intel®

# ISIS-IV USER'S POCKET REFERENCE



Order Number: 121890-001

## CONTENTS

## Notational Conventions

| | |
|---|---|
| UPPERCASE | Must be entered in either uppercase or lowercase in the order shown. |
| *italics* | Indicate variable information. |
| [ ] | Indicate optional arguments or parameters. |
| [ , . . . ] | Indicate that the preceding item may be repeated, but each repetition must be separated by a comma. |
| { } | Select one and only one of the enclosed entries. |
| { }... | Select at least one of the enclosed entries. |
| \| | Separates options within brackets [ ] or braces { }. |
| punctuation | Punctuation other than ellipses, braces, and brackets must be entered as shown. |
| ‹cr› | Indicates a carriage return. |
| file | :device: |

or

$$\left\{\begin{array}{l}\text{:F}n\text{:} \\ \textit{pathname}/\end{array}\right\} \textit{filename}$$

where

:F*n*: is the directory identifier that is assigned to the directory that contains *filename*. The value of *n* is an integer between 0 and 9 inclusive. If :F*n*: is not specified, :F0: is assumed.

*pathname* is a valid designation for a file; in its entirety, it consists of a *directory* and a *filename.*

*filename* is the name of a file that consists of 1-6 alphanumeric characters or wildcards* and an optional extension that consists of 1-3 alphanumeric characters or wildcards.

Wildcards:

An asterisk (*) matches any sequence of characters. A question mark (?) matches any single character.

## System Designated Device Names

:F0: thru :F9: directory identifiers
:BB: Byte Bucket
:BI: Batch input
:BO: Batch output
:CI: Console input
:CO: Console output
:LP: Line printer
:SP: Spool printer
:VI: Video terminal keyboard
:VO: Video terminal screen

## Control Characters

| | |
|---|---|
| RUBOUT | Deletes preceding character. Repeated usage is allowed. |
| CTRL-P | Allows literal entry of control characters (including itself). |
| CTRL-Q | Resumes console display after CTRL-S is entered. |
| CTRL-R | Redisplays current input line as modified. |
| CRTL-S | Stops console display and delays program execution. |
| CTRL-X | Deletes all characters since last carriage return. |
| CTRL-Z | Enters end-of-file. |

## Console Commands

### Program Execution Commands

***filename*** — execute the named program

*progname*[ *parameters*]

where

| | |
|---|---|
| *parameters* | are parameters needed by *progname.* |

**DEBUG** — execute a program under the Monitor

`DEBUG [[:F`*n*`:]`*filename*`]`

**EXIT** — terminate ISIS-IV execution

`EXIT`

**SUBMIT** — take console commands from a disk file

`SUBMIT [:F`*n*`:]`*filename*`[(`*parameter*`[,...])]`

## File Control Commands

**ACCESS —** list or change access rights of a file

```
ACCESS { :Fn:          }
       { pathname/     } filename[ switch]
```

where

*switch* is the owner access right.

**iNDX Access Rights**

| Identifier | Options |
|---|---|
| OWNER | O — Owner of the file or directory<br>W — World or public users |
| ACCESS | Data files:<br>R — Read a file<br>W — Write a file<br>D — Delete a file<br>Directory files:<br>L — List a directory<br>A — Add a directory entry<br>D — Delete a directory |
| RIGHT | 0 — Deny access right (reset, off)<br>1 — Grant access right (set, on) |

**ASSIGN —** assign a directory identifier

```
ASSIGN [{n, :Fn:} TO y]
```

where

*y* is a fully qualified directory pathname or the word NULL.

**ATTRIB —** list or change the write-protect status of a file

```
ATTRIB [:Fn:]filename [attriblist] [Q]
```

where

*attriblist* is W0 (resets the write-protect attribute) or W1 (sets).

Q specifies query mode operation.

