogram; each may be of form &n where n is statement number

**COMMON /r/a ($k_1$),b($k_2$),.../r/c($k_3$),d($k_4$),...**

a,b,...,c,d... – variable or array names
$k_1$,$k_2$,...,$k_3$,$k_4$... – 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 alphameric characters; first is alphabetic; names must be embedded in slashes

**CONTINUE** no operands

**DATA $v_1$...,$v_n$/$i_1$ *$d_1$...,$i_n$ *$d_n$/,$v_{n+1}$,...,$v_m$/$i_{n+1}$ *$d_{n+1}$,...,$i_m$*$d_m$/,...**

$v_1$,...,$v_m$ – variables, subscripted variables (subscripts must be integer constants), or array names
$d_1$,...,$d_m$ – values of integer, real, complex, logical or literal hexadecimal data constants
$i_1$,...,$i_m$ – unsigned integer constants indicating number of consecutive variables to be assigned value of $d_1$,...,$d_m$

**DO** x i = $m_1$, $m_2$, $m_3$

x: end of range; i: DO variable; $m_1$: initial value; $m_2$: test value; $m_3$: increment

x – number of executable statement that follows DO statement
i – nonsubscripted integer variable
$m_1$,$m_2$,$m_3$ – either unsigned integer constants greater than 0, or unsigned nonsubscripted integer variables greater than 0; sum $m_2$+$m_3$ +1 must not exceed size of virtual storage; ($m_3$ optional; if omitted, value assumed to be 1; preceding comma must be omitted)

**DOUBLE PRECISION** a,b,c,...

a,b,c,... – variable names that may be dimensioned in statement, or function names

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

$a_1$, $a_2$, $a_3$,...,$a_N$

$a_1$, $a_2$, $a_3$,...,$a_n$–array names
$k_1$,$k_2$,$k_3$,...,$k_n$ – 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 ($a_1$,$a_2$,$a_3$,...,$a_n$)

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

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

**EQUIVALENCE** (a,b,c,...), (d, e, f,...)

a, b, c, d, e, f,... – 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 $a$*$s_1$ ($k_1$ )/$x_1$ /,$b$*$s_2$ ($k_2$)/$x_2$/,...,$z$*$s_n$ ($k_n$ )/$x_n$/

type – INTEGER, REAL, LOGICAL, or COMPLEX

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

a,b,...,z – variable, array, or function names

($k_1$),($k_2$),...,($k_n$) – 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

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

**Format statement**

x **FORMAT** ($c_1$ ,$c_2$ ,...,$c_n$/$c_1$′,$c_2$′,...,$c_n$′/...)

x – statement number (1 through 5 digits)

$c_1$,$c_2$ ,...,$c_n$ and $c_1$′,$c_2$′,...,$c_n$ ′ – 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

aGw.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 ($a_1$,$a_2$,$a_3$,...,$a_n$)**

type – integer, real, complex, or logical
name – name of FUNCTION subprogram
*s – optional; represents one of permissible length specifications for associated type
$a_1$,$a_2$,$a_3$,...,$a_n$ – nonsubscripted 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 ($x_1$ ,$x_2$ ,$x_3$ ,...,$x_n$ ), i**

$x_1$,$x_2$,...,$x_n$, – executable statement numbers
i – nonsubscripted integer variable in range: $1 \leqslant i \leqslant n$

**GO TO (unconditional statement)**

**GO TO X**

X – executable statement number

**IF (arithmetic statement)**

**IF (a) $x_1$,$x_2$,$x_3$**

a – arithmetic expression; not complex
$x_1$,$x_2$,$x_3$ – 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($a_1$,$a_2$,...),...,type*s($a_1$,$a_2$,...)**

type – integer, real, complex, or logical
*s – optional; one of permissible length specifications for associated type
$a_1$,$a_2$,... – 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,...,i**

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 of 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 ($a_1$,$a_2$,$a_3$,...,$a_n$)**

name – subprogram name

$a_1$,$a_2$,$a_3$,...,$a_n$ – 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 | Special Characters | EBCDIC Punches | BCDIC Punches |
| O | 11-6 | | | |
| P | 11-7 | | | |
| Q | 11-8 | + | 12-6-8 | 12 |
| R | 11-9 | - | 11 | 11 |
| S | 0-2 | / | 0-1 | 0-1 |
| T | 0-3 | = | 6-8 | 3-8 |
| U | 0-4 | . | 12-3-8 | 12-3-8 |
| V | 0-5 | ) | 11-5-8 | 12-4-8 |
| W | 0-6 | * | 11-4-8 | 11-4-8 |
| X | 0-7 | , | 0-3-8 | 0-3-8 |
| Y | 0-8 | ( | 12-5-8 | 0-4-8 |
| Z | 0-9 | ' | 5-8 | 4-8 |
| $ | 11-3-8 | blank | no punch | no punch |
| (see notes) | | (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**

| Function | Name | Definition | I or O* | Arguments (no., type) | Returned Function |
|---|---|---|---|---|---|
| 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 | $\frac{2}{\pi}\int_0^x e^{-u^2}\,du$ | O | 1 Real *4 | Real *4 |
| | DERF | $\frac{2}{\pi}\int_0^x e^{-u^2}\,du$ | O | 1 Real *8 | Real *8 |
| Complemented Error Function | ERFC | $\frac{2}{\pi}\int_x^\infty e^{-u^2}\,du$ | O | 1 Real *4 | Real *4 |
| | DERFC | $\frac{2}{\pi}\int_x^\infty e^{-u^2}\,du$ | 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 | $arg_1$ (mod $arg_2$) | I | 2 Integer *4 | Integer *4 |
| | AMOD | $arg_1$ (mod $arg_2$) | I | 2 Real *4 | Real *4 |
| | DMOD | $arg_1$ (mod $arg_2$) | I | 2 Real *8 | Real *8 |

