CP/M®

SUMMARY GUIDE

FOR VERSION 1.4 & 2.0

EDITED BY

BRUCE A. BRIGHAM

The Rosetta Stone

INTRODUCTION

Some of the information contained in this booklet was extracted from various manuals written by Digital Research, Compiler Systems Inc. and Microsoft. All paraphrased material contained in the CP/M® portion was used with permission of Digital Research, holder of the copyright. All paraphrased material contained in the CBASIC™ portion was used with permission of Compiler Systems Inc., holder of the copyright. The BASIC-80 portion is Copyrighted (C) 1979 Microsoft and reprinted with permission.

CBASIC-2 and Microsoft BASIC-80 are individually alphabetized by name which interlaces string functions, commands, arithmetic functions and so on. This kind of "total alphabetizing" saves time and eliminates the need for an index. It is not intended to be used as a learning guide and only has limited explanations and examples. If more detail is needed, the user should consult his/her user manual. We welcome any and all criticism.

This guide will be kept up-to-date as new software releases are received from the various producers.

COPYRIGHT NOTICE:

TABLE OF CONTENTS

File Name References 1
Disk References 1
Switching Disks 2
Logical Names 2
Physical Names 2
Common File types 2
Special Functions Keys 3

Direct Commands
- ERA erase files 4
- DIR directory list 4
- REN rename 4
- SAVE save program 4
- TYPE type file 4
- USER selects new user 4

Transient Commands
- ASM CP/M assembler 6
- DDT debugger 8
- DUMP 10
- ED editor 5
- LOAD 10
- MOVCPM 14
- PIP 11
- STAT 9
- SUBMIT 16
- SYSGEN 15
- XSUB 17

Commonly Used Programs
- DESPOOL 18
- MAC 19
- TEX 23
- CBASIC 25
  - Compiler time toggles 35
  - Compiler directives 35
  - Error messages 36
- MICROSOFT BASIC 80 Ver 5.0 43
  - Compiler commands 56
  - Compiler error messages 57
  - Edit mode subcommands 53
  - Operators 54
  - PRINT USING format 55
  - Special characters 53
  - Table of error codes 52

FILE NAME REFERENCES

Names are of the form PPPPPPPP.SSS
where PPPPPPPP is the primary name (1-8 chars).
where SSS is the secondary name (1 to 3 chars).

Unambiguous File Names (ufn)
file name completely specified.

Ambiguous File Names (afn)
parts of the file name are unspecified.

A question mark (?) in the file name will be replaced with any ASCII char. (except . , ; : = ? * [ ])

An asterick (*) is used to specify either the primary or secondary or both file names. (Sometimes called wild card)

File name examples:

SAMPLE.PRN
a file with primary name SAMPLE and a secondary name of PRN.

SAMPLE?.ASM
A file with many possible names SAMPLE1.ASM or SAMPLES.ASM are 2 valid names for this example of an ambiguous file name.

SORTME.*
A file named SORTME with any extension name (wild card option)

*.COM
All files with the secondary name of COM

.
All files are referenced.

DISK REFERENCES

All references to files on other drives are performed by simply typing the drive letter (A, B, C. . .P) followed by a colon followed by the file name.

e.g. B:SAMPLE.ASM
will reference the file named SAMPLE.ASM on drive B.

SWITCHING DISKS

Type the drive name desired followed by a colon (:)

TYPICAL COMMAND: A > B:
CP/M will respond with B> indicating the new drive = B.

LOGICAL NAMES

CON:	system CONsole
PTR:	Paper Tape Reader
PUN:	paper tape PUNch
LST:	LiST device (printer)

PHYSICAL DEVICE NAMES

TTY:	TeleTYpe device
CRT:	Cathode Ray Tube
BAT:	console is reader (RDR) output to current LST: device
UC1:	User defined Console #1
PTR:	high speed Paper Tape Reader
UR1:	User defined Reader #1
UR2:	User defined Reader #2
PTP:	high speed Paper Tape Punch
UP1:	User defined Punch #1
UP2:	User defined Punch #2
LPT:	Line Printer
UL1:	User defined List device
A:	disk A
B:	disk B
C:	disk C
D:	disk D for CP/M 2.0, up to 16 physical drives (P:)

COMMON FILE TYPES

.ASM	standard assembler source files
.BAK	backup file created by ED
.BAS	BASIC file
.COM	an executable memory image file
.DAT	DATA file
.DOC	messages or documentations
.FOR	FORTRAN type file
.HEX	an INTEL formatted HEX file (generated by ASM)
.INT	an INTERMEDIATE file created by EBASIC and CBASIC
.LIB	a LIBRARY file used by ED
.LST	LiST file created by CBASIC
.MSG	same as .DOC
.PRN	printout of text formatters, compilers, assembler. . .
.REL	relocatable files generated by MICROSOFT Basic-80
.SAV	a system file on CP/M 2.0
.SUB	source of SUBMIT utility
.SYM	symbol table, generated by macro assemblers
.TEX	ASCII data for Text-formatters
.TXT	same as .DOC
.$$$	a TEMPORARY file

SPECIAL FUNCTION KEY COMMANDS

rubout
Delete last character typed on console.

CTRL/B
Deactivates and removes DESPOOL (optional program).

CTRL/C
CP/M system reboot (warm boot).

CTRL/E
Physical end of line, carriage is returned but not sent.

CTRL/F
Actives DESPOOL (if purchased and installed).

CTRL/H
Backspace one (1) character position and delete that character. If the character was a TAB, it moves back to where it was (Ver 2.0 only).

CTRL/I
Tabs to next 8th column (8, 16, 24. . .).

CTRL/J
Line feed – same as carriage return, terminates current input (Ver 2.0 only).

CTRL/M
Carriage return – terminates input.

CTRL/P
Copy all console output to list device until next CTRL/P.

CTRL/R
Retype current command line.

CTRL/S
Stop console output, continue on any character.

CTRL/U
Delete the entire line typed at the console.

CTRL/X
CP/M 1.4 – same as CTRL/U.
CP/M 2.0 – backspace to beginning of current line.

CTRL/Z
End of input from console (ED & PIP).

DIRECT (BUILT-IN) COMMANDS

The (*) applies to all files in version 1.4 but only applies to the current USER in version 2.0. The following 6 commands are built-in or DIRECT:

DIR afn	request a list of all file names in the DIRectory which satisfy the ambiguous file name (afn)
ERA afn	ERAses all files whose name satisfies the afn (must have extension name)
TYPE afn	TYPE the file specified on the console
SAVE n ufn	SAVE n (decimal) pages (256 bytes ea.) on disk with name specified. To calculate n, take the high byte of the last address of the program in memory and convert it to decimal. (e.g. 0100H –> 2135H. . .21H = 33D therefore n = 33)
REN ufn1=ufn2	RENname existing file ufn2 with the first name specified (ufn1)
USER n	a new USER disk area will be established (Boot = USER 0, CP/M 2.0 only)
TRANSIENT	see transient commands. . .

---------EXAMPLES---------

DIR*.COM	lists all .COM type files at the console.
DIR B:	list all files contained on drive B.
ERA B:SAMPLE.*	erases all files with the primary name of SAMPLE from drive B.
TYPE SAMPLE.ASM	types the contents of file SAMPLE.ASM on the console.

SAVE 33 SAMPLE. CON

copies 0100H –> 21FFH to the file named SAMPLE.COM on the current drive.

REN SAMPLE1.ASM=SAMPLE2.ASM

file SAMPLE2.ASM is now named SAMPLE1.ASM.

USER 2	logs in user 2. (Ver 2.0)

TRANSIENT COMMANDS

ED (Editor) COMMANDS

nA	append lines
+–B	begin/bottom of buffer
+–nC	move n character positions
+–nD	delete n characters in current line
E	end edit and close files (normal exit)
nFs	find string "s"
H	end edit, close and reopen same files
I	insert characters before current line
nJ	place string in juxtaposition
+–nK	kill lines
+–nL	move down/up n lines
nM	macro definition
nNs	find next occurrence of string "s" (entire file)
O	return to original file
+–nP	move and print pages
Q	quit with no file change
R	read library file
nS	substitute strings
+–nT	type n lines (0 will type contents of current line)
+–U	translate lower to upper case if U no translation if –U
V	produces absolute line numbers (not stored on disk)
–V	turns off the absolute line numbers
OV	prints memory buffer statistics (free/total)
nW	write n lines to disk
nX	transfers n lines to temporary file
OX	erases the temporary file
nZ	sleep
+–n	move and type (+–LT)
n:T	move to line n and type that line (The T can be replaced with any ED command)
CR	Advance one line
–	Go back one line

TRANSIENT COMMANDS

ASM (Assembler)

INVOKING THE ASSEMBLER

The extension name is used to specify the source or destination of a file. The file type is .ASM.

TYPICAL COMMAND A > ASM filename.abc

where a = designates the disk name which contains the source file (valid char. = A, B, C. . .Y)

b = designates the disk name which will receive the .HEX file (valid char. = A, B, C. . .Z where Z = skip the generation of the .HEX file)

c = designates the disk name which will receive the .PRN file (valid char. = A, B, C. . .Z where X will send .PRN to the console and Z = skip .PRN)

--------EXAMPLES--------

ASM SAMPLE	Assemble SAMPLE.ASM on current drive and place SAMPLE.HEX and SAMPLE.PRN on current drive.
ASM SAMPLE.AZZ	Assemble SAMPLE.ASM on drive A and don't generate the .HEX or .PRN file (syntax checker).
ASM SAMPLE.ABX	Indicates that SAMPLE.ASM is to be taken from drive A, the .HEX file is to be sent to drive B and .PRN is sent to the console.