**COPY** — copy a file

```
COPY [:Fn:]infile[,...]

TO {[:Fn:][outfile]} [switches]
   {:device:        }
```

where

| | |
|---|---|
| *infile* | is a file (or group of files when using the wild card construct) to be copied. |
| *outfile* | is a file to be created or recreated. |
| *switches* | are one or more of the following: |
| | Q specifies the query mode. |
| | B deletes an existing file without displaying ALREADY EXISTS prompt. |
| | U opens *outfile* for update instead of deleting it. |

**CREATE** — create a directory

```
CREATE {:Fn:      } new directory name
       {pathname/ }
```

where

| | |
|---|---|
| *new directory name* | is a string of up to 14 alphanumeric characters. |

**Default Directory Access Rights**

| Operation | Action |
|---|---|
| CREATE a directory on a 5¼ inch flexible diskette | GRANT:<br>World Delete Access<br>World List Access<br>World Add Entry Access |
| CREATE a directory on a Winchester device or hard disk | GRANT:<br>Owner Delete Access<br>Owner List Access<br>Owner Add Entry Access<br>DENY:<br>World Delete Access<br>World List Access<br>World Add Entry Access |

**DELETE** — delete a file

`DELETE [:Fn:]`*filename* `[Q] [,... [Q]]`

where

| | |
|---|---|
| Q | specifies the query mode. |

**DIR** — list a directory

`DIR [FOR` *filename*`] [TO` *listfile*`] [`*switches*`]`

where

| | |
|---|---|
| *listfile* | is the name of a file or output device where the directory listing will be displayed. |
| *switches* | are one or more of the following, separated by spaces: |
| | 0-9 lists the directory. If omitted, 0 is assumed. |
| | TE lists the extended directory. |
| | SP lists the contents of the network spooler print queue when workstation is connected to the network. |
| | O prints the directory in a single column format. |

**REMOVE** — delete a directory

`REMOVE` { `:Fn:` / *pathname/* } *directory name*

**RENAME** — rename a file

`RENAME [:Fn:]`*oldname* `TO [:Fn:]`*newname*

where

| | |
|---|---|
| *oldname* | is the name of an existing file to which the user has been granted delete access rights. |
| *newname* | is the new name to be assigned to *oldname*. |

**SPACE** — display the volume information of the specified file

```
SPACE /volume name
```

where

| | |
|---|---|
| *volume name* | is the volume root directory for the given physical device. |

**VERS** — display the ISIS utility program version numbers

```
VERS command
```

where

| | |
|---|---|
| *command* | is one of the ISIS command programs. |

**WHO** — display the name of the user

```
WHO
```

### Code Conversion Commands

**HEXOBJ** — convert hexadecimal code to absolute object code

```
HEXOBJ hexfile TO absfile [START (addr)]
```

where

| | |
|---|---|
| *hexfile* | is the file of machine object code in hexadecimal format. |
| *absfile* | is the output file from HEXOBJ containing the absolute object module that can be loaded for execution under ISIS-IV. |
| START (*addr*) | is used to include a starting address (the address of the first instruction to be executed) in the absolute object module. |

**OBJHEX** — convert ISIS absolute object code to hexadecimal code

```
OBJHEX absfile TO hexfile
```

## Program Control Commands

**LIB** — create and control program libraries

```
LIB
  CREATE libfile
  ADD sourcefile[(modname,...)]
   [,...] TO libfile
  DELETE libfile(modname,...)
  LIST libfile[(modname,...)]
  [,...][TO listfile]
   [PUBLICS]
  EXIT
```

**LINK** — combine program files and resolve external addressing

```
LINK inputlist TO outputfile[controls]
```

where

| | |
|---|---|
| *inputlist* | can be *filename*[(*modname*,...)],... or PUBLICS (*filename*,...),... |

Controls: MAP
NAME (*modname*)
PRINT (*filename*)

**LOCATE** — convert relocatable object to absolute addresses for execution

`LOCATE` *inputfile*[`TO` *outputfile*][*controls*]

where

*controls* are:

MAP
COLUMNS(*number*)
PRINT(*file*)
SYMBOLS
LINES
PUBLICS
PURGE
ORDER(*segment sequence*)
CODE(*address*)
DATA(*address*)
STACK(*address*)
MEMORY(*address*)
/*common name*/(*address*)
//(*address*)
RESTART0
START(*address*)
STACKSIZE(*value*)
NAME(*name*)

## System Calls

The ISIS-IV system calls can be called from a PL/M or Assembler language program. If the program makes an ISIS-IV system call, link the object program with SYSTEM.LIB using the LINK program.