| Function | Name | Definition | I or O* | Arguments (no., type) | Returned Function |
|---|---|---|---|---|---|
| Absolute value | IABS | \| arg \| | I | 1 Integer *4 | Integer *4 |
| | ABS | | I | 1 Real *4 | Real *4 |
| | DABS | | I | 1 Real *8 | Real *8 |
| | CABS | $\lvert\sqrt{a^2+b^2}\rvert$ for a+bi = arg | O | 1 Complex *8 | Real *4 |
| | CDABS | | O | 1 Complex *16 | Real *8 |
| Truncation | INT | Sign of arg times largest integer ≤ \| arg \| | I | 1 Real *4 | Integer *4 |
| | AINT | | I | 1 Real *4 | Real *4 |
| | IDINT | | I | 1 Real *8 | Integer *4 |
| Largest value | AMAX0 | Max($arg_1$, $arg_2$,...) | I | ≥2 Integer *4 | Real *4 |
| | AMAX1 | | I | ≥2 Real *4 | Real *4 |
| | MAX0 | | I | ≥2 Integer *4 | Integer *4 |
| | MAX1 | | I | ≥2 Real *4 | Integer *4 |
| | DMAX1 | | I | ≥2 Real *8 | Real *8 |
| Smallest value | AMIN0 | Min($arg_1$, $arg_2$,...) | I | ≥2 Integer *4 | Real *4 |
| | AMIN1 | | I | ≥2 Real *4 | Real *4 |
| | MIN0 | | I | ≥2 Integer *4 | Integer *4 |
| | MIN1 | | I | ≥2 Real *4 | Integer *4 |
| | DMIN1 | | I | ≥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 | $C=arg_1+iarg_2$ | 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 Hex | Storage Estimate Dec | Format |
|---|---|---|---|---|---|---|---|
| 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 | ($a_1$,$b_1$,$f_1$,... $a_n$,$b_n$,$f_n$) a=upper limit b=lower limit F=format specification<br>0<br>1<br>2<br>3<br>4<br>5<br>6<br>7<br>8<br>9 | O | 3 | 48 | 168 | <br>hex<br>Logical *1<br>Logical *4<br>Integer *2<br>Integer *4<br>Real *4<br>Real *8<br>Complex *8<br>Complex *16<br>literal character |
| Dump specified storage area and resume processing | PDUMP | ($a_1$,$b_1$,$f_1$,... $a_n$,$b_n$,$f_n$) 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 $>16^{63}$ =2 for no over-or-under-flow =3 for $<16^{-65}$ | 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 protype 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 | 1 byte, absolute expression, right justified |
| | | ,data address | absolute, relocatable, or complex expression |
| | | ,flag values | absolute expression |
| | | ,count | absolute expression |
| | | Channel command operation codes and flag representations are summarized under "Appendix" | |
| | CNOP | byte alignment, word type {double / single} | b=n, where n = 0,2,4, or 6<br>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]<br>[,read only storage]<br>[,variable section length]<br>[,privileged section]<br>[,section includes SYS entry points] | PUBLIC<br>,READONLY<br>,VARIABLE<br>,PRVLGD<br>,SYSTEM |
| [symbol] | DC | [duplication factor]<br>constant-type<br>[ length { bits / bytes } ]<br>[scale]<br>[exponent]<br>constant[,...] | <br>(see *)<br>L.n (see **)<br>Ln (see **)<br>Sn (see ***)<br>En (see †)<br>(see ††) |

**'constant' or (constant)**

| * Type | ** L.n Ln (n= ) | *** Sn | † En | †† Constant specified as | Implied length | Implied alignment | Truncating/ padding side |
|---|---|---|---|---|---|---|---|
| C | 1 bit to 256 bytes | | | 'up to 256 characters' | as needed | byte | right |
| X | | | | 'hex digits' | | | left |
| B | | | | 'binary digits' | | | left |
| F | 1 bit to 8 bytes | -187 to +346 | -85 to +75 | 'decimal digits,...' | 4 | word | left |
| H | | | | | 2 | half word | left |
| E | | 0 to 2L-2[2] | -85 to +75 | | 4 | word | right |
| D | | | | | 8 | double word | right |

[1] L is length constant; no negative scaling

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| P<br>Z | 1 bit to 16 bytes | | | decimal digits ,... | as needed | byte | left |
| A | 1 bit[2] to 4 bytes | | | (any expression,...) | 4 | word | left |

[2] 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 | | | (relocatable expression ,...) | 4 | word | |
| R | 1 bit to 4 bytes | | | (external symbol,...) | 4 | word | |
| S | 2 bytes | | | [3] | 2 | half word | |

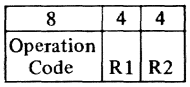
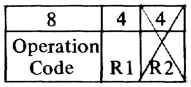
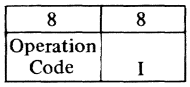
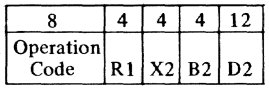
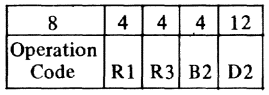
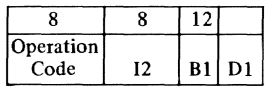
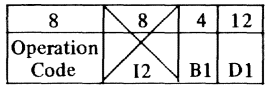
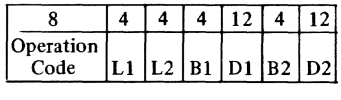
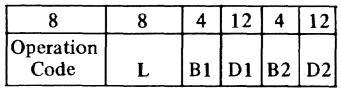
[3] one absolute or relocatable expression, or two absolute expressions

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Y | 1 bit to 2 bytes[4] | | | (any expression | 2 | half word | left |

[4] one absolute or relocatable expression, or two absolute expressions

| | | | |
|---|---|---|---|
| | DROP | reg1 [,reg2,...,reg16] | absolute value |
| [symbol] | DS | [duplication factor]<br>[constant type]<br>[ length { bits / bytes } ]<br>[scale]<br>[exponent] | same as DC except as noted below |
| | | 1. Specification of 'constant' operand, optional<br>2. 'Constant' operand reserves space but does not store data<br>3. Maximum length for C and X constants, 65,535 bytes<br>4. Duplication factor of 0 forces alignment to assumed alignments in DC | |
| [symbol] | DSECT | none | |
| | END | [control transfer point] | relocatable or absolute expression<br>default = first instr of CSECT |
| | ENTRY | entry point [,...] | relocatable symbols |
| | EJECT | none | |
| [symbol] | EQU | self-defining term, previously defined symbol, or combining expression | |
| | EXTRN | external symbol [,...] | relocatable symbols |
| | ICTL | beginning source column<br>[,ending source column]<br>[,continue column] | b=decimal digit, range, 1-40<br>e=decimal digit, range, 41-80; default=71<br>c=decimal digit, range, 2-40; default=16<br>if no ICTL is used, b=1, e=71, c=16. |
| | ISEQ | [sequence field - left col]<br>[sequence field - right col] | decimal digits.<br>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 } | ON<br>OFF |
| | | [ code { executable / all / none } ] | GEN<br>FULLGEN<br>NOGEN |
| | | [ ,constants { print full const / print 8 bytes or less } ] | DATA<br>NODATA |
| | 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,<br>reg1,[,reg2,...,reg16] | absolute or relocatable value,<br>absolute expression |