NUMERIC CONSTANTS

B	binary constant (base 2)
O	octal constant (base 8)
Q	octal constant (base 8)
D	decimal constant (base 10)
n.	where n is decimal number (e.g. 19.)
H	hexadecimal constant (base 16)

ASSEMBLER DIRECTIVES

ORG	set the program or data origin
END	end program, optional start address
EQU	numeric "equate"
SET	numeric "set"
IF	begin conditional assembly
ENDIF	end of conditional assembly
DB	define data bytes
DW	define data words
DS	define data storage area

ARITHMETIC AND LOGICAL OPERATORS

a+b	unsigned arithmetic sum of a and b
a–b	unsigned arithmetic difference between a and b
+b	unary plus (produces b)
–b	unary minus (identical to O–b)
a*b	unsigned magnitude multiplication of a and b
a/b	unsigned magnitude division of a by b
a MOD b	remainder after a/b
NOT b	logical inverse of b (O's become l's, l's become O's) where b is considered a 16-bit value
a AND b	bit-by-bit logical AND of a and b
a OR b	bit-by-bit logical OR a and b
a XOR b	bit-by-bit logical exclusive OR of a and b
a SHL b	the value which results from shifting a to the left by an amount b, with zero fill
a SHR b	the value which results from shifting a to the right by an amount b, with zero fill

ERROR MESSAGES

D	Data error: element in data statement cannot be placed in the specified data area.
E	Expression error: expression is ill-formed and cannot be computed at assembly time.
L	Label error: label cannot appear in this context (may be duplicate label).
N	Not implemented: features which will appear in future ASM versions (e.g. MACRO's) are recognized but flagged in this version.
O	Overflow: expression is too complicated (i.e., too many pending operators) to compute, simplify it.
P	Phase error: label does not have the same value on two subsequent passes through the program.
R	Register error: the value specified as a register is not compatible with the operation code.
U	Undefined label: label referenced doesn't exist.
V	Value error: operand encountered in expression is improperly formed.

TRANSIENT COMMANDS

DYNAMIC DEBUGGING TOOL (DDT)

As	"ASSEMBLE" mnemonic code starting at address s.
Ds,f	"DISPLAY" memory locations starting at s and ending at f.
Fs,f,c	"FILL" memory location starting at location s and ending at location f with data c.
Gs,b	"GO" and execute at location s and break at location b (b is optional).
Ifilename	"INSERT" the file named into the file control block (005CH) which will be read with the READ command.
Ls,f	"LIST" the assemble language mnemonics of the addresses specified, s is the start and f is the finish address.
Ms,d,f	"MOVE" data where s is the start address f is the final address and d is the destination.
Rb	"READ" the file specified by the INSERT command with a bias of b. (Bias optional and 0100H is assumed).
Ss	"SET" examine and optional alter any memory location s (a period "." terminates SET).
Tn	"TRACE" will display all registers and flags during the execution of a program for n steps.
U	"UNTRACE" is the same as TRACE but the registers are not displayed.
X	"EXAMINE" all registers.
Xr	"EXAMINE" register r, where r is any valid register name (A, B, C, D, E, H, I, L, M, P, S, Z).

ERRORS

"?" Means one of three (3) things:

1. The file can't be opened.
2. A checksum error occured in a HEX file.
3. The assembler/disassembler was overlayed.

RESTART DDT

by executing a RST 7 (FF).

RUBOUT

will abort long displays.

TRANSIENT COMMANDS

STAT command

STAT cr — calculates the storage remaining on all active drives.

STAT B: cr — selects drive B, then calculates the storage.

STAT SAMPLE.* cr — specifies a set of files to be scanned by STAT (all files with the primary name of SAMPLE).

STAT B: =R/O cr — sets drive B to read only until a warm or cold boot takes place.

STAT DEV: cr — produces a list of each physical device currently assigned to each logical unit.

STAT VAL: cr — prints the possible values which can be taken on for each logical device (CP/M 1.4).

STAT VAL: cr — produces a summary of the new 2.0 commands.

STAT USR: cr — produces a list of USER numbers which have files on the currently addressed disk.

STAT SAMPLE. COM $S cr — produces the output shown below ($S is optional).

Size	Recs	Bytes	Ext	Acc	
48	48	6K	1	R/O	A:SAMPLE.COM

STAT B: DSK: — lists the drive characteristics of the disk B. B: can be in the range of A: → P: (16 drives)

STAT SAMPLE.PRN $R/O cr — sets the file SAMPLE.PRN to read only (CP/M 2.0).

STAT SAMPLE. PRN $R/W cr — sets the file back to read/write (CP/M 2.0).

STAT B:PIP.COM $SYS cr — makes PIP a system file – will not be displayed when DIR is issued (CP/M 2.0).

STAT B:PIP.COM $DIR cr — makes PIP a system file – opposite of $SYS (2.0).

TRANSIENT COMMANDS

REASSIGNMENT OF THE LOGICAL DEVICES

STAT logical=physical,. . .logical=physical cr

--------EXAMPLES--------

STAT CON: = CRT: cr
STAT PUN: = TTY: ,LST:=LPT: cr

LOAD command

The LOAD command reads the file specified which is assumed to be in INTEL "hex" format and produces a memory image file which can be subsequently executed.

Typical Command:

LOAD SAMPLE	this will produce a file by the name SAMPLE.COM from the file named SAMPLE.HEX.
LOAD B:SAMPLE2	this will load the .HEX file from drive B and produce the corresponding .COM file.

In all cases, the file created is ready to be executed by simply typing its name. The file is now considered a transient program.

e.g. A>SAMPLE cr will run the program SAMPLE.

DUMP command

This DUMP program is used to display the contents of a binary disk file at the console in hexadecimal form. Of course this can be sent to the printer by typing (CTRL P) before the carriage return.

Typical Command:

DUMP SAMPLE.COM will produce a hexadecimal table of that file.

NOTE: DDT produces a nicer table due to the fact that the ASCII codes for the memory contents are displayed to the right of the HEX code.

TRANSIENT COMMANDS

PIP command

INVOKING PIP

1. type PIP cr
2. type PIP "command line" cr

COMMAND LINE FORM

destination = source#1, source#2, . . ., source#3 cr

DISK REFERENCES

file names prefixed by a A: or B: . . . P: will be accessed to/from that drive.

DISK TO DISK COPY EXAMPLES

PIP B:SAMPLE.* cr

copies all files from the current drive to drive B with the primary name of SAMPLE.

PIP B:=C:SAMPLE.* cr

copies all files with the primary name SAMPLE from drive C to drive B.

PIP cr
*SAMPLE.ASM=B: cr

invokes PIP, and then takes the file SAMPLE.ASM from drive B and copies it to the current drive.

PIP A:SAMPLE.ASM=B: cr

copies the file SAMPLE.ASM from drive B to drive A.

PHYSICAL/LOGICAL DEVICE REFERENCE

PIP can directly address any of the logical or physical devices already discussed. They are listed again for convenience.

LOGICAL	PHYSICAL:			
CON: =	TTY:	CRT:		UC1:
RDR: =	TTY:	PTR:	UR1:	UR2:
PUN: =	TTY:	PTP:	UP1	UP2:
LST: =	TTY:	CRT:	LPT:	UL1:

Additional device names which can be used in PIP:

NUL:	sends 40 nulls to the device specified.
EOF:	sends end-of-file (CTRL–Z) to device.
INP:	special PIP input source which can be "patched" into the PIP program itself. Check page 21 of the CP/M Features and Facilities manual.
OUT:	special PIP output handler patched into PIP (similar to INP:).
PRN:	Same as LST:, except that the tabs are expanded at every eighth character position, lines are numbered and page ejects are inserted every 60 lines with an initial eject (same as [t8np]).

OPTIONAL TRAILING PIP PARAMETERS:

B	Block mode transfer: data is buffered by PIP until an ASCII x-off (ctrl–s) is received from the source device.
Dn	delete characters which extend past column n during the transfer of data to the destination from the character source (truncates long lines).
E	Echo all data to the console as it is happening.
F	Filter form feeds (ctrl–L) from the file.
Gn	get file from USER n (CP/M 2.0 n=Ø to 15).
H	Hex data transfer: checks and corrects for proper INTEL hex format (console is prompted).
I	Ignore ":OO" records in the transfer of Intel hex format file (automatically sets H parameter).
L	Translate upper case alphabetics to lower case.
N	Add line numbers to each line transferred starting at 1, incrementing by 1 (no leading zeros). If N2 is used, leading zeros are included.
O	Object file transfer (non-ASCII) the normal CP/M end of file is ignored.
Pn	Include page ejects at every n lines (with an intial page eject). If n = 1 or is excluded altogether, page ejects occur every 60 lines. If the F parameter is used, form feed suppression takes place before the new page ejects are inserted.

TRANSIENT COMMANDS

Qs^z Quit copying from the source device or file when the string s (terminated by ctl–Z) is encountered.

R read system files (CP/M 2.0).

Ss^z Start copying from the source device when the string s is encountered (terminated by ctl–z). The S and Q parameters can be used to "abstract" a particular section of a file (such as a subroutine). The start and quite strings are always included in the copy operation.

Tn Expand tabs (ctl–I characters) to every nth column during the transfer of characters to the destination from the source.

U Translate lower case alphabetics to upper case during the copy operation.

V Verify that data has been copied correctly by rereading after the write operation (the destination must be a disk file).

W write over R/O files without console interrogation.

Z Zero the parity bit on input for each ASCII character.

PIP OPTION EXAMPLES

PIP B:SAMPLE.ASM=A:SAMPLE.BAK[UVN] cr

this will copy the backup file SAMPLE.BAK from drive A to drive B, changing its name to SAMPLE.ASM, translates lower case to upper case, number each line and verify a good copy.

PIP cr

*SUBR.ASM=TEST.ASM[SSTART: ^ZQEND^Z]

copy from file TEST.ASM to file SUBR.ASM of the current drive, all text starting from the string START: and ending after the first occurrance of the string END (extracts part of a file).

PIP LST:=SAMPLE.PRN[t8np] cr

print on the LIST device the file SAMPLE.PRN expanding the TABs to 8 columns, number the lines and page eject every 60 lines.