Assembler Language Calls:

The interface between the Assembler language program and ISIS is accomplished by calling a single ISIS entry point (labeled ISIS) and passing two parameters:

Parameter 1: System Call Identifier (passed in register C)

| SYSTEM CALL | IDENTIFIER |
|---|---|
| OPEN | 0 |
| CLOSE | 1 |
| DELETE | 2 |
| READ | 3 |
| WRITE | 4 |
| SEEK | 5 |
| LOAD | 6 |
| RENAME | 7 |
| CONSOL | 8 |
| EXIT | 9 |
| ATTRIB | 10 |
| RESCAN | 11 |
| ERROR | 12 |
| WHOCON | 13 |
| SPATH | 14 |
| GETATT | 17 |
| GETD | 26 |

Parameter 2: address of control block containing additional parameters for the call (passed in register pair DE)

**NOTE**

*parameter*=pass VALUE of parameter
*parameter$p*=pass ADDRESS of parameter

Every parameter must be passed as a two-byte quantity.

## File Input/Output Calls

**CLOSE** — terminate input/output operations on a file

```
CALL CLOSE(conn, status$p)
```

**OPEN** — initialize file for input/output operations

```
CALL OPEN(conn$p, path$p, access, echo, status$p)
```

where

*access* is a value indicating the access mode for which the file is being opened.

1=read
2=write
3=read and write)

**READ** — transfer data from file to memory

```
CALL READ(conn, buf$p, count, actual$p, status$p)
```

**RESCAN** — position marker to beginning of line

```
CALL RESCAN(conn, status$p)
```

**SEEK** — position file marker

```
CALL SEEK(conn, mode, block$p, byte$p, status$p)
```

where

*mode* is a value from 0 through 4 that indicates what action should be performed on MARKER.

0=return to current position
1=move backward
2=move to specific position
3=move forward
4=move to end of file

**SPATH** — obtain file information

```
CALL SPATH(path$p, info$p, status$p)
```

**WRITE** — transfer data from memory to file

`CALL WRITE(`*conn*, *buf$p*, *count*, *status$p*`)`

## Directory Maintenance Calls

**ATTRIB** — change the write-protect status of a file

`CALL ATTRIB(`*path$p*, *atrb*, *onoff*, *status$p*`)`

where

*atrb* is a number indicating the attribute to be changed.

0=invisible attribute*
1=system attribute*
2=write-protect attribute (access right switches)
3=format attribute*
(*attributes that are valid for ISIS-II and III(N) only)

**DELETE** — delete a file from the directory

`CALL DELETE(`*path$p*, *status$p*`)`

**GETATT** — obtain write-protect information

`CALL GETATT(`*file$p*, *attrib$p*, *status$p*`)`

where

*attrib$p* is the address of a one-byte field to which the write-protect status of the file is to be returned.

bit 0=invisible*
bit 1=system*
bit 2=write-protect
bit 3, 4, 5, 6=reserved
bit 7=format*
(*valid for ISIS-II and III(N) only)

**GETD** — obtain file device directory information

`CALL GETD(`*did*, *conn$p*, *count*, *actual$p*, *table$p*, *status$p*`)`

**RENAME** — change a filename

```
CALL RENAME(old$p, newpath$p, status$p)
```

## Console Device Assignment and Error Message Output Calls

**CONSOL** — change console device

```
CALL CONSOL(ci$path$p, co$path$p, status$p)
```

**ERROR** — output error message on system console

```
CALL ERROR(errnum)
```

**WHOCON** — determine file assigned as system console

```
CALL WHOCON(conn, buf$p)
```

## Program Execution Calls

**EXIT** — terminate the program and return to ISIS-IV

```
CALL EXIT
```

**LOAD** — load a file of executable code and transfer control

```
CALL LOAD(path$p, load$offset, control$sw, entry$p, status$p)
```

where

*control$sw* is a value indicating where control is transferred after the load.

0=calling program
1=loaded program
2=Monitor

## Error Message Codes

### Resident Routine Error Message Codes (8080/8085 Mode)

1. Fatal error. Too few buffers were allocated.
2. Illegal active file table number.
3. Fatal error. Active file table is full.
4. Incorrectly specified filename.
5. Unrecognized device name.
6. Attempt to write to input device.
7. Fatal error. The device is full.
8. Attempt to read from output device.
9. Directory is full.
10. Pathname is not on same device.
11. File already exists.
12. File is already open.
13. No such file.
14. Write-protected file encountered.
15. Fatal error. ISIS overwrite.
16. Fatal error. Bad load format.
17. Not a device file.
18. Illegal ISIS commands.
19. Attempted seek on non-disk file.
20. Attempted back seek too far.
21. Cannot rescan.
22. Illegal access mode to open.
23. Missing filename.
24. Fatal error. Device input/output hardware error.
25. Illegal echo file.
26. Illegal attribute identifier.
27. Illegal seek command.

