C&C Computers and Communications

# N82-BASIC REFERENCE CARD

PC-8300

NEC

Printed in Japan
78118302

PC-8300-RC

NEC

# TABLE OF CONTENTS

## SPECIAL SYMBOLS

**APOSTROPHE (')**
Same as single quotation mark. Can be used to precede remarks or comments in a statement. Equivalent to "REM".

**COLON (:)**
Used to separate multi-statements within one program line, which saves memory space.

**COMMA (,)**
The comma separates variables or data within a print command into 14 unit widths called space zones.

**DOLLAR ($)**
Indicates that the variable preceding the dollar sign is to be handled in BASIC as a literal character string.

**DOUBLE QUOTATION MARKS (" ")**
Used to enclose character strings.

**EXCLAMATION MARK (!)**
Indicates that the variable preceding it is to be handled in BASIC as a single precision variable.

**HYPHEN (-)**
The hyphen indicates a range, in place of the word "to", e.g. 1 – 19. The same character as the minus sign.

**PERCENT (%)**
Indicates that the variable preceding the percent sign is to be handed by BASIC as an integer.

**PERIOD (.)**
The period is used to refer to the last program line input. It is also used to point to the line in which an error has occurred during program execution.

**POUND (#)**
Indicates that the variable preceding the pound sign is to be handled in BASIC as a double precision variable.

**QUESTION MARK (?)**
This symbol can be used as a short form of "PRINT".

**SEMICOLON (;)**
The semicolon is used in PRINT and INPUT statements as a non-spacing separator.

## RELATIONAL OPERATORS

= Equal to
<>, >< Not Equal to (Greater-than or Less-than)
< Less-than
> Greater-than
<=, =< Less-than or Equal to
>=, => Greater-than or Equa to

## LOGICAL OPERATORS

(True-False Truth Tables)

| NOT | AND | OR | XOR | IMP | EQV |
|---|---|---|---|---|---|
| | T AND T=T | T OR T=T | T XOR T=F | T IMP T=T | T EQV T=T |
| NOT T=F | T AND F=F | T OR F=T | T XOR F=T | T IMP F=F | T EQV F=F |
| NOT F=T | F AND T=F | F OR T=T | F XOR T=T | F IMP T=T | F EQV T=F |
| | F AND F=F | F OR F=F | F XOR F=F | F IMP F=T | F EQV F=T |

## HIERARCHY OF OPERATIONS

1. Expressions enclosed by parentheses
2. Functions (ABS, SQR, etc.)
3. Exponential Arithmetic (^)
4. Negative Sign (–)
5. Multiplication and Division (*,/)
6. Integer Division (\)
7. Modular Division (MOD)
8. Addition and Subtraction (+, –)
9. Relational Operators (=, <, >, <>, <=, =>, etc)
10. Logical Operator NOT
11. Logical Operator AND
12. Logical Operator OR
13. Logical Operator XOR
14. Logical Operator IMP
15. Logical Operator EQV

# BASIC INSTRUCTIONS

## Format Description

1. Capitalized words are BASIC Reserved Words.

2. Lower case words contained within angle bracket < > symbols are parameters to be supplied by you.

3. Parentheses ( ), comma " , " double quotation marks "", equal sign = etc. are required to be typed in as shown in the format.

4. Braces { } indicate that the enclosed clause is optional.

5. Brackets [ ] denote that one of the enclosed words must be chosen.

## A

**ABS**
ABS (<numeric