## Basic Instruction Formats

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

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[4] | Add normalized, double | 66 | RX | R1,D2(X2,B2) |
| ADDR[4] | 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[4] | Add normalized, mixed | 76 | RX | R1,D2(X2,B2) |
| BAL[1] | Branch and link | 45 | RX | R1,D2(X2,B2) |
| BALR[1] | 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[1] | Branch on count | 46 | RX | R1,D2(X2,B2) |
| BCTR[1] | Branch on count | 06 | RR | R1,R2 |
| BXH[1] | Branch on index high | 86 | RS | R1,R3,D2(B2) |
| BXLE[1] | 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(L,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[2] | Insert storage key | 09 | RR | R1,R2 |
| L | Load | 58 | RX | R1,D2(X2,B2) |
| LA[3] | 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[2] | 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[2] | Load PSW | 82 | SI | D1(B1) |
| LR | Load | 18 | RR | R1,R2 |
| LRA[2] | 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[4] | 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[4] | Multiply normalized, double | 65 | RX | R1,D2(X2,B2) |
| MDDR[4] | 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[2] | 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[4] | Subtract normalized, double | 67 | RX | R1,D2(X2,B2) |
| SDDR[4] | 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[2] | 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[4] | 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[2] | Set system key | 08 | R | R1,R2 |
| SSM[2] | 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) |
| STMC[2] | Store multiple control | B0 | RS | R1,R3,D2(B2) |
| STRD[4] | Store rounded, long | 61 | RX | R1,D2(X2,B2) |
| STRE[4] | 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[4] | Subtract normalized, mixed | 77 | RX | R1,D2(X2,B2) |
| TCH[2] | Test channel | 9F | SI | D1(B1) |
| TIO[2] | Test I/O | 9D | SI | D1(B1) |
| TM | Test under mask | 91 | SI | D1(B1),I2 |
| TR | Translate | DC | SS | D1(L,B1),D2(B2) |
| TRT | Translate and test | DD | SS | D1(L,B1),D2(B2) |
| TS | Test and set | 93 | SI | D1(B1) |
| UNPK | Unpack | F3 | SS | D1(L1,B1),D2(L2,B2) |
| WRD[2] | Write direct | 84 | SI | D1(B1),I2 |
| X | Exclusive OR | 57 | RX | R1,D2(X2,B2) |
| XC | Exclusive OR | D7 | SS | D1(L,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) |

[1] Limited to 24-bit addressing capability; exercise caution when using in 32-bit mode.

[2] Privileged instructions.

[3] Users in 24-bit mode must not exceed 24-bit limitation when loading address or using LA adding technique.

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

## Assembler User Macro Instructions

### DATA SET MANAGEMENT

**Define Data Set**

DDEF CDD
DCB DEL
CAT DCBD
FINDDS* FINDJFCB*

**Connect Data Set to System**

OPEN DUOPEN (VAM only)

**Access Data Set**

| VSAM | VISAM | VPAM |
|---|---|---|
| GET | GET | FIND |
| PUT | PUT | STOW |
| PUTX | READ | **IOREQ** |
| SETL | WRITE | IOREQ |
| **BSAM** | SETL | CHECK |
| READ | ESETL | VCCW |
| WRITE | DELREC | |
| CHECK | RELEX | |
| DQDECB | **QSAM** | |
| GETPOOL | GET | |
| GETBUF | PUT | |
| FREEBUF | PUTX | |
| FREEPOOL | RELSE | |
| BSP | TRUNC | |
| CNTRL | CNTRL | |
| FEOV | SETL | |
| POINT | | |
| NOTE | | |

**Manipulate Entire Data Set**

| *Copy Data Set* | *Bulk Output* |
|---|---|
| CDS | PR |
| | PU |
| | WT |

**Catalog Data Set Attributes**

CAT
DEL

**Disconnect Data Set from System**

CLOSE DUPCLOSE (VAM only)

**Remove Data Set from System**

ERASE
REL

**Language Processing**

LPCEDIT
LPCINIT

### PROGRAM CONTROL MANAGEMENT

**Virtual Storage Management**

GETMAIN DCLASS*
FREEMAIN RSPRV*
CSTORE LSCHP*
CKCLS*

**Program Linking and Loading**

ADCOND DELETE
ADCON SAVE
ARM RETURN
CALL DELET*
LOAD DLINK*
ENTER* RESUME*
LIBESRCH* STORE*

**Interruption Handling**

SIR SIEC
SPEC DIR
SSEC SAI
SEEC RAE
SAEC INTINQ
STEC CLATT
USATT AETD
ATPOL* ITI*
PTI*

**Transfer to Command Mode**

PAUSE ABEND
COMMAND OBEY
EXIT RTRN*
CLIC* CLIP*

**Communications with SYSIN/SYSOUT**

GATRD GTWRC MCAST
GATWR GTWSR
GTWAR SYSIN
MSGWR PRMPT

**Communications with Operator and Log**

WTO WTL
WTOR WTOA

**Task/Timing Maintenance**

STIMER EBCDTIME USAGE
TTIMER REDTIM*

**Command Creation**

BPKD GDV

**System-Oriented Macro Instructions**

AUXPG* VSENDR*
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 $ @ # |
| alphameric | A - Z $ @ # 0-9 |
| special characters | + - , = . * ( ) / & and blank |
| special symbols | 1-6 alphameric characters and + - . * ' / & |
| relocatable expression | changes when program is located |
| absolute expression | value remains constant when program is relocated |
| 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 \| IMPLICIT \| INTERNAL } [,EP=symbol]<br>[,LDERR={pass return codes - CODE \| take error exit - ERR}]<br>[,DELOPT={specified module only - SMO \| all associated modules - SDM}]<br>[,HSHTAB={XPDS \| NORM}] |
| | ADCOND | none |
| symbol | AETD | [( {ep-symbol, sa-symbol} ,...)] |
| [symbol] | ARM | loc-addrx,extref-addrx |
| | | return data:<br>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<br>,area-addr[,length {'S'\|value}] |
| [symbol] | WRITE | decb-symbol,type-SF,dcb-addr,area-addr<br>[,length {'S'\|value}] |
| [symbol] | CHECK | decb { addrx \| (1) } |
| [symbol] | DQDECB | decb { addrx \| (1) } |
| | | return data:<br>reg 0 = number of unchecked DECBs in queue<br>reg 1 = pointer to list of unchecked DECBs |
| [symbol] | GETPOOL | dcb { addrx \| (1) } , { number-value,length-value \| (0) } |
| [symbol] | GETBUF | dcb { addrx \| (1) } ,register-absexp |
| [symbol] | FREEBUF | dcb { addrx \| (1) } ,register-absexp |
| [symbol] | FREEPOOL | dcb { addrx \| (1) } |