28. Missing extension.
29. Fatal error. Premature EOF.
30. Fatal error. Device not ready.
31. Cannot seek on write only file.
32. Cannot delete open file.
33. Fatal error. Illegal system call parameter.
34. Fatal error. Invalid return switch in a LOAD system call.
35. Seek past EOF.
61. Device not assigned.
62. Reserved.
63. Synchronization error.
64. Network Comm error.
65. Local Comm error.
66. Illegal attribute for iNDX file.
70. iNDX file access right violation.
71. Illegal operation on public file.
72. Maximum number of files on a device exceeded.
73. Attempt to delete a non-empty directory.
74. Illegal pathname syntax.
75. Non-terminating path element is not a directory.
76. Attempt to create a connected iNDX file.
77. Username/Password mismatch.
78. Username not known.
79. File error on system file.
80. iNDX file detached, device dismounted.
81. Maximum remote attaches exceeded.
82. Illegal password syntax.
83. Illegal username syntax.
88. Unknown remote error.

**NOTE**

When error 24 occurs, an additional message that reports the hexadecimal exception code number of the operating system, the device identifier, and the corresponding error message will appear.

`EXCEPTION` *xxxx*`H: ` *yyy*`ERR, ` *zzz*

where

*xxxx*H — is the hexadecimal number of the exception code of the operating system.

When the last digit of the exception code number is a 0 or a 1 (i.e., 3011H), the error message will be for a miniature flexible diskette.

If the last digit is a 2, 3, 4, or 5 (i.e., 3013H), the error will be for a hard disk.

If the last digit is a 6 (i.e., 3016H), the error message will be for a Winchester device.

*yyy* — is the device identifier (i.e., FIl identifies a miniature flexible diskette).

*zzz* — is the corresponding error message (i.e, ATTEMPT TO WRITE WITH HARDWARE WRITE PROTECT SET) that has the following meaning:

For a miniature flexible diskette if $n$=0 or 1:

```
301n    E$B$DELETED$DAM
302n    E$B$DATA$CRC
304n    E$B$ID$CYL$MIS
308n    E$B$SECTOR$BOUNDS
30An    E$B$ID$CRC
30En    E$B$NO$ID$AM
30Fn    E$B$BAD$DATA$AM
310n    E$B$DATA$OVERRUN
320n    E$B$WRITE$PROTECT
```

```
340n    E$B$DRIVE$FAULT
370n    E$B$NO$SECTOR
371n    E$B$BAD$TRACK
372n    E$B$SEEK$MIS
378n    E$B$UNEXPECTED
380n    E$B$DEVICE$NOT$READY
```

For a hard disk if *n*=2, 3, 4, or 5:

```
301n    ID field miscompare
302n    Data Field CRC
304n    Seek error
308n    Sector address out of bounds
30An    ID field CRC
30Bn    Protocol error
30Cn    Cyl addr out of bounds
30En    Sector not found
30Fn    No data field data mark
310n    Format overrun
320n    Write protected
340n    Drive write error
380n    Drive not ready
```