TRANSIENT COMMANDS

MOVCPM

The MOVCPM allows the user to reconfigure the CP/M system for any memory size (1K boundaries). Two (2) parameters are optionally given after the command MOVCPM. The first is used indicate the memory size desired and the second is used indicate the disposition of the newly created system when MOVCPM terminates (remain in memory for SYSGEN or execute). The following examples will more clearly show these options:

MOVCPM cr	relocate and execute CP/M for management of the current memory configuration. Upon completion of the relocation, the new system is executed but not recorded on disk.
MOVCPM 32 cr	create a relocated CP/M system for management of a 32K system (range = 16–64) and execute the system as in example 1.
MOVCPM * * cr	construct a relocated memory image for the current memory configuration but leave the memory image in memory for preparation of a SYSGEN.
MOVCPM 48 * cr	construct a relocated memory image for a 48K memory system and leave the memory image in memory for preparation of a SYSGEN. (See SYSGEN for a complete example.)

SYSGEN

The SYSGEN program enables the user to copy or create a new operating system from the old disk to a new one. The prompting messages are self-explanatory but examples are shown below for clarity.

This example is a simple backup procedures that assumes the user wants to copy the operating system on drive A to drive B. The user's response is underlined.

```
A > SYSGEN cr
SYSGEN VERSION 1.4
SOURCE DRIVE NAME (OR RETURN TO SKIP) A
SOURCE ON A, THEN TYPE RETURN cr
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION ON B, THEN TYPE RETURN cr
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) cr
A>
```

In the next example, if a "MOVCPM 48 *" was previously executed, the SOURCE DRIVE NAME (OR RETURN TO SKIP) would be skipped (type a cr) to allow the newly created system in memory to become the SOURCE. The DESTINATION DRIVE NAME would be the same as before.

e.g.

```
A > MOVCPM 48 *

CONSTRUCTING 48K CP/M VERS 1.4
READY FOR "SYSGEN" OR
"SAVE 32 CPM48.COM"
A > SYSGEN cr
SOURCE DRIVE NAME (OR RETURN TO SKIP) cr
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION ON B, THEN TYPE RETURN cr
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) cr
A>
```

The system that was just moved is ready to run if it is a standard CP/M system but if any customizing is needed the user should consult his/her owner's manual.

TRANSIENT COMMANDS

SUBMIT ufn parm#1 . . . parm#n

The SUBMIT command allows CP/M commands to be batched together for automatic processing. The ufn given in the SUBMIT command must be the filename of a file which exists on the currently logged drive with an assumed file type of "SUB." The SUB file contains CP/M prototype commands with possible parameter substitution.

The prototype command file is created using the ED program. The dollar sign ($) is used to represent the parameters which are in the form: $1 $2 $3. . .

--------EXAMPLE--------

Suppose the file ASSEMBLE.SUB exists on the currently logged disk and contains the following prototype commands:

```
ASM $1
DIR $1.*
TYPE $1.PRN
ERA $1.PRN
LOAD $1.HEX
REN $2.COM=$1.COM
```

and the command

```
A > SUBMIT ASSEMBLE SAMPLE GOOD cr
```

is issued by the operator. The SUBMIT program reads the ASSEMBLE.SUB file, substituting SAMPLE for all occurrences of $1 and GOOD for all occurrences of $2 resulting in a file of $$$.SUB with the following substitutions:

```
ASM SAMPLE
DIR SAMPLE.*
TYPE SAMPLE.PRN
ERA SAMPLE.PRN
LOAD SAMPLE.HEX
REN GOOD.COM=SAMPLE.COM
```

This file is then executed as if the user typed each command and waited for each command to finish. This is a real time saver if the same procedure is repeated many times.

XSUB*

XSUB extends the capability of SUBMIT to include line input to programs as well as the console command processor. The XSUB command is included as the first line of your SUBMIT file and when executed, self-relocates directly below the CCP.

If the SUBMIT file SAVER.SUB contained:

```
XSUB
DDT
I$1.HEX
R
GØ
SAVE 1 $2.COM
```

and the command:

```
A > SUBMIT SAVER SAMPLE DONE
```

was typed, the following would take place:

1. XSUB program loads
2. DDT loads right after
3. DDT executes
   ISAMPLE.HEX
   R
   GØ
4. SAVE 1 DONE.COM

XSUB stays active until a cold boot is performed. Because DESPOOL lives and hides in the same area, it must be loaded before XSUB.

*Used with permission of Digital Reserach, holder of the copyright.

DESPOOL™

This program is optional and can be purchased from Digital Research. It allows a file to be typed on the list device while the console device is being used. To install DESPOOL, just type it's name. (It must be on the current drive). It announces itself and disappears.

Typical Command:

A > DESPOOL cr

There are 2 commands the user has:

CNTL F	starts the printing
CNTL B	stops the printing and removes itself

DESPOOL™ is a registered trademark of Digital Research.

ARITHMETIC AND LOGICAL OPERATORS:

a+b	arithmetic sum of a and b
a–b	arithmetic difference between a and b
a*b	unsigned magnitude division of a by b
a/b	unsigned magnitude division of a by b
a MOD b	remainder after division of a by b
a SHL b	a shifted left by b, with zero right fill
a SHR b	a shifted right by b, with zero left fill
NOT b	bit-by-bit logical inverse of b
a EQ b	produces true if a equals b, false otherwise
a LT b	produces true if a is less than b
a LE b	produces true if a is less or equal to be
a GT b	produces true if a is greater than b
a GE b	produces true if a is greater or equal to be
a AND b	produces the bitwise logical AND of a and b
a OR b	produces the bitwise logical OR of a and b
a XOR b	produces the logical exclusive OR of a and b
HIGH b	is identical to b SHR 8 (high order byte of b)
LOW b	is identical to b AND OFFH (low order byte of b)

ASSEMBLER DIRECTIVES

ORG	sets the program or data origin
END	terminates the physical program
EQU	performs a numeric "equate"
SET	performs a numeric "set" or assignment
IF	begins conditional assembly
ELSE	is an alternate to a previous IF
ENDIF	marks the end of conditional assembly
DB	defines data bytes or strings of data
DW	defines words of storage (double bytes)
DS	reserves uninitialized storage areas
PAGE	defines the listing page size for output
TITLE	enables pages titles and options

INLINE MACROS

REPT–ENDM Repeat the statements between the REP and the ENDM to the value of expression.

```
label:  REPT expression
        statement–1
        statement–2
        ....
        statement–n
label:  ENDM
```

MAC™ is a trademark of Digital Research.

MAC™ Cont'd.

IRPC–ENDM – causes the assembler to re-read a bounded set of statements and replace identifier with each character in the list.

```
label:   IRPC identifier, character-list
         statement–1
         statement–2
         ....
         statement–n
label:   ENDM
```

The "identifier" is any valid assembler name, not including embedded "$" separators, and "character-list" denotes a string of characters, terminated by a delimiter (space, tab, end-of-line, or comment).

IRP–ENDM is similar in function to the IRPC, except that the controlling identifier can take on a multiple character value.

```
label:   IRP identifier,   cl–1, cl–2, . . ., cl–n
         statement–1
         statement–2
         ....
         statement–m
label:   ENDM
```

On the first iteration, the character-list given by "cl–1" is substituted for the identifier wherever the identifier occurs in the bounded statement group (statements 1 through m). On the second iteration, cl–2 becomes the value of the controlling identifier. Iteration continues in this manner until the last character-list is encountered and processed.

EXITM – can occur within the body of a macro and, upon encountering the EXITM statement, the macro assembler aborts expansion of the current macro level.

```
         macro-heading
         statement–1
         ...
label:   EXITM
         ...
         statement–n
         ENDM
```

The EXITM statement normally occurs within the scope of a surrounding conditional assembly operation.

MAC™ is a trademark of Digital Research.

MAC™ Cont'd.

LOCAL – it is often useful to "generate" labels for jumps or data references which are unique on each repetition of a macro. This facility is the opposite of GLOBALS used in other assemblers.

```
        macro-heading
label:  LOCAL    id–1, id–2, . . ., id–n
        . . .
        ENDM
```

"Macro-heading" is a REPT, IRPC, or IRP heading and id–1 through id–n represent one or more assembly language identifers which do not contain embedded "$" separators. The LOCAL statement must occur within the body of a macro definition.

DEFINITION AND EVALUATION OF STORED MACROS

MACRO-ENDM

The prototype statements for a stored macro are given in the macro body enclosed by the MACRO and ENDM pseudo operations, taking the general form.

```
macname MACRO      d–1, d–2, . . ., d–n
        statement–1
        statement–2
        . . .
        statement–3
        statement–m
label:  ENDM
```

Where the "macname" is assembly language identifier, d–1 through d–n constitutes a (possibly empty) list of assembly identifers without imbedded "$" separators and statements–1 through m are the macro prototype statement.

MAC™ is a trademark of Digital Research.

ASSEMBLY PARAMETERS

In general, the macro assembler is initiated with the name of the source file, followed by the assembly parameters, indicated by a preceding dollar symbol ($).

A	controls the source disk for the .ASM file.
H	controls the destination of the .HEX machine code file.
L	controls the source disk for the .LIB files (see MACLIB).
M	controls MACRO listings in the .PRN file.
P	controls the destination of the .PRN file containing the listing.
Q	controls the listing of LOCAL symbols.
S	controls the generation and destination of the .SYM file.
1	controls pass 1 listing.

The A, H, L, and S parameters are followed by the drive name to obtain or receive the data.

Example: MAC SAMPLE $PB AA HB SX

which directs the .PRN file to disk B, reads the .ASM file from disk A, directs the .HEX file to the B disk, and sends the .SYM file to the user's console. Remember X = console Parameters L, S, M, Q, and 1 can be preceded by either + or −.

+L	list the input lines read from the macro library.
−L	suppress listing of the macro library.
+S	append the .SYM to the end of the .PRN output.
−S	suppress the generation of the symbol table.
+M	list all macro lines as they are processed.
−M	suppress all macro lines as they are read.
+M	list only "HEX" generated by macro expansions.
+Q	suppress all LOCAL symbols in the symbol list.
+1	produce a listing file on the first pass.
−1	suppress listing on pass 1 (default).

e.g. MAC SAMPLE $PX+S−M

assembles the file SAMPLE.ASM with listing to console, symbols at console and no listing of generated macros.