| Name | Operation | Operands |
|---|---|---|
| | **BSAM** | (Continued) |
| [symbol] | BSP | dcb {addrx / (1)}<br>return data:<br>reg 15 = 00 if normal completion or if permanent positioning error occurs<br>04 backspacing did not occur |
| [symbol] | CNTRL | dcb {addrx / (1)} , {action-code[,number-value] / (0)} |
| [symbol] | FEOV | dcb {addrx / (1)} |
| [symbol] | POINT | dcb {addrx / (1)} ,loc {addrx / (0)} |
| [symbol] | NOTE | dcb {addrx / (1)}<br>return data:<br>reg 1 = block ID of last block read or written |

| | | |
|---|---|---|
| [symbol] | CALL | entry {symbol / (15)} ,[(param-addr,...)],[VL]<br>[,loadtype-{E / I}] [,ID=absexp] |
| | | register setting:<br>reg 1 = contains address of parameter list<br>reg 14= valid return address |
| [symbol] | CAT | oplist {text / addr} |
| | | where oplist equals:<br>{dsname$_1$-name,state{N / U},[access{R / U}],[dsname$_2$-name] / GDG=name,gnumber-integer,[action{A / D}] [,ERASE={Y / N}]} |
| | | return data:<br>reg 15 = code<br>(hex) significance<br>00 cataloged<br>04 name cannot be changed since new dsname not unique; no cataloging<br>08 invalid element in input string<br>0C no cataloging for other reasons<br>10 data set name not unique, already in catalog<br>14 data set volume not mounted, cannot catalog<br>18 ABEND request<br>20 Vam data set not GDG or rename option<br>24 Open DCB |

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

*(continued)*

<table>
<tr><th>Name</th><th>Operation</th><th>Operand</th></tr>
<tr><td></td><td></td><td>34 data set copied and original erased, new data set not renumbered, not a line data set<br>38 data set copied, new data set not renumbered, old data set not erased</td></tr>
<tr><td>[symbol]</td><td>CLATT</td><td>none</td></tr>
<tr><td>[symbol]</td><td>CLOSE</td><td>({dcb-addr,[opt{REREAD / LEAVE}]},...)[,TYPE=T]</td></tr>
<tr><td>[symbol]</td><td>COMMAND</td><td>[message {text / addrx / (1)}]</td></tr>
<tr><td>[symbol]</td><td>CSTORE</td><td>module-symbol,epname-symbol,address-addr,length-value,attribute-value<br><br>return data:<br>reg 15 = Code Significance<br>00 normal return<br>04 module name or entry point name already in use</td></tr>
<tr><td></td><td>DCB</td><td>
<table>
<tr><th></th><th colspan="6">Applicable Access Methods</th></tr>
<tr><th></th><th>VSAM</th><th>VISAM</th><th>VPAM</th><th>BSAM</th><th>QSAM</th><th>IOREQ</th></tr>
<tr><td>[,DDNAME=symbol]</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<br>VSP</td><td>VI<br>VIP</td><td>VP</td><td>PS</td><td>PS</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></tr>
<tr><td>[,LRECL=absexp]</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</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></tr>
<tr><td>[,OPTCD=code]</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td></tr>
<tr><td>[,SYNAD=symbol]</td><td></td><td>X</td><td>X*</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>[,PAD=absexp]</td><td></td><td>X</td><td>X*</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></tr>
<tr><td>[,DEVD=code]</td><td></td><td>1</td><td>1</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>[,KEYLEN=absexp]</td><td></td><td>X</td><td>X*</td><td>X</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>X</td></tr>
<tr><td>[,PRTSP=absexp]</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>X</td></tr>
<tr><td>[,STACK=absexp]</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></tr>
<tr><td>[,BLKSIZE=absexp]</td><td></td><td></td><td></td><td>X</td><td>X</td><td></td></tr>
<tr><td>[,IMSK=code]</td><td></td><td></td><td></td><td>X</td><td>X</td><td></td></tr>
<tr><td>[,EXLST=symbol]</td><td></td><td></td><td></td><td>X</td><td>X</td><td></td></tr>
<tr><td>[,NCP=absexp]</td><td></td><td></td><td></td><td>X</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></tr>
<tr><td>[,BFALN=code]</td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>[,BUFL=absexp]</td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>[,BFTEK=code]</td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>[,BUFCB=addr]</td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>[,EROPT=code]</td><td></td><td></td><td></td><td></td><td>X</td><td></td></tr>
</table>
* = VISAM members of partitioned data set<br>
1. = some established value assumed by system
</td></tr>
</table>

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

| Name | Operation | Operands |
|---|---|---|
| [symbol] | DQDECB | decb {addrx / (1)} |
| | | return data:<br>reg 0 = number of unchecked DECBs in the queue<br>reg 1 = pointer to list of unchecked DECBs |
| | DUPCLOSE | $dcb_1$-addr,$dcb_2$-addr |
| | DUPOPEN | $dcb_1$-addr,$dcb_2$-addr[,opt-code] |
| | | For valid codes, see OPEN |
| [symbol] | EBCDTIME | oplist {text / addr},[time-addr][,L={integer / (register)}] |
| [symbol] | ERASE | dsname {text / addr} |
| | | return data: fourth byte of general register 15 set to:<br>code — significance<br>00 no error detected<br>04 not class D or batch monitor entry<br>08 invalid return code<br>0C invalid delimiters in data set name<br>10 no data set name supplied<br>14 invalid return code from CHEKDS module<br>18 data set name not in catalog or TDT<br>1C partitioned data set not fully qualified name<br>20 member of partitioned data set not found in POD<br>24 data set not cataloged<br>28 data set on public volume<br>2C data set is member of partitioned data set<br>30 user does not own data set in ERASE batch monitor entry<br>34 sharing/access conflicts prevent processing<br>38 no catalog entry for ERASE batch monitor entry<br>3C data set name undefined; return code from DDEF<br>44 data set not on direct access<br>48 volume not found<br>4C system JFCB<br>58 data set in use<br>5C resources exceeded |
| [symbol] | EXIT | message {text / addrx / (1)} [,NOMSG] |
| [symbol] | FREEMAIN | {PAGE[,VAR] / R},LV={value / (0)},A={addrx / (1)} |
| | | return data:<br>reg 15 = code (hex) — significance<br>00 normal return<br>04 unsuccessful, no storage freed |