For a 35 megabyte Winchester if *n*=6:

```
302n    Data field ECC
304n    Drive Seek error
30An    ID field ECC
30En    Sector not found
311n    Controller RAM fail
312n    Controller ROM fail
313n    Seek in progress
314n    Track type disallows operation
315n    Beyond end of media
316n    Illegal sector size
317n    Controller diagnostic fault
318n    No index signal
319n    Invalid function code
31An    Invalid address
320n    Write protected
340n    Drive fault
372n    ID Cylinder address does not
        match seek
378n    Unexpected error code
380n    Drive not ready
```

```
3C00    E$B$BAD$COMMAND$CODE
3C10    E$B$BLOCK$OVERFLOW
3C20    E$B$72$INVALID
3C30    E$B$MINI$Q
3C40    E$B$8089
3C50    E$B$NO$CONTROLLER
3CF0    E$B$72$PROTOCOL

3D80    E$KEYBOARD$ABORT
3DC0    E$B$5440$PROTOCOL
3DC1    E$B$5440$Q
3DC2    E$B$5440$CONFIG
3DC3    E$B$UNKNOWN$INT
3DC4    E$B46UF$SEG$OVERFLOW
3DE0    E$B$SBIOS$ACCESS
3DF0    E$INSUFFICIENT$MIP$HEADERS
3E01    E$B$PRINTER$TIMEOUT
3E02    E$B$PRINTER$FAULT
3E04    E$B$PRINTER$NOT$READY
3E10    E$B$PRINTER$PROTOCOL
3F00    E$B$WINC$DATA$STRUC
3F01    E$B$WINC$PROTOCOL

4000    E$OPEN
4001    E$NOPEN
4002    E$FTYPE
4003    E$SYNTAX
4004    E$DEVICE$NOT$READY
4005    E$DEVICE$IO$ERROR
4006    E$COMM$ERROR
4007    E$NODE$NOT$READY
4008    E$MARKED$DELETED
4009    E$OPEN$MODE

4010    E$CONNECTIONS$EXIST
4011    E$DIR$NOT$EMPTY
4012    E$LIMIT
4013    E$EXIST
4014    E$CONSOLE
4015    E$LOG$NAME$EXISTS
4016    E$ILLEGAL$DEVICE$ID
4017    E$SYSTEM$DEVICE
4018    E$LOG$NAME$DOES$NOT$EXIST
4019    E$BAD$PATH
```

```
401A    E$NOT$DIRECTORY
401B    E$PATH$DOES$NOT$EXIST
401C    E$ATTACHED
401D    E$DETACHED
401E    E$NETWORK

4FFF    E$CONSISTENCY

8004    E$PARAM

CCB6    M$DISALLOWED$QUERY
CCB7    M$SINGLE$COMP
CCB8    M$DIR$EXIST
CCB9    M$DEL$CREATE$CONFLICT
CCBA    M$DELETE$ACCESS
CCBB    M$DIR$REQD
CCBC    M$DISPLAY$ACCESS
CCBD    M$ADD$ACCESS
CCBE    M$ILLEGAL$WILDCARD
CCBF    M$DISALLOWED$WILDCARD
CCC0    M$PARENT$NEXIST
CCC1    M$DATA$DIR$OPTIONS
CCC2    M$DIR$OPTION
CCC3    M$DATA$OPTION
CCC4    M$DIR$CREATE
CCC5    M$PASSWORD
CCC6    M$ILLEGAL$VALUE
CCC7    M$USER$ID
CCC8    M$USER$NAME
CCC9    M$NOT$COMPLETED
CCCA    M$CANNOT$CREATE
CCCB    M$READ$ACCESS
CCCC    M$COMMAND$SYNTAX
CCCD    M$SYSTEM$ERROR
CCCE    M$PATH$SYNTAX

DFFD    E$ISIS$WP$CONSISTENCY
DFFE    E$VNEXIST
DFFF    E$USER$SUPPORT

E000    E$SYN$SCAN$OVF
E001    E$SYN$PARSE$OVF
E002    E$SYN$REMOVE$OVF
E003    E$SYN$BUF$OVF
E004    E$SYN$VERSION
E005    E$SYN$CONSISTENCY
```

| | |
|---|---|
| E11A | Bad parameter to monitor routine |
| E119 | Unsupported monitor function was called |
| E11D | No IEU memory |
| E115 | Cannot load ISIS.LM |
| E111 | IEU hardware not responding |
| E10B | IEU board not responding (damaged software possible) |
| FFEB | E$PASSWORD$MISMATCH |
| FFEC | E$USER$UNKNOWN |
| FFED | E$USER$ID |
| FFEE | E$SUPER$USER |
| FFEF | E$FATAL$UDF$FNEXIST |
| FFF0 | E$FATAL$IO$ERROR |
| FFF1 | E$REBOOT$REQUIRED |
| FFF2 | E$MULTIPLE$ID |
| FFF3 | E$MULTIPLE$NAME |
| FFF4 | E$SYSTEM$FILE |
| FFF5 | E$IO |
| FFF9 | E$BOUNDS |
| FFFA | E$FILE$TOO$LONG |
| FFFB | E$DELETED |
| FFFC | E$END$OF$FILE |