MAC™ is a trademark of Digital Research.

TEX COMMAND SUMMARY

.AD	Adjust Margins (initially on)
.BP +–n	Begin Page
.BR	Break
.CE n	Conditional Page
.DS	Double Space (init. off)
.HE s	Heading (init. off)
.HM +–n	Heading Margin (init. 2 lines)
.IG	Ignore
.IN +–n	Indent (init. O)
.LI	Literal
.LL +–n	Line Length (init. 70)
.LS n	Line Spacing (init. 1)
.MB +–n	Margin, Bottom (init. 5)
.MT +–n	Margin, Top (init. 6)
.NA	No Admustment (init. off)
.OP	Omit Page Numbers (init. off)
.PA n	Page Advance
.PL +–n	Page Length (init. 66)
.PN +–n	Page Number (init. 1)
.PO +–n	Page Offset (init. 8)
.PP n	Paragraph (init. 6)
.QI	Quit Indentation
.SP n	Space n Lines (init. 1)
.SS	Single Space (init. on)
.TI +–n	Temporary Indent (init. O)

RUN-TIME PARAMETERS

$C	Redirects output to print on console device.
$E	Redirects error messages to print on list device.
$F	Enables user to forms-Feed on printer, if capability exists.
$L	Redirects output to print on list device.
$P	Stops printing at end of page, for paper change; resumes printing with <cr> .
$S	Produces no output file; prints error messages as usual.

TYPICAL TEX COMMAND:

TEX SAMPLE $L F

will send output to LIST device and enables forms-feed if LIST device has capability.

TEX SAMPLE $P $E

will stop printing at end of page and prints all error messages on LIST.

TEX ERROR MESSAGE:

OPENING SOURCE*	Invalid or missing source file name.
PARAMETER SCAN*	Invalid character in parameter string.
FILE READ*	Invalid file format.
DISK WRITE*	Cannot write output file from memory to disk.
READING EOF*	End-of-file mark detected where not expected.
FITTING A WORD	Usually, a long word cannot be formatted successfully into a short line length.
COMMAND VERIFY	Invalid command after a "." at beginning of input line.
HM COMMAND	Heading margin value greater than or equal to the top margin.
MB COMMAND	Bottom margin greater than or equal to PL value – MT value.
MT COMMAND	Top margin greater than or equal to PL value – MB value.
PP COMMAND	Paragraph indent value greater than line length.

ABS(X)
returns the absolute value of the X. (e.g. Y = ABS (X))

ASC(A$)
returns the ASCII numeric value of the first character of the A$. (e.g. Y%=ASC(A$))

ATN(X)
returns the arctangent of the X. (e.g. Y=ATN(X))

CALL <exp>
used to link to a machine language subroutine. The exp is the address of the subroutine. Return to CBASIC by executing a 8080 RET instruction. (e.g. CALL 5)

CHAIN <exp>
transfers control from the program currently being executed to the program selected by the expression. (e.g. CHAIN "B:PAYROLL.COM")

CHR$(I%)
returns a character string of length 1 consisting of the character whose ASCII equivalent I%. (e.g. PRINT CHR$(BELL%))

CLOSE <exp> , <exp>
closes the file specified by each exp . (e.g. CLOSE 1,2,3)

COMMAND$
returns a string which contains the CP/M command line. (e.g. CRUN2 PAYROLL NOCHECKS TOTAL) will return: NOCHECKS TOTAL

COMMON <var> , <var>
specifies that the variables listed will be common to the main program and all programs executed through a CHAIN statement. COMMON statements must be the first statements in a program (if used).

CONSOLE
restores printed output to console (see LPRINTER). (e.g. CONSOLE)

CONCHAR%
reads one (1) character from the keyboard (integer returned) (e.g. I%=CONCHAR% or IF CONSTAT% THEN CHAR$ = CONCHAR%)

CONSTAT%
returns the console status as an integer value. If the console is ready, a logic true is returned. (e.g. IF CONSTAT% THEN GOSUB 100)

COS(X)
returns the cosine of the X. (e.g. Y=COS(B))

CREATE <exp> | RECL <exp> | AS <exp> | BUFF <exp> RECS <exp> |
is the same as OPEN except that a new file is created. (e.g. CREATE TEMP$ RECL 20 AS 6)

DATA <constant> { , <constant> }
DATA statements define string and floating point constants which are assigned to variables using a READ statement. (e.g. DATA 10.0, "word", 100)

DEF <function name> [(<dummy argument list >)] = <exp> specifies a user defined function which returns a value of the same type as the <function name> . (e.g. DEF FNA(X,Y) = X+Y−Z)

DEF <function name> [(<dummy arg list >)]
this is the multiple line function where dummy list is the same as the single line function. The DEF must end with FEND. Explain by example:

```
DEF FN.OMEGA(PI,f)
        W=2*PI*f
        PRINT W
        RETURN
FEND
```

DELETE <exp> { , <exp> }
erases the active file referenced by <exp> . (e.g. DELETE 6,7)

DIM <ident> (<subscript list>) { , <ident> (<subscript list>) }
dynamically allocates space for numeric or string arrays. (e.g. DIM B$ (2,5,10),C(1 + 7.3,N),D%(1)).

END
indicates the end of the source program. (e.g. 10 END)

EXP(X)
returns the value of "e" raised to the power of X. (e.g. Y=10*EXP(2))

FEND
used to end a multiple line DEF instruction. See DEF.

FILE <var> [(<exp>)] , <var> [(<var>)]
opens files used by the program. The order of the names determines the numbers used to reference the files in READ and PRINT statements. (e.g. FILE INPUT$, OUTPUT$)

FOR <index> = <exp> TO <exp> [(STEP <exp>)]
execution of all statements between the FOR statement and its corresponding NEXT statement is repeated until the indexing variable reaches the exit criteria. (e.g. FOR I=1 TO 10 STEP 3)

FLOAT(I%)
converts the argument I% into a real value. (DOLLAR = FLOAT(COST%)

FN <ident>
FN refers to a user-defined function. See DEF (e.g. FNA(A,B))

FRE
returns the number of bytes of unused space in the free storage area. (e.g. Y=FRE)

GOSUB or GO SUB <line number>
control is transferred to the subroutine labeled with the <line number>. (e.g. GO SUB 100)

GOTO or GO TO <line number>
execution continues at the statement labeled with the <line number> . (e.g. GO TO 10).

IF <exp> THEN <line number>
(e.g. IF Y=X THEN 300)

IF <exp> THEN <statement list> [ELSE <statement list>]
operates the same as any other IF instruction. (e.g. IF X=A+B THEN PRINT MSG$ ELSE PRINT "too big")

IF END # <exp> THEN <line number>
if end of file is detected control is transferred to the line number. (e.g. IF END # 1 THEN 100)

INITIALIZE
initializes a new diskette after a disk is replaced. Similar to a ^ C when under the operating system. Equivalent to the CALL 264 in CBASIC 1.0. (e.g. INITIALIZE)

INP (<exp>)
performs an input operation on the 8080 machine port represented by the value of the <exp> . (e.g Y=INP(2)).

INPUT [<prompt string> ;] <variable> { , <variable> }
prints prompting string and a line of input data is read from the console and assigned to the variables as they appear in the variable list. (e.g. INPUT "SIZE OF ARRAY?"; N).

INPUT [<prompt string> ;] LINE <var>
same as INPUT except only one variable is permitted and it must be a string. (e.g. INPUT "NAME & AGE"; NAME$, AGE%)

INT(X)
returns the integer part of the variable. (e.g. Y=INT(X))

INT%(X)
converts X into an integer. See FLOAT% (e.g. J% = INT%(DOLLAR)

LEFT$(A$,I%)
returns a string consisting of the first I% characters of A$. (e.g. B$= LEFT$ (A$, 3).

LEN(A$)
returns the length of the string A$. (e.g. Y=LEN(C$ + B$)).

LET <variable> = <exp>
the <exp> is evaluated and assigned to the <variable> appearing on the left side of the equal sign. (e.g. LET A=B+C)

LOG(X)
returns the natural logarithm of the absolute value of the X. (e.g. Y= LOG (X))

LPRINTER [width <exp>]
directs all output in a PRINT statement to the printer which is set to an optional WIDTH at execution time (see CONSOLE). (e.g. LPRINTER WIDTH = 120)

MATCH(pattern$, object$, start%)
returns the position of the first occurance of "pattern$" in "object$" starting with character position "start%".

1. # will match any digit (0-9)
2. ! will match any upper/lower letter (a-z)
3. ? will match any character
4. # ! ? will match themselves if immediately preceded by a /. (e.g. MATCH "IT", "HE HIT IT", 1) returns 5.

MID$ (A$, I%, J%)
returns a string consisting of the J% characters of A$ starting at the I% character. (e.g. B$=MID$(OBJECT$,LOC%,3)

NEXT [<ident> { ,<ident> }]
a NEXT statement denotes the end of the closest unmatched FOR statement. (e.g. NEXT I)

ON <exp> GOTO <line number> { , <line number> }
The <exp> , rounded to the nearest integer value is used to select the <line number> at which execution will continue. If the<exp> evaluates to 1, the first <line number> is selected and so forth. (e.g. ON I GOTO 10,20,30,40)

OPEN<exp>[RECL<exp>] AS<exp>[BUFF<exp>REC <exp>]
opens files on the system disks where<exp>is the filename (w/ optional drive). The RECL is an optional fixed record length. The AS assigns an identification number to the file. BUFF specifies the number of disk sectors to maintain in memory at one time (default=1). RECS specifies the size of the disk sector (not used at this time but REC must follow BUFF <exp>. (e.g. OPEN "ACCOUNT.DAT" AS 9 BUFF 6 REC 128) (e.g. OPEN mail.list$ AS 10)

OUT<exp>,<exp>
the low-order eight bits of the integer portion of the second <exp> is sent to the 8080 machine output port selected by the integer portion of the first expression modulo 256. (e.g. OUT 3,10)

PEEK (<exp>)
returns the contents of the memory location given by <exp> (e.g. Y=PEEK(256))

POKE <exp1>,<exp2>
the low-order 8 bits of<exp2> is stored at the memory address pointed to by <exp1> . (e.g. POKE 255,ASC("?"))