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

| | | |
|---|---|---|
| | | 8 attention interruption occurred during input or output; record unpredictable<br>10 SYSIN/SYSOUT received from terminal keyboard<br>20 SYSIN received from card reader at terminal |
| [symbol] | GATWR | msg-addr,length-addr |
| | | return data: see GATRD codes 8 and 10 |
| [symbol] | GDV | [ com- { addrx \| text \| (1) } ] |
| [symbol] | GETMAIN | { PAGE[,VAR] \| R } ,LV= { value[,PR=integer] \| (0) }<br>[,PACK=0\|1\|2] { }<br>[,EXIT=RETURN] |
| | | return data:<br>reg 1 = address of allocated virtual storage<br>reg 15 = code<br>(hex) significance<br>00 normal return<br>04 request cannot be satisfied<br>08 bytes specified with invalid protection class |
| [symbol] | GTWAR | msgout-addr,lengthout-addr,<br>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,<br>msgin-addr,lengthin-addr[,SIC] |
| | | return data: see GATRD |
| [symbol] | INTINQ | icb-addr [ ,MODE= { R \| W \| CLEAR \| ( C, branch-addr, TYP-code ) } ] |
| | | return data:<br>reg 15 = 00 normal<br>04 undefined routine specified (modes C, W, CLEAR)<br>08 bad parameter list, conditions specified cannot be met (modes W, C) |

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

| | | |
|---|---|---|
| [symbol] | IOREQ | decb-symbol, type- { N \| B } ,dcb-addr,<br>vccw-addr,length-value,sio-value ,IMSK=code |
| | | return data:<br>reg 15 = code<br>(dec) significance<br>0 I/O initiated<br>4 NCP value in data control block exceeded (I/O not initiated), or DECB "active" or in "wait" state<br>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)<br>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 {WRITE \| 01 / READ \| 02 / NOP \| 03 / SENSE \| 04 / TIC \| 08 / READBK \| 0C} / absolute expression}<br>data-relexp<br>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]<br>return codes<br>rlength = actual byte length of response<br>reg 15 = code — significance<br>0 — no attention interrupt: no error in response length (if applicable)<br>4 — response too long for area specified<br>8 — attention interrupt occurred; response unpredictable. |
| [symbol] | OBEY | [com- {addrx / text / (1)}] |
| [symbol] | OPEN | ({dcb-addr,[($opt_1$-code[,$opt_2$-code])]},...)<br>where $opt_1$ code can be:<br>INPUT — input data set; value is assumed if opt is omitted<br>OUTPUT — output data set<br>INOUT — data control block opened as if INPUT opt was specified: both input and output operations allowed<br>OUTIN — data control block opened as if OUTPUT opt was specified; input and output operations allowed<br>RDBACK — positions an input data set to be read backward (magnetic tape only)<br>UPDAT — allows updating of existing data set (direct-access devices only) |
| [symbol] | PAUSE | [message- {text / addrx / (1)}] |

| | | |
|---|---|---|
| [symbol] | PR | oplist- { text / addr }<br><br>where oplist equals:<br>dsname-name,[startno-integer],[endno-integer],<br>[ spacing- { EDIT / { 1 / 2 / 3 } ,[H],[lines-integer],[P] } ]<br>[,ERASE={ Y / N }] ,[error- { ACCEPT / SKIP / END }] [,form-specysm]<br><br>return data:<br>reg 1 = batch sequence number<br>reg 15 = code: 0 — PR request accepted. All other codes — register 15 contains two-byte message number. |
| [symbol] | PRMPT | msgid- { addr / text } ,resp opt { N / P / U } [,user resp-addr]-<br>[,( { param { addr / text } } ,... )]<br><br>return codes:<br>reg 15 = code: 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 predefined 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 |
| [symbol] | PU | oplist- { text / addr }<br><br>where oplist equals:<br>dsname-name,,[startno-integer],[endno-integer],<br>[select- { 1 / 2 / 3 / EDIT } ],[,ERASE={ Y / N }] [,form-specsym]<br><br>return data:<br>reg 1 = batch sequence number<br>reg 15 = Code: 0 — PU request was accepted; All other codes — register 15 contains two-byte message number |

**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] | RELSE | dcb-{addrx \| (1)} |
| [symbol] | TRUNC | dcb-{addrx \| (1)} |
| [symbol] | SETL | dcb-{addrx \| (1)},type{C\|R\|B\|E\|P}<br>[,llimit-{addrx \| (0)}] |

| | | |
|---|---|---|
| [symbol] | RAE | [area-addr] |
| [symbol] | REL | oplist-{text \| addr} |
| | | where oplist equals:<br>ddname-symbol[,dsname-name] |
| | | return data:<br>reg 15 = code<br>(hex) significance<br>00 normal<br>04 defaulted, invalid ddname<br>08 attention interrupt<br>0C reserved ddname specified<br>10 undefined ddname<br>14 uncataloged on public storage<br>18 undefined dsname<br>20 spurious input<br>24 unable to unload all modules loaded from library |
| [symbol] | RETURN | [($reg_1$-integer[,$reg_2$-integer])][,T]<br>[,RC={absexp \| (15)}] |
| [symbol] | SAEC | [EP={symbol \| 0} [,DCB=addr]]<br>[,COMAREA=addr]<br>[{INTTYP \| ATTNTYP}={([A\|S\|R] [,code] ...) \| NULL \| SAVE \| RESTORE}]<br>[,MF=L\|(E,icb-{addrx \| (1)})] |

| | | |
|---|---|---|
| [symbol] | SAI | [area-addr] |
| [symbol] | SAVE | (reg$_1$ -integer[,reg$_2$ -integer]),[T]<br>[,entry point ID -{characters / *}]<br>where * denotes name field symbol |
| [symbol] | SEEC | [EP={symbol / 0}] [,COMAREA=addr]<br>[,INTTYP=integer] [,MSGAREA=addr]<br>[,MSGLTH=integer] [,MF=L\|(E,icb-{addrx / (1)})] |
| [symbol] | SIEC | [EP={symbol / 0}] [,DCB=addr]<br>[,COMAREA=addr][,MF=L\|(E,icb-{addrx / (1)})] |
| [symbol] | SIR | (icb-addr,...)[,PRTY-integer]<br>[,INHIBIT={YES\|NO}]<br>return data:<br>reg 1 = address of invalid ICB<br>reg 15 = code — condition<br>04 — ICB contains invalid DCB (for input/output, asynchronous ICBs only), or invalid time or clock number specified<br>08 — ICB specified previously by another SIR macro instruction<br>0C — invalid parameter (ICB, length invalid, nonprivileged user requested privileged priority or has privileged flag on in parameter area)<br>10 — total user time will exceed 7.5 hours; time not set |
| [symbol] | SPEC | [EP=symbol] [,COMAREA=addr]<br>[,INTTYP={NULL / SAVE / RESTORE / (({A\|S\|R},{ALL / intype{code-integer / integer-integer / integer-range},...}))}]<br>[,MF=L\|(E,icb-{addrx / (1)})] |
| [symbol] | SSEC | [EP={symbol / 0}] [,COMAREA=addr]<br>[,INTTYP=integer]<br>[,MF=L\|(E,icb-{addrx / (1)})] |
| [symbol] | STEC | [EP={symbol / 0}] [,COMAREA=addr]<br>[,INTTYP={TASKn,{DINTVL=addr / BINTVL=addr} / REALn{DINTVL=addr / BINTVL=addr / TOD=addr / DO{W\|M\|Y}=addr}}]<br>[,MF=L\|(E,icb-{addrx / (1)})] |