## Non-Resident Routine Error Message Codes (8080/8085 Mode)

201. Unrecognized switch.

202. Unrecognized delimiter.

203. Invalid syntax.

206. Illegal disk label.

208. Checksum error.

209. Relocation file sequence error.

210. Insufficient memory.

211. Record too long.

212. Illegal relocation type.
213. Fixup bounds error.
214. Illegal SUBMIT parameter.
215. Argument too long.
216. Too many parameters.
217. Object record too short.
218. Illegal record format.
219. Phase error.
220. No EOF record in object module file.
221. Segment overflow during LINK operation.
222. Unrecognized record in object module file.
223. Fixup record pointer is incorrect.
224. Illegal record sequence in object module file in LINK.
225. Illegal module name specified.
226. Module name exceeds 31 characters.
227. Command syntax requires left parenthesis.
228. Command syntax requires right parenthesis.
229. Unrecognized control specified in command.
230. Duplicate symbol found.
231. File already exists.
232. Unrecognized command.
233. Command syntax requires a TO clause.
234. Filename illegally duplicated in command.
235. File specified in command is not a library file.
236. More than 249 common segments in input files.
237. Specified common segment not found in object file.
238. Illegal stack content record in object file.
239. No module header in input object file.
240. Program exceeds 64K bytes.

# Hexadecimal-Decimal Conversion

| Most Significant Byte | | | | Least Significant Byte | | | |
|---|---|---|---|---|---|---|---|
| Digit 4 | | Digit 3 | | Digit 2 | | Digit 1 | |
| HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 4 096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 8 192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 12 288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 16 384 | 4 | 1 024 | 4 | 64 | 4 | 4 |
| 5 | 20 480 | 5 | 1 280 | 5 | 80 | 5 | 5 |
| 6 | 24 576 | 6 | 1 536 | 6 | 96 | 6 | 6 |
| 7 | 28 672 | 7 | 1 792 | 7 | 112 | 7 | 7 |
| 8 | 32 768 | 8 | 2 048 | 8 | 128 | 8 | 8 |
| 9 | 36 864 | 9 | 2 304 | 9 | 144 | 9 | 9 |
| A | 40 960 | A | 2 560 | A | 160 | A | 10 |
| B | 45 056 | B | 2 816 | B | 176 | B | 11 |
| C | 49 152 | C | 3 072 | C | 192 | C | 12 |
| D | 53 248 | D | 3 328 | D | 208 | D | 13 |
| E | 57 344 | E | 3 548 | E | 224 | E | 14 |
| F | 61 440 | F | 3 840 | F | 240 | F | 15 |

# MON 85 (Monitor) Commands

## Program Execution Commands

**G** — execute command

G [ *start-address* ] [ , { *break-address* } ]

**N** — single step command

[ *count* ] N [ P ] [ *start-address* ] [ , ]

## Memory Control Commands

**C** — compare command

C *range*, *destination-address*

**D** — display command

[ *count* ] D [ { W } ] [ *range* ] [ , ]
[ { X } ]

**F** — find command

F *range*, *data*

**M** — move command

M *range*, *destination-address*

**S** — substitute command

[ *count* ] S [ W ] [ *start-address* ] [ = *expression* ]
[ *expression* ] . . . [ , ]

## Register Command

**X** — examine register command

X [ *register* { [ = *expression* ] | , } ]

## Utility Command

**P** — print value command

P [ { T / S } ] [ { *address* | *expression* | *literal* } [ , ] ... ]

## Monitor I/O Interface Routines

**CI** — console input routine

Reads a character entered at keyboard. Returned as byte value in PL/M or in A register in ASM.

**CO** — console output routine

Sends a single character to the console if ISIS-IV is operating in the foreground mode, or to the log file if ISIS-IV is operating in the background mode.

**IOCDR2** — keyboard interrupt control

Transmits an address value in BC to K.I.C. The B and C registers and CPU condition codes are affected. (Only those commands for enabling, disabling, and servicing keyboard interrupts that occur on level 6 of the Local Priority Interrupt Controller are accepted.)

## System Status Routines

**CSTS** — console status routine

00H — no key pressed
0FFH — key pressed returned as byte value in PL/M or in the C register in ASM

**MEMCK** — memory check

Value returned as address value in PL/M or in the H and L registers in ASM.

3065 Bowers Avenue, Santa Clara, California 95051
(408) 987-8080
Printed in U.S.A.