POS
returns the current position of the output line buffer pointer. Either the console or the printer. (e.g. PRINT TAB(POS + 3);X)

CBASIC™ is a trademark of Complier Systems.

PRINT # <exp> , <exp> ; <exp> { , <exp>} (1) RANDOM FILE
PRINT # <exp> ; <exp> { , <exp> } (2) SEQUENTIAL FILE
PRINT <exp> <delim> { <exp> <delim>} (3) CONSOLE

a PRINT statement sends the value of the expressions in the expression list to either a disk file (type (1) and (2)) or the console (type (3)).

delim can be a ";" (skip 1 space) or a"," (tab to next 20th tab stop)

(e.g. PRINT #FILE,HERE; A,B,C,D$)
(e.g. PRINT #2;A,B,C$,D)
(e.g. PRINT "THIS IS TEST NO. ",N)

PRINT USING <format string> ; [<file reference>] <exp>

The "format string" is composed of data fields and literal data. The "format string" is any string expression. The <exp> consists of expressions separated by commas or semicolons. The "file reference" is optional and will direct the output to the disk file. There are 4 field types which are shown below as examples:

1. String character field (! prints only one character)
   e.g. NAME$="John"
   PRINT USING "!.";NAME$,"SMITH"
   will print J.S.

2. Fixed length string fields (/ /)
   e.g. PART$="FILE NO. IS/. . . . ./":N=333
   PRINT USING PART$; N
   will print FILE NO. IS 333

3. Variable length string fields (specified by &)
   e.g. MONEY$ = "ONE DOLLAR"
   PRINT USING "& &"; "GIVE HIM",MONEY$
   will print GIVE HIM ONE DOLLAR

4. Numeric data string fields (specified by #)
   e.g. PRINT USING "##,### ";100,1000,10000
   will print 100 1,000 10,000

e.g. COST = 123456.78
PRINT USING "**##,###,###.## "; COST
will print *123,456.78

PRINT USING "$$##,###,###.## "; COST
will print $123,456.78

PRINT USING "###- ### -"; 10, 10, -10, -10
will print 10 100E-01 10- 100E-01-

PRINT USING "-##### "; 10, -10
will print 10 - 10

CBASIC ™ is a trademark of Compiler Systems, Inc.

PRINT USING <exp> ; # <exp> ; <exp> , <exp>
PRINT USING <exp> ; # <exp> , <exp> ; <exp> { , <exp> }

these are used to write formatted data to a disk file. They follow the same rules as the PRINT USING above.

(e.g. B$="BROWN": COSTS$="$$#,###.##": PRICE=1234.56
PRINT USING "THE QUICK "&" FOX " +COST$; #2; B$, PRICE
will print THE QUICK BROWN FOX COST $1,234.56)

RANDOMIZE

a RANDOMIZE statement initializes the random number generator. (e.g. RANDOMIZE)

READ <var> { , <var> } (1) DATA
READ # <exp> , <exp> ; <var> , <var> (2) RANDOM FILE
READ # <exp> ; <var> { , <var> } (3) SEQUENTIAL FILE

a type (1) READ statement assigns values from DATA statements to the variables in the list. Type (2) reads a random record specified by the second expression from the disk file specified by the first expression and assigns the fields in the record to the variables in the variable list. The type (3) READ statement reads the next sequential record from the file specified by the expression.

(e.g. 100 READ A,B,C$)
(e.g. 200 READ #1, I; PAY.REG)
(e.g. READ #2; NAME$,NO,ZIP)

READ # <exp> ; LINE <var>

reads sequentially all the data from the specified file until a carriage return is encountered. All data is assigned to <var> . (e.g. READ #6; LINE data$)

READ # <exp1> , <exp2> ; LINE <var>

reads record specified by <exp 2> of the file specified by <exp1> . All data is assigned to <var> . (e.g. READ #7, KEY%; LINE TOTAL. STRING$)

REM (<remark>)

The REM statement may be used to document a program. (e.g. 10 REM this is a remark).

REMARK (<string>)

same as REM.

CBASIC™ is a trademark of Compiler Systems.

RENAME (newname$, oldname$)

changes the name of the file selected by oldname$ to newname$. (e.g. RENAME "SAMPLE.BAS", "TEMP.BAS") (e.g. IF RENAME (NEWFILE$,TEMP.BAS) THEN RETURN)

RESTORE

a RESTORE statement allows rereading the DATA statements. (e.g. 10 RESTORE)

RETURN

control is returned from a subroutine to the calling routine. (e.g. RETURN)

RIGHT$ (A$,I%)

returns a string consisting of the I% rightmost character of A$. (e.g. B$=RIGHT$(TEST$,2)

RND

The RND function generates a uniformly distributed random number between 0 and 1. (e.g. X=10*RND)

SADD(A$)

returns the address of the string assigned to the argument A$. (e.g. X%=SADD(TOTAL$)

SAVEMEM <constant> , <exp>

reserves space (value of constant) for a machine language program by the name of <exp> . (e.g. SAVEMEM 256, "TEST.COM")

SGN(X)

if X > 0, returns a 1. If X = 0, returns a 0. If X < 0, returns a −1. (e.g. Y=SGN(X))

SIN(X)

returns the sine of the X. (e.g. X=SIN(Y))

SIZE (A$)

returns the size, in blocks, of the file specified by A$. (X% = SIZE ("SAMPLE.PRN"))

STOP

program execution terminates and all open files are closed. The print buffer is emptied and control returns to the host system. (e.g. STOP)

STR$ (X)
returns the ASCII string which represents the value of the X. See VAL. (e.g. B$=STR$ (3.141617))

SQR (X)
returns the square root of the absolute value of the X. (e.g. X=SQR(Y)).

TAB (<exp>)
positions the output buffer pointer to the position specified by the integer value of the <exp>. (e.g. PRINT TAB (10);X)

TAN (X)
returns the tangent of the expression (in radians). (e.g. X=TAN(A)).

TRACE
CRUN <filename> TRACE <number>, <number>
The first optional number is used to specify the statement where the trace is to begin. The 2nd optional number specifies where the trace is to stop. NOTE: TOGGLE E MUST BE SET AT COMPILE-TIME. (e.g. CBASIC SAMPLE $E
CRUN SAMPLE TRACE 10,100)

UCASE$ (A$)
returns a string in which the lower case characters in A$ have been translated to uppercase. (IF UCASE$(ANS$) = "YES" THEN RETURN)

VAL (A$)
converts A$ into a floating point number. See STR$. (e.g. X=VAL(A$)).

WEND
denotes the end of the closest unmatched WHILE. See WHILE.

WHILE <exp>
all statements between the WHILE and WEND statement will be repeated until the value of the expression is zero. If <exp> = 0 initially, the loop is skipped.

```
e.g. WHILE X > 0
       X = X + 1
       PRINT X
     WEND
```

CBASIC™ is a trademark of Compiler Systems.

XREF <filename> [disk ref] [$ <toggles>] [' <title> ']
produces a file which contains an alphabetized list of all identifiers used in a CBASIC program. Toggles are explained below and 'title' is used to print a title on each page.

TOGGLES:

A listing to list device and disk file.

B. suppresses output to disk file.

C same effect as specifying A and B

D output to be 80 columns wide instead of 132.

E produces output with only the identifiers and their usage.

F(x) changes page length to x.

G suppresses heading lines and formfeeds.

(e.g. XREF SAMPLE B: $EAF(40))
(e.g. XREF PAYROLL CF(55) ' RUN 15 on 12/25/81')

CBASIC Compile-Time TOGGLES

The toggles are invoked by typing a "$" before the desired letter. (e.g. CBASIC SAMPLE $BE)

B suppresses the listing of the program on the console during compilation (initially off).

C suppresses the generation of an INT file (initially off).

D suppresses translation of lower case letters to upper case (initially off). (e.g. "YES" doesn't = "yes")

E will cause the run-time program to accompany any error messages with the CBASIC line number in which the error occurred (initially off).

F will cause the compiled output listing to be printed on the system list device in addition to the console (initially off).

G will cause the compiled output listing to be written to the diskette (initially off).

COMPILER DIRECTIVES

Directives are used to control the action of the compiler.

%LIST
will turn on the listing after a %NOLIST was executed.

%NOLIST
will stop the listing at the console during compile time.

%INCLUDE <filename>
insert the file specified in the INCLUDE statement in the source listing immediately following the %INCLUDE. (e.g. %INCLUDE B:SAMPLE.BAS)

%PAGE <constant>
sets the length of a page output to the printer. Must be positive integer (default = 64).

%EJECT
positions the listing directed to the printer and the disk to the top of the next page (outputs a formfeed).

%CHAIN <constant>, <constant>, <constant>, <constant>
used to set the size of the main program's constant, code, data and variable areas.

CBASIC™ ERROR MESSAGES

AC The string used as the argument in an ASC function evaluated to a null string.

BF A branch into a multiple line function from outside the function was attempted.

BN An invalid numeric constant was encountered (Compile-time Error).

The value following the BUFF option in an OPEN or CREATE statement is less than 1 or greater than 52.

CC A chained program's code area is larger than the main program's code area. Use the %CHAIN directive in the main program.

CD A chained program's data area is larger than the main program's data area. Use the %CHAIN directive in the main program.

CE The file being closed could not be found in the directory. This could occur if the file name had been changed with the RENAME function.

CF A chained program's constant area is larger than the main program's constant area. Use the %CHAIN directive in the main program.

CI An invalid file name was detected in a %INCLUDE directive. The file name may not contain a?,*, or:.

CP A chained program's variable storage area is larger than the main program's variable storage area. Use the %CHAIN directive in the main program.

CS A chained program reserved a different amount of memory with a SAVEMEM statement than the main program (Compile-time Error).