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

**VSAM (Virtual sequential access method)**

| | | |
|---|---|---|
| [symbol] | GET | dcb-{addr \| (1)} [,area-{addrx \| (0)}]<br>return data:<br>reg 1 = in locate mode, address of next sequential record |
| [symbol] | PUT | dcb-{addrx \| (1)} [,area-{addrx \| (0)}]<br>return data:<br>reg 1 = in locate mode, address of output buffer for next sequential output record |
| [symbol] | PUTX | dcb-{addrx \| (1)} |
| [symbol] | SETL | dcb-{addrx \| (1)},type-{R \| B \| E \| P} [,llimit-{addrx \| (0)}] |

**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 | decb-symbol,type-{KY \| KZ \| KX},dcb-addr ,area-addr, key-addr |
| [symbol] | WRITE | decb-symbol,type{- KR \| KS \| KT},dcb-addr ,area-addr, key-addr |
| [symbol] | SETL | dcb-{addrx \| (1)} ,type-{R\|B\|E\|P\|K\|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]<br>return data:<br>reg 0 = length of user data in POD<br>reg 1 = points to parameter list<br>reg 15 = code<br>(hex) significance<br>00 successful completion of FIND<br>04 Member or alias not located by FIND<br>08 data control block, indicated in macro instruction, in use to create member; execution of STOW must be complete before FIND<br>10 length specified in macro instruction not large enough for user data<br>14 member to be located already open for this data control block, due to previous FIND |
| [symbol] | STOW | dcb {addrx \| (1)} , [area-{addrx \| (0)}]<br>,type {N\|NA\|NAR\|R\|U\|D\|DA\|C\|CA}<br>return data<br>bits 24-31 reg 15:<br>code (hex) significance<br>00 successful completion of STOW<br>04 new name or alias already in use (N, NA, C, or CA)<br>08 member name not in POD (U, D, DA, or CA)<br>10 old member name not in POD (C), alias not in POD (DA), old alias not in POD (CA)<br>14 invalid STOW type requested<br>18 user data exceeds maximum length of 510 bytes |

| Name | Operation | Operand |
|---|---|---|
| [symbol] | WT | oplist-{text / addr}<br>where oplist equals:<br>dsname$_1$-name,dsname$_2$-name,[volume-alphnum],[factor-integer],[startno-integer],[endno-integer]<br>[,spacing-{EDIT / {1 / 2 / 3},[H],[lines-integer],[P]}]<br>[,ERASE={Y / N}]<br>return data:<br>reg 1=address of batch sequence number<br>reg 15 = code: 0 — WT request accepted; All other codes — register 15 contains two-byte message number |
| [symbol] | WTL | message-text<br>return data:<br>reg 15 = code<br>(hex): 0 successful; 4 attention interruption; C invalid message length, no message sent |
| [symbol] | WTO | message-text<br>return data:<br>reg 15 = code<br>(hex): 0 successful; 4 attention interruption; C invalid message length, no message sent |
| [symbol] | WTOA | message-text<br>return data:<br>reg 15=code<br>(hex): 0 successful; 4 attention interruption; C invalid message length; no message sent |
| [symbol] | WTOR | message-text,reply-addr,length-value<br>return data:<br>reg 15 = code<br>(hex): 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





# APPENDIX

**Permanent Storage Assignment**

| Address Dec | Hex | Binary | Length | Purpose |
|---|---|---|---|---|
| 0 | 0 | 0000 0000 | double word | IPL PSW } overlayed |
| 8 | 8 | 0000 1000 | double word | IPL CCW1 } after |
| 16 | 10 | 0001 0000 | double word | IPL CCW2 } IPL |
| 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.

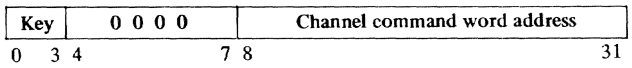
To find a hex equivalent of a decimal number: (A) Locate the decimal number that is next-lower than that number; this is the first hex digit; (b) subtract the lower decimal number from the original decimal number; (c) repeat step A for the decimal difference, to get the next hex digit; then continue 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 |
| 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 |
| 2 | 2,097,152 | 2 | 131,072 | 2 | 8,192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 3,145,728 | 3 | 196,608 | 3 | 12,288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 4,194,304 | 4 | 262,144 | 4 | 16,384 | 4 | 1,024 | 4 | 64 | 4 | 4 |
| 5 | 5,242,880 | 5 | 327,680 | 5 | 20,480 | 5 | 1,280 | 5 | 80 | 5 | 5 |
| 6 | 6,291,456 | 6 | 393,216 | 6 | 24,576 | 6 | 1,536 | 6 | 96 | 6 | 6 |
| 7 | 7,340,032 | 7 | 458,752 | 7 | 28,672 | 7 | 1,792 | 7 | 112 | 7 | 7 |
| 8 | 8,388,608 | 8 | 524,288 | 8 | 32,768 | 8 | 2,048 | 8 | 128 | 8 | 8 |
| 9 | 9,437,184 | 9 | 589,824 | 9 | 36,864 | 9 | 2,304 | 9 | 144 | 9 | 9 |
| A | 10,485,760 | A | 655,360 | A | 40,960 | A | 2,580 | A | 160 | A | 10 |
| B | 11,534,336 | B | 720,896 | B | 45,056 | B | 2,816 | B | 176 | B | 11 |
| C | 12,582,912 | C | 786,432 | C | 49,152 | C | 3,072 | C | 192 | C | 12 |
| D | 13,631,488 | D | 851,968 | D | 53,248 | D | 3,328 | D | 208 | D | 13 |
| E | 14,680,064 | E | 917,504 | E | 57,344 | E | 3,584 | E | 224 | E | 14 |
| F | 15,728,640 | F | 983,040 | F | 61,440 | F | 3,840 | F | 240 | F | 15 |
| 6 | | 5 | | 4 | | 3 | | 2 | | 1 | |

**Hex-Binary Power Table**

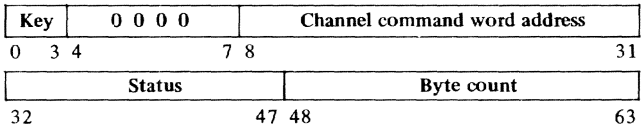
| POWERS OF 16 | | POWERS OF 2 | |
|---|---|---|---|
| $16^n$ | n | $2^n$ | n |
| 1 | 0 | 512 | 9 |
| 16 | 1 | 1 024 | 10 |
| 256 | 2 | 2 048 | 11 |
| 4 096 | 3 | 4 096 | 12 |
| 65 536 | 4 | 8 192 | 13 |
| 1 048 576 | 5 | 16 384 | 14 |
| 16 777 216 | 6 | 32 768 | 15 |
| 268 435 456 | 7 | 65 536 | 16 |
| 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