A COMMON statement, which was not the first statement in a program, was detected. Only a compiler directive such as %CHAIN may proceed a COMMON statement.

CU A close statement specified a file number that was not active.

CV An improper definition of a subscripted variable in a common statement. Possibly the subscript count is not a constant or there is more than one constant. One constant must appear in parenthesis and it specifies the number of subscripts.

DE A disk error occurred while trying to read the BAS file.

DF An OPEN or CREATE was specified with a file number that was already active. (CBASIC-2)

There was no space on the disk or the disk directory was full. The intermediate file was not created. (CBASIC-1)

DL The same line number was used on two different lines. Other compiler errors may cause a DL error message to be printed even if duplicate line numbers do not exist.

DP A variable in a DIM statement was previously defined.

DR Disk read error (reading unwritten data in random access).

DU A DELETE statement specified a file number that was not active.

DW An error occurred while writing to a file. This occurs when either the directory or the disk is full.

DZ A number was divided by zero. The result is set to the largest valid CBASIC number.

EF A read past the end of file occurred on a file for which no IF END statement had been executed. (CBASIC-2)

A number in exponential format was input with no digits following the E. (CBASIC-1)

ER An attempt was made to write a record of length greater than the maximum record size specified in the associated OPEN, CREATE or FILE statement.

FA A function name appears on the left side of an assignment statement but is not within that function. In other words the only function name that may appear to the left of an equal sign is the name of the function currently being compiled.

FD A function name that has been previously defined is being redefined in a DEF statement.

FE A mixed mode expression exists in a FOR statement which the compiler can not correct. Probably the expression following the TO is of a different type than the index.

FI An expression which is not an unsubscripted numeric variable is being used as a FOR loop index.

FL A field length greater than 255 bytes was encountered during a READ LINE. The first 255 characters of the record are retained; the other characters are ignored.

FN A function reference contains an incorrect number of parameters.

FP A function reference parameter type does not match the parameter type used in the function's DEF statement.

FR An attempt was made to rename a file to an existing file name.

FT A file statement was executed when 20 files were already active.

FU A function has been referenced before it has been defined (Compile-time Error).

An attempt was made to read or write a file that was not active (Run-time Error).

IC Invalid character in BASIC statement.

IE An expression used immediately following an IF evaluates to type string. Only type numeric is permitted.

IF A variable used in a file statement is of type numeric where type string is required (Compile-time Error).

A file name was invalid. Most likely an invalid character was found in the file name. A colon may never appear imbedded in the name proper. Question marks and asterisks may only appear in ambiguous file names. This error will also result if the file name was a null string (Run-time Error).

CBASIC™ ERROR MESSAGES

II Invalid input from the console.

IP An input prompt string was not surrounded by quotes.

IR A record number less than one was specified.

IS A subscripted variable was referenced before it was dimensioned.

IT An invalid compiler directive was encountered. A parameter required by the directive may be out of range or missing. Or the directive may be misspelled.

IU A variable defined as an array is used with no subscripts.

IV An attempt was made to execute an INT file created by a version 1 compiler. To use CRUN2 a program must be recompiled using the version 2 compiler, CBAS2. This error will also result from attempting to execute an INT file which is empty.

IX A FEND statement was encountered prior to executing a RETURN statement. All multiple line functions must exit with a RETURN statement.

LN The argument given in the LOG function was zero or negative. The value of the argument is returned.

LW A line width less than 1 or greater than 133 was specified in an LPRINTER WIDTH statement.

ME An error occurred while creating or extending a file because the disk directory was full.

MF An expression evaluates to type string when type numeric is required. (CBASIC)

File identifier too large or zero. (EBASIC only)

MM Variables of type string and type numeric are combined in the same expression.

MP The third parameter in a MATCH function was zero or negative.

MS A numeric expression was used where a string expression is required.

CBASIC™ is a trademark of Compiler Systems.

CBASIC ™ ERROR MESSAGES

ND A FEND statement was encountered without a corresponding DEF statement. This error could be the result of an improper DEF statement.

NE A negative number was specified following the raise to a power operator (^). The absolute value is used in the calculation.

NF The file number specified was less than 1 or greater than 20.

NI No INT file found in the directory (CBASIC-1).

A variable referenced by a NEXT statement does not match the variable referenced by the associated FOR statement (CBASIC-2).

NM There was insufficient memory to load the program.

NN An attempt was made to print a number with a PRINT USING statement but there was not a numeric data field in the USING string.

NP No applicable production exists.

NS No BASIC file found (EBASIC only).

An attempt was made to print a string with a PRINT USING statement but there was not a string field in the USING string (Both CBASIC's).

NU A NEXT statement occurs without an associated FOR statement.

OD A READ statement was executed with no DATA statement, or all data statements having already been read.

OE An attempt was made to open a file that didn't exist and for which no IF END statement had been previously executed.

OF A branch out of a multiple line function from inside the function was attempted.

A calculation produced a number too large. The result is set to the largest valid CBASIC number (CBASIC-2 WARNING).

CBASIC™ ERROR MESSAGES

OI The expression specified in an ON. . .GOSUB or an ON. . .GOTO statement evaluated to a number less than 1 or greater than the number of line numbers contained in the statement.

OM The program ran out of memory during execution.

OO More than 25 ON statements were used in the program.

PM A DEF statement appeared within a multiple line function. Functions may not be nested.

QE An attempt was made to PRINT to a file string containing a quotation mark.

RB Random access was attempted to a file activated with the BUFF option specifying more than one buffer.

RE An attempt was made to read past the end of a record in a fixed file.

RG A RETURN occurred for which there was no GOSUB.

RU A random read or print was attempted to other than a fixed file.

SB An array subscript was used which exceeded the boundaries for which the array was defined.

SE The source line contained a syntax error.

SF A SAVEMEM statement uses an expression of type numeric to specify the file to be loaded. The expression must be a string. Possibly the quotation marks are left off a string constant.

SL A concatenation operation resulted in a string of more than 255 bytes.

SN A subscripted variable contains an incorrect number of subscripts.

CBASIC™ ERROR MESSAGES

SO The expression is too complex and should be simplified and placed on more than one line (Run-time Error).

The file specified in a SAVEMEM statement could not be located on the referenced disk. The expression specifying the file name must include the type if one is present. A type of COM is not forced (Compile-time Error).

SQ A negative number was specified in the SQR function. The absolute value is used.

SS The second parameter of a MID$ function was zero or negative.

TF An attempt was made to have more than 20 active files simultaneously.

TL A TAB statement contained a parameter less than 1 or greater than the current line width.

TO The program is too large for the system. The program must be simplified or the system size increased.

TZ Attempt to evaluate tangent of "pi over two".

UL A line number that does not exist has been referenced.

UN A PRINT USING statement was executed with a null edit string.

US A string has been terminated by a carriage return rather than by quotes.

VO Variable names are too long for one statement.

WE The expression immediately following a WHILE statement is not numeric.

WN WHILE statements are nested to a depth greater than 12. CBASIC has an arbitrary limit of 12 for nesting of WHILE statement.

WR An attempt was made to write to a file after it had been read but before it had been read to the end of the file.

WU A WEND statement occurs without an associated WHILE statement.

ABS(exp)	Absolute value of expression	Y=ABS(A+B)
ATN(exp)	Arctangent of the expression (in radians)	PRINT ATN(A)
ASC(string)	Returns the ASCII value of the first character of a string	PRINT ASC(A$)
AUTO	AUTO [line] [,inc] *Generate line numbers automatically.*	AUTO 100,50
CALL	CALL variable [(arg list)] *Call an assembly language or FORTRAN subroutine.*	CALL ROUT (I,J,K)
CDBL(exp)	Convert the expression to a double precision number	A=CDBL(Y)
CHAIN	CHAIN [MERGE] filename [,[line exp] [,ALL] [,DELETE range]] *Call a program and pass variables to it.* *MERGE with ASCII files allows overlays.* *If line exp is omitted, CHAINed program starts with the first line.* *,ALL means all variables will be passed, otherwise variables designated with COMMON.* *DELETE allows deletion of an overlay before CHAIN is executed.*	CHAIN "PROG1",1000 CHAIN MERGE"OVRLY2",1200
CHR$(exp)	Returns a one-character string whose character has the ASCII code of exp	PRINT CHR$(48)
CINT(exp)	Convert the expression to an integer	B=CINT(B)
CLEAR	CLEAR [,[exp1] [,exp2]] *Clear program variables. Exp1 sets end of memory and exp2 sets amount of stack space.*	CLEAR ,32768 CLEAR ,,2000
CLOSE	CLOSE [[#] f [,[#] f...]] *Closes disk files. If no argument, all open files are closed.*	CLOSE 6
COMMON	COMMON list of variables *Pass variables to a CHAINed program.*	COMMON A,B( ),C$
CONT	CONT *Continue program execution.*	CONT

COS(exp)	Cosine of the expression (in radians)	A=COS(2.3)
CSNG(exp)	Convert the expression to a single precision number	C=CSNG(X)
CVI(string) CVS(string) CVD(string)	Converts a 2-character string to an integer (CVI). Converts a 4-character string to a single precision number (CVS). Converts an 8-character string to a double precision number (CVD).	Y!=CVS(N$) A%=CVI(B$) C#=CVD(X$)
DATA	DATA list of constants *Lists data to be used in a READ statement.*	DATA 2.3,"PLUS",4
DEF	DEF FNx[(arg list)]=exp *Define an arithmetic or string function.*	DEF FNA (X,Y)= SQR(X*X+Y*Y)
	DEF USRn=address *Define the entry address for the nth assembly language subroutine.*	DEF USR3=&2000
	DEFtype range(s) of letters *Define default variable types where "type" is INT, SNG, DBL, or STR.*	DEFINT I-N DEFSTR A,W-Z DEFDBL D
DELETE	DELETE start line[- end line] *Delete program lines.*	DELETE 20 DELETE 20-25
DIM	DIM list of subscripted variables *Allocate space for arrays and specify maximum subscript values.*	DIM A(3),B$(10,2,3)
EDIT	EDIT line number *Edit a program line. See Edit Mode Subcommands.*	EDIT 110
END	END *Stop program, close all files and return to BASIC command level.*	END
EOF(f)	Returns true (−1) if file is positioned at its end	IF EOF(1) GOTO 300
ERASE	ERASE variable [,variable...] *Release space and variable names previously reserved for arrays.*	ERASE A,B$

ERL	Error line number	PRINT ERL
ERR	Error code number	IF ERR=62 THEN...
ERROR	ERROR code *Generate error of code (see table). May call user ON ERROR routine or force BASIC to handle error.*	ERROR 17
EXP(exp)	Raises the constant e to the power of expression	B=EXP(C)
FIELD	FIELD [#] f,n AS string variable [,n AS string variable ...] *Define fields in a random file buffer. Note: PRINT#[USING] and [LINE] INPUT# statements to random files write and read data into the FIELD buffer.*	FIELD #1,3 AS A$,7 AS B$
FILES	FILES [filename] *List files in disk directory that match filename. ? matches any character. * matches any name or extension.*	FILES FILES "*.BAS" FILES "TEST.BAS" FILES "B:*.*"
FIX(exp)	Returns truncated integer of expression	J=FIX(A/B)
FOR	FOR variable=exp TO exp [STEP exp] *Used with NEXT statement to repeat a sequence of program lines. The variable is incremented by the value of STEP.*	FOR DAY=1 TO 5 STEP 2
FRE(exp)	Gives memory free space not used by BASIC	PRINT FRE(0)
FRE(string)	Returns remaining memory free space	PRINT FRE(A$)
GET	GET [#] f [,record number] *Read a record from a random disk file.*	GET #1,17*I+1
GOSUB	GOSUB line number *Call a BASIC subroutine by branching to the specified line number. See RETURN.*	GOSUB 210
GOTO	GOTO line number *Branch to specified line number.*	GOTO 90
HEX$(exp)	Converts a number to a hexadecimal string	H$=HEX$(100)

IF/THEN IF exp THEN statement [:statement...] [ELSE statement...]
If exp is not zero, the THEN clause is executed. Otherwise, the ELSE clause or next statement is executed.
IF X <Y THEN Y=X ELSE Y=A

IF/GOTO IF exp GOTO line [ELSE statement...]
If exp is not zero, the GOTO clause is executed. Otherwise the ELSE clause or next statement is executed.
IF ENDVAL > 0 GOTO 200

INP(port) Inputs a byte from an input port
PRINT INP(21)

INPUT INPUT [;] [prompt string;] variable [,variable...]
INPUT [;] [prompt string,] variable [,variable...]
Read data from the terminal. Semicolon after INPUT suppresses echo of carriage return/line feed. Semicolon after prompt string causes question mark after prompt. Comma after prompt string suppresses question mark.
INPUT "VALUES";A,B

INPUT #f, variable [,variable...]
Read data from a disk file.
INPUT #1,A,B

INKEY$ Returns either a one-character String read from terminal or null String if no character pending at terminal
A$=INKEY$

INPUT$(length [,[#] f]) Returns a string of length characters read from console or from a disk file. Characters are not echoed.
X$=INPUT$(4)
X$=INPUT
X$=INPUT$(5,#2)

LINE INPUT LINE INPUT [;] [prompt string;] string variable
Read an entire line from the terminal. Semicolon after LINE INPUT suppresses echo of carriage return/line feed.
LINE INPUT A$
LINE INPUT "NAME";N$

LINE INPUT #f,string variable
Read an entire line from a disk file.
LINE INPUT #2,B$

INSTR([exp,]string 1,string2) Returns the first position of the first occurrence of string2 in string1 starting at position exp
INSTR(A$,":")
INSTR(3,X$,Y$)

INT(exp) Evaluates the expression for the largest integer contained
C=INT(X+3)

KILL KILL filename
Delete a disk file.
KILL "INVEN.BAS"

LEFT$(string,length) Returns leftmost length characters of the string expression
B$=LEFT$(X$,8)

LEN(string) Returns the length of a string
PRINT LEN(B$)

LET [LET] variable=exp
Assign a value to a variable.
LET X=I+5

LINE INPUT — LINE INPUT [;] [prompt string;] string variable

Read an entire line from the terminal. Semicolon after LINE INPUT suppresses echo of carriage return/line feed.

```
LINE INPUT A$
LINE INPUT "NAME";N$
```

LINE INPUT #f,string variable

Read an entire line from a disk file.

```
LINE INPUT #2,B$
```

LIST — LIST [line[-[line]]]

List program lines at terminal.

```
LIST 100-1000
```

LLIST — LLIST [line[-[line]]]

List program lines at printer.

```
LLIST 50-
```

LOAD — LOAD filename [,R]

Load a program file. ,R option means RUN.

```
LOAD "INVEN"
```

LOC(f) — Returns next record number to read or write (random file), or number of sectors read or written (sequential file)

```
PRINT LOC(1)
```

LOG(exp) — Gives the natural logarithm of the expression

```
D=LOG(Y-2)
```

LPOS(n) — Returns carriage position of line printer (n is dummy argument)

```
IF LPOS(3) > 60...
```

LSET — LSET field variable=string exp

Store data in random file buffer left justified. Or left justify a non-disk string in a given field.

```
LSET A$="JOHN JONES"
LSET B$=MKS$(MAX)
```

MERGE — MERGE filename

Merge program on disk with program in memory. Program on disk must have been SAVEd in ASCII mode.

```
MERGE "SUB1"
```

**MID$** — MID$(string1,n[,m]) =string2

Replace a portion of string1 with string2. Start at position n and replace m characters.

```
MID$ (A$,14)="KS"
```

MID$(string,start [,length])

Returns characters from the middle of the string starting at the position specified to the end of the string or for length characters

```
A$=MID$(X$,5,10)
```

MKI$(value) — Converts an integer to a 2-character string (MKI$).
MKS$(value) — Converts a single precision value to a 4-character string (MKS$).
MKD$(value) — Converts a double precision value to an 8-character string (MKD$).

```
LSET D$=MKS$(A)
LSET A$=MKI$(B%)
```

NAME	NAME old filename AS new filename *Change the name of a disk file.*	NAME "SUB1" AS "SUB2"
NEXT	NEXT variable [,variable...] *Delimits the end of a FOR loop.*	NEXT I
NEW	NEW *Delete current program and variables.*	NEW
NULL	NULL exp *Set the number of nulls printed after each line.*	NULL 2
OCT$(exp)	Converts a number to an octal string	O$=OCT$(100)
ON ERROR GOTO	ON ERROR GOTO line *Enables error trap subroutine beginning at specified line. If line=0, disables error trapping. If line=0 inside error trap routine, forces BASIC to handle error.*	ON ERROR GOTO 1000
ON/GOSUB	ON exp GOSUB line [,line] *GOSUB to statement specified by expression. (If exp=1, to 20; if exp=2, to 20; if exp=3, to 40; otherwise, error.)*	ON DATE%+1 GOSUB 20,20,40
ON/GOTO	ON exp GOTO line [,line...] *Branch to statement specified by exp. (If exp=1, to 20; if exp=2, to 30; otherwise, error.)*	ON INDEX GOTO 20,30
OPEN	OPEN mode,[#] f,filename [reclen] *Open a disk file. Mode must be one of:* *I (sequential input file)* *O (sequential output file)* *R (random input/output file)*	OPEN "O",#1,"OUTPUT"
OPTION BASE	OPTION BASE n *Declare the minimum value for array subscripts. n is 0 or 1.*	OPTION BASE 1
OUT	OUT port,byte *Puts byte specified to output port specified.*	OUT 41,16+DATAO%
PEEK(exp)	Reads a byte from memory location specified by expression	PRINT PEEK(&2000)
POKE	POKE address,byte *Puts byte specified into memory location specified.*	POKE &23100,255

POS(n)	Returns carriage position of terminal (n is dummy argument)	IF POS(3) > 60...
PRINT	PRINT [USING format string;] exp [,exp...] *Print data at the terminal using the format specified. See table for format characters.*	PRINT USING "!";A$,B$
	PRINT #f, [USING format string;] exp [,exp...] *Write data to a disk file.*	PRINT #4,A,B
	LPRINT [USING format string;] variable [,variable] *Write data to a line printer.*	LPRINT A,B
PUT	PUT [#] f [,record number] *Write data from a random buffer to a data file.*	PUT #3,4
RANDOMIZE	RANDOMIZE [exp] *Reseed the random number generator.*	RANDOMIZE 5
READ	READ variable [,variable...] *Read data from a DATA statement into the specified variables.*	READ I,X,A$
REM	REM any text *Allows user to insert comments in program (not executed).* *NOTE: ":" does not terminate a REM statement.*	REM COMPUTE AVERAGE
RENUM	RENUM [[new line] [,[old line] [,inc]]] *Renumber program lines.*	RENUM 100,,100
RESET	RESET *Reinitialize CP/M disk information. Use after changing diskettes.*	RESET
RESTORE	RESTORE [line number] *Resets DATA pointer so that DATA statements may be re-read.*	RESTORE
RESUME	RESUME or RESUME 0 *Returns from ON ERROR routine to statement that caused error.*	RESUME
	RESUME NEXT *Returns to statement after the one that caused the error.*	RESUME NEXT
	RESUME line *Returns to the specified line.*	RESUME 100

RETURN	RETURN *Return from subroutine to statement following last GOSUB executed.*	RETURN
RIGHT$(string,length)	Returns rightmost length characters of the string expression	C$=RIGHT$(X$,8)
RND[(exp)]	Generates a random number. Expression: < 0 seed new sequence =0 return previous random number >0 or omitted, return new random number	E=RND(1)
RSET	RSET field variable=string exp *Store data in a random file buffer right justified. Or right justify a non-disk string in a given field.*	RSET B$="CORRECT" RSET C$=MKS$(COUNT)
RUN	RUN [line number] *Run a program (from line number).* RUN filename [,R] *Load a program from disk and run it. ,R used to keep files open.*	RUN RUN 50 RUN "TEST"
SAVE	SAVE filename [,A or ,P] *Save the program in memory with name "filename.", A saves program in ASCII. ,P protects file.*	SAVE "PROG",P
SGN(exp)	1 if expression > 0 0 if expression = 0 −1 if expression < 0	B=SGN(X+Y)
SIN(exp)	Sine of the expression (in radians)	B=SIN(A)
SPACE$(exp)	Returns a string of exp spaces	S$=SPACE$(20)
SPC(exp)	Used in PRINT statements to print spaces	PRINT SPC(5),A$
SQR(exp)	Square root of expression	C=SQR(D)
STOP	STOP *Stop program execution, print BREAK message, and return to command level.*	STOP