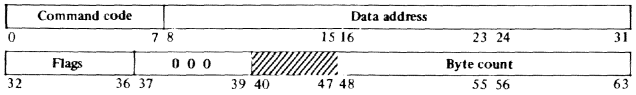

| Bits | Meaning |
|---|---|
| 0-3 | storage protection key = 0-15 |
| 4-7 | must be all 0s |
| 8-31 | address of first CCW in main storage associated with start-I/O instruction |

## Channel Status Word




| 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 transfor 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 8-bit code 0123 4567 | Hex | Dec | M-T on† 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 | | |
| Switch-ing | Release device | X | 1001 0100 | 94 | 148 | | |
| | 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 | Note 1 | | | | | |
| | EQ* | | 0010 0101 | 25 | 37 | | |
| | HI* | | 0100 0101 | 45 | 69 | | |
| | EQ or HI* | | 0110 0101 | 65 | 101 | | |
| | no compare* | | 0101 0101 | 55 | 85 | | |
| | set compare* | | 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 | 0000 0110 | 06 | 06 | 86 | 134 |
| | Key & data | of bytes | 0000 1110 | 0E | 14 | 8E | 142 |
| | Count, key & data | trans-ferred | 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 | DL | 0000 0101 | 05 | 05 | | |
| | Key & data | KL & DL | 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

## Channel Command Codes

| Device | Command for CCW | | 8-bit code P | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Hex | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2702 | Write | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01 | |
| | Read | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02 | |
| | Sense | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04 | |
| | **Orders** | | | | | | | | | | | | |
| | Autowrap | | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 05 | |
| | Dial | | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 29 | |
| | Break | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0D | |
| | Prepare | | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 06 | |
| | Inhibit | | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0A | |
| | Search | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0E | |
| | Sadzer | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 13 | |
| | Sadone | | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 17 | |
| | Sadtwo | | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1B | |
| | Sadthree | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1F | |
| | Enable | | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 27 | |
| | Disable | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 2F | |
| | Release | | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | D4 | |
| | Reserve | | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | F4 | |
| | I/O no-op | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 03 | |
| | Test I/O | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00 | |
| 1052 | Read inquiry BCD | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0A | 10 |
| | Read reader 2BCD | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02 | 02 |
| | Write BCD, auto carriage return | | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 09 | 09 |
| | Write BCD, no auto carriage return | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01 | 01 |
| | No Op | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 03 | 03 |
| | Sense | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04 | 04 |
| | Alarm | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0B | 11 |
| 2540 | Read, feed, select stacker SS | Type AA | | S | S | D | 0 | 0 | 0 | 1 | 0 | | |
| | Read | Type AB | | 1 | 1 | D | 0 | 0 | 0 | 1 | 0 | | |
| | Read, feed (1400 compatibility mode only) | | | 1 | 1 | D | 1 | 0 | 0 | 1 | 0 | | |
| | Feed, select stacker SS | Type BA | | S | S | 1 | 0 | 0 | 0 | 1 | 1 | | |
| | PFR punch, feed, select stacker SS | Type BA | | S | S | D | 0 | 1 | 0 | 0 | 1 | | |
| | Punch, feed, select stacker SS | Type BB | | S | S | D | 0 | 0 | 0 | 0 | 1 | | |

| SS | Stacker |
|---|---|
| 00 | R1 |
| 01 | R2 |
| 10 | RP3 |

| D | Data Mode |
|---|---|
| 0 | EBCDIC |
| 1 | column binary |

| Device | Command for CCW | 8-bit code 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Hex | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 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 | | |

| C | H | A | N | Channel |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |

| C | H | A | N | Channel |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 |
| 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 0 | 0 | 12 |

* 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.

| C | C | C | Control codes | Hex | Dec |
|---|---|---|---|---|---|
| 0 | 0 | 0 | REW | 7 | 7 |
| 0 | 0 | 1 | RUN | 0F | 15 |
| 0 | 1 | 0 | ERG | 17 | 23 |
| 0 | 1 | 1 | WTM | 1F | 31 |
| 1 | 0 | 0 | BSR | 27 | 39 |
| 1 | 0 | 1 | BSF | 2F | 47 |
| 1 | 1 | 0 | FSR | 37 | 55 |
| 1 | 1 | 1 | FSF | 3F | 63 |

| D | D | 7-Track density |
|---|---|---|
| 0 | 0 | 200 |
| 0 | 1 | 556 |
| 1 | 0 | 800 |
| 1 | 1 | 800 |

**Overrides data Converter On

| 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 | | |

## Virtual PSW

| P | Not Used | Task mask X | A | T | I | ILC | CC | FO | DO | EU | SF | Interrupt code |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3 | | | 7 | 8 9 | 10 11 | 12 | 13 | 14 | 15 | 16 31 |

| Instruction address |
|---|
| 32 63 |

| Bit | Meaning |
|---|---|
| 0 | privilege-state mask {0=privileged, 1=nonprivileged} |
| 1-3 | not used |
| 4-7 | Task mask (1=allows interrupts; 0=disallows interrupts) |
| 4 | X=external interrupt mask |
| 5 | A=asynchronous interrupt mask |
| 6 | T=timer interrupt mask |
| 7 | I=synchronous interrupts |
| 8-9 | ILC=instruction length code |

| If ILC bit is | then | Instruction length is | and | OP-code bits were |
|---|---|---|---|---|
| 01 | | 1 half word | | 00 |
| 10 | | 2 half words | | 01 or 10 |
| 11 | | 3 half words | | 11 |

10-11 CC = condition code

| CC bits | Meaning | Branch-on-condition |
|---|---|---|
| 00 | 0-balance | equal - 1000 |
| 01 | negative | low - 0100 |
| 10 | positive | high 0010, 0001 |
| 11 | overflow | combination - 1101 |

| | |
|---|---|
| 12-15 | program masks for special conditions |
| 12 | fixed point overflow mask |
| 13 | decimal overflow mask |
| 14 | exponential overflow |
| 15 | loss of significance |
| 16-31 | interrupt codes (see table, later in Appendix) |
| 32-63 | instruction address (32-bit virtual address) |

Note: Task dumps use Virtual PSW.

## Extended Program Status Word

| Spare | 24-32 bit mode | Transla. ctrl. | I/O mask | Ext. mask | Protec. key | AMWP | ILC | CC | Prog. mask | Spare |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 3 | 4 | 5 | 6 | 7 | 8 11 | 12 15 | 16 17 | 18 | 19 20 23 | 24 31 |

| Instruction Address |
|---|
| 32 63 |

| Bit | Meaning |
|---|---|
| 0-3 | spare (must be 0s) |
| 4-5 | DAT mode and control bits |
| 4 | 24- or 32-bit address mode |
| 5 | translation control |

| Bit 4 | Bit 5 | Mode of Operation |
|---|---|---|
| 0 | 0 | no translation, except for LRA operand; 24-bit logical address (LRA only) |
| 0 | 1 | translation for all program-generated addresses; 24-bit logical address |
| 1* | 0 | no translation except for LRA operand; 32-bit logical address (LRA only) |
| 1* | 1 | translation for all program-generated addresses; 32-bit logical address |

*Specification exception, if no 32-bit option; recognized as part of execution of first instruction after extended PSW is loaded.

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

| PSW | Control register 6 | | |
|---|---|---|---|
| Bit 13 | 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 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—Interruption Codes

| Interruption | Interruption code (PSW Bits 16-31) | | Dec. | Hex. | ILC | 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 | uuu1uuuu | | | 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 |

1. Abbreviations:
   - **u** - unpredictable
   - r - R1 and R2 fields in supervisor call instruction
   - a - I/O device address
   - b - I/O channel address

2. 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.

3. 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 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.

4. 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.

5. 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 interrupts 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 thru 31 | 01 thru 1F | | As in preceding "TSS/360 Interruption Codes" |
| 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 | nonpriviliged 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 $255_{10}$ 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

## 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. logon smith,john
 2.  TSS/360 LEVEL 8.C
     TASKID=0040 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.G0060V00, ACCESS:  RW
    SMITH***.LIST.SORT.G0001V20, ACCESS:  RW
    SMITH***.ASMMAC.S0000V02, ACCESS:  RO, OWNER:  TSS*****
    SMITH***.ASMNDX.S0000V02, ACCESS:  RO, OWNER:  TSS*****
 5. ddef lib,vb,,tibl,option=joblib
 6. edit source.invest
    0000100 C      investment return calculation
    0000200  1     rate=.06
    0000300  2     prin=5000.00
    0000400  3     annual=1000.00
    0000500  4     yrs=10
    0000600        prin=prin*(1+rate)+annual
    0000700  15    print 16,1,prin
 7. 0000800 insert 500,10
    0000510 C      do 100 i=1,iyrs
 8. 0000520 insert 700
    0000800  100   continue
    0000900  16    format (1x,i6,f12.2)
    0001000        stop
    0001100        end
 9. 0001200 list
    0000100 C      INVESTMENT RETURN CALCULATION
    0000200  1     RATE=.06
    0000300  2     PRIN=5000.00
    0000400  3     ANNUAL=1000.00
    0000500  4     YRS=10
    0000510        DO 100 I=1,IYRS
    0000600        PRIN=PRIN*(1+RATE)+ANNUAL
    0000700  15    PRINT 16,1,PRIN
    0000800  100   CONTINUE
    0000900  16    FORMAT (1X,I6,F12.2)
    0001000        STOP
    0001100        END
*10. erd, ftn name=invest,stored=y
 11. call invest
 12.      1     6300.00
    CHCRW420     TERMINATED: STOP
 13. qualify invest, display prin,yrs
    PRIN=+.62999951E+04
    YRS=+.10000000E+02
 14. display iyrs
*   IYRS=0
 15. unload invest, edit source.invest
    0001200  context SCO,,yrs,iyrs; list SCO
    0000500  4     IYRS=10
    end; ftn name=invest,stored=y
    0164          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     6300.00
          2     7677.99
          3     9138.66
          4    10686.98
          5    12328.19
          6    14067.87
          7    15911.93
          8    17866.64
          9    19938.63
         10    22134.93
    CHCRW400     TERMINATED: STOP
17. dss? source.invest
    SMITH***.SOURCE.INVEST
    VOLUME: P38229 (2314)
    ORGANIZATION: VI                    PAGES: 0000001
    REFERENCE DATE: 254/70              CHANGE DATE: 254/70
    RECORD FORMAT: V                    RECORD LENGTH: 0000132
    KEY LENGTH: 0020007                 KEY POSITION: 0000004
18. pog? lib1
    INVEST
19. permit source.invest,n,ro,*all
20. logoff
21. B007          LOGOFF AT 14:18 ON 09/11/70
    TERMINAL LOGICALLY DISCONNECTED, RECONNECT OR HANG UP.
```

## 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 assgined 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:

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

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. logon smith,john
2.  TSS/360 LEVEL 8.0
     TASKID=0048 LOGON AT 14:19 ON 09/11/70
3. edit source.binom
4. 0000100        write (1,10)
   0000200  10    format (1x, 'enter input in the form aab')
   0000300        read (2,11)maa,i
   0000400  11    format (i2,i1)
   0000500  20    k=1
   0000600  30    do 100 j=1,i
   0000700  40    k=k*j
   0000800  50    maa=maa*(maa-1)
   0000900  100   continue
   0001000  110   num=maa/k
   0001100        write (1,120)num
   0001200  120   format (i10)
   0001300        stop
   0001400        end
   0001500 _end
5. ftn binom,stored=y
6. call binom
   ENTER INPUT IN THE FORM AAB
   074
7. -21306501
   CHCRW400     TERMINATED: STOP
8. qualify binom
9. at 100, display maa,i,k,j
   00001
10. call binom
    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=-511355038
    I=4
    K=24
    J=4
11. -21306501
    CHCRW400     TERMINATED: STOP
```

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     PROCEEDING(PCS): ALL AT STATEMENTS REMOVED BY UNLOAD COMMAND IN BINOM
    edit source.binom
    0001500 _update
    550             msave=maa
    800  50         if(j-1) 50,100,60
    850  60         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
    CHCRW400     TERMINATED: STOP
15. list? source.binom
    0000100       WRITE (1,10)
    0000200  10   FORMAT (1X,'ENTER INPUT IN THE FORM AAB')
    0000300       READ (2,11)MAA,I
    0000400  11   FORMAT (I2,I1)
    0000500  20   K=1
    0000550       MSAVE=MAA
    0000600  30   DO 100 J=1,I
    0000700  40   K=K*J
    0000800  50          IF(J-1) 60,100,60
    0000850  60          MAA=MAA*(MSAVE-J+1)
    0000900  100  CONTINUE
    0001000  110  NUM=MAA/K
    0001100       WRITE (1,120)NUM
    0001200  120  FORMAT (I10)
    0001300       STOP
    0001400       END
    logoff
    0007         LOGOFF AT 14:43 ON 09/11/70
    TERMINAL LOGICALLY DISCONNECTED, RECONNECT OR HANG UP.
```

# 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



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"