SPECIAL CHARACTERS (↑ means control)

↑A	Enters Edit Mode on line being typed or last line typed
↑C	Interrupts program execution, returns to BASIC command level and types OK
↑G	Rings the bell at the terminal
↑H	Deletes last character typed
↑I	Tab. Tab stops are every 8 columns
↑O	Halts/resumes program output
↑R	Retypes the line currently being typed
↑S	Suspends program execution
↑Q	Resumes execution after control-S
↑U	Deletes line being typed
↑X	Deletes line being typed
< return >	Ends every line typed in
< linefeed >	Used to break a logical line into physical lines
< rubout >	Deletes last character typed
< escape >	Escapes Edit Mode subcommands
.	Current line for EDIT, RENUM, DELETE, LIST, LLIST commands
&O or &	Prefix for octal constant
&H	Prefix for hexadecimal constant
:	Separates statements typed on the same line
?	Equivalent to PRINT statement (L? is not equivalent to LPRINT)

EDIT MODE SUBCOMMANDS

Subcommand	Function
A	Restore original line and restart EDIT at the start of the line.
nCc	Change n character(s).
nD	Delete n character(s) at the current position.
E	End editing and save changes but don't type the rest of the line.
Hstring < escape >	Delete the rest of the line and insert string.
Istring < escape >	Insert string at current position.
nKc	Kill all characters up to the nth occurrence of c.
L	Print the rest of the line and go to the start of the line.
Q	Quit editing and restore original line.
nSc	Search for nth occurrence of c.
Xstring < escape>	Go to the end of the line and insert string.
< rubout >	Backspace over characters. In Insert mode, delete characters.
< return >	End editing and save changes.
< space >	Move to the next character

OPERATORS

Symbol	Function
=	Assignment or equality test
–	Negation or subtraction
+	Addition or string concatenation
*	Multiplication
/	Division (floating point result)
↑	Exponentiation
\	Integer division (integer result)
MOD	Integer modulus (integer result)
NOT	One's complement (integer)
AND	Bitwise AND (integer)
OR	Bitwise OR (integer)
XOR	Bitwise exclusive OR (integer)
EQV	Bitwise equivalence (integer)
IMP	Bitwise implication (integer)
=, <, >, <=, =<, >=, => <>	Relational tests (result is TRUE = –1 or FALSE = 0)

The precedence of operators is:

(1) Expressions in parentheses
(2) Exponentiation (A ↑ B)
(3) Negation (–X)
(4) *, /
(5) \
(6) MOD
(7) +, –
(8) Relational operators (=, <>, <, >, <=, >=)
(9) NOT
(10) AND
(11) OR
(12) XOR
(13) IMP
(14) EQV

PRINT USING Format Field Specifiers

NUMERIC

Specifier	Possible Digits	Field Characters	Definition	Example
#	1	1	Numeric field	# # # #
.	0	1	Decimal point	# . #
+	0	1	Print leading or trailing sign. Positive numbers will have "+", negative numbers will have "−".	+ # # # # # +
− .	0	1	Trailing sign. Prints "−" if negative, otherwise blank.	# #.# #−
**	2	2	Leading asterisk	**###.##
$$	1	2	Floating dollar sign. $ is placed in front of the leading digit.	$$##.##
**$	2	3	Asterisk fill and floating dollar sign	**$#.##
,	1	1	Use comma every three digits (left of decimal point only.)	##,###.##
↑ ↑ ↑ ↑	0	4	Exponential format. Number is aligned so leading digit is non-zero.	#.##↑↑↑↑
underscore	0	1	Next character literal	_!#.#

STRING

!	Single character	!
\<spaces>\	2+ number of spaces character field	\ \
&	Variable length field	&

MICROSOFT BASIC COMPILER

The following direct mode commands are not implemented on the compiler and will generate an error message:

AUTO	CLEAR	CLOAD
CSAVE	CONT	DELETE
EDIT	LIST	LLIST
RENUM	COMMON	SAVE
LOAD	MERGE	ERASE
		NEW

BASIC COMPILER COMMANDS AND SWITCHES

Format of a BASIC compiler command:

[device:] [obj filename] [,[device:] [list filename]]=[device:]source filename[/switch...]

Switches:

/E Use /E if the program contains an ON ERROR GOTO statement with the RESUME <line number> statement. Line numbers will be included in the binary file.

/X Use /X if the program contains an ON ERROR GOTO statement with the RESUME, RESUME 0, or RESUME NEXT statement. Line numbers will be included in the binary file.

/N Do not list generated object code.

/D Generate debug/checking code at runtime.

/S Quoted strings of more than 4 characters will be written to the binary file as they are encountered.

/4 Compiler will recognize the lexical conventions of the Microsoft 4.51 BASIC-80 Interpreter. (May not be used together with /T.)

/C Relax line numbering constraints. Line numbers may be in any order or they may be eliminated, but they may not be repeated. With /C, the underline character causes the remainder of the physical line to be ignored, and the next physical line is considered to be a continuation of the current logical line. /C and /4 may not be used together.

/T Use BASIC-80 version 4.51 execution invention. (May not be used together with /4)

/Z Use Z80 opcodes wherever possible.

SAMPLE COMPILE AND GO

1. Compile TEST.BAS to create TEST.REL

```
A>BASCOM
*TEST,TTY:=TEST/N/D
```

2. Link TEST.REL with BASLIB.REL and execute

```
A>L80
*TEST/G
```

BASIC COMPILER ERROR MESSAGES

Compile-time Fatal Errors:

SN	Syntax error
OM	Out of memory
SQ	Sequence error
TM	Type mismatch
TC	Too complex
BS	Bad subscript
LL	Line too long
UC	Unrecognizable command
OV	Math overflow
/0	Division by zero
DD	Array already dimensioned
FN	FOR/NEXT error
FD	Function already defined
UF	Function not defined
WE	WHILE/WEND error
/E	Missing /E switch
IN	INCLUDE ERROR
LS	Long string constant
/X	Missing /X switch

Compile-time Warning Errors:

ND	Array not dimensioned
SI	Statement ignored

Run-time Error Messages:

2	Syntax error
3	RETURN without GOSUB
4	Out of data
5	Illegal function call
6	Floating overflow or integer overflow
9	Subscript out of range
11	Division by zero
14	Out of string space
20	RESUME without error
21	Unprintable error
50	Field overflow
51	Internal error
52	Bad file number
53	File not found
54	Bad file mode
55	File already open
57	Disk I/O error
58	File already exists
61	Disk full
62	Input past end
63	Bad record number
64	Bad filename
67	Too many files

DEC - HEX - OCT - ASCII

DEC	HEX	OCT	ASCII
0	00	000	NUL
1	01	001	SOH
2	02	002	STX
3	03	003	ETX
4	04	004	EOT
5	05	005	ENQ
6	06	006	ACK
7	07	007	BEL
8	08	010	BS
9	09	011	HT
10	0A	012	LF
11	0B	013	VT
12	0C	014	FF
13	0D	015	CR
14	0E	016	SO
15	0F	017	SI
16	10	020	DLE
17	11	021	DC1
18	12	022	DC2
19	13	023	DC3
20	14	024	DC4
21	15	025	NAK
22	16	026	SYN
23	17	027	ETB
24	18	030	CAN
25	19	031	EM
26	1A	032	SUB
27	1B	033	ESC
28	1C	034	FS
29	1D	035	GS
30	1E	036	RS
31	1F	037	US
32	20	040	SP
33	21	041	!
34	22	042	"
35	23	043	#
36	24	044	$
37	25	045	%
38	26	046	&
39	27	047	'
40	28	050	(
41	29	051	)
42	2A	052	*
43	2B	053	+
44	2C	054	,
45	2D	055	-
46	2E	056	.
47	2F	057	/
48	30	060	0
49	31	061	1
50	32	062	2
51	33	063	3
52	34	064	4
53	35	065	5
54	36	066	6
55	37	067	7
56	38	070	8
57	39	071	9
58	3A	072	:
59	3B	073	;
60	3C	074	<
61	3D	075	=
62	3E	076	>
63	3F	077	?
64	40	100	@
65	41	101	A
66	42	102	B
67	43	103	C
68	44	104	D
69	45	105	E
70	46	106	F
71	47	107	G
72	48	110	H
73	49	111	I
74	4A	112	J
75	4B	113	K
76	4C	114	L
77	4D	115	M
78	4E	116	N
79	4F	117	O
80	50	120	P
81	51	121	Q
82	52	122	R
83	53	123	S
84	54	124	T
85	55	125	U
86	56	126	V
87	57	127	W
88	58	130	X
89	59	131	Y
90	5A	132	Z
91	5B	133	[
92	5C	134	\
93	5D	135	]
94	5E	136	↑
95	5F	137	←
96	60	140	`
97	61	141	a
98	62	142	b
99	63	143	c
100	64	144	d
101	65	145	e
102	66	146	f
103	67	147	g
104	68	150	h
105	69	151	i
106	6A	152	j
107	6B	153	k
108	6C	154	l
109	6D	155	m
110	6E	156	n
111	6F	157	o
112	70	160	p
113	71	161	q
114	72	162	r
115	73	163	s
116	74	164	t
117	75	165	u
118	76	166	v
119	77	167	w
120	78	170	x
121	79	171	y
122	7A	172	z
123	7B	173	{
124	7C	174	|
125	7D	175	}
126	7E	176	~
127	7F	177	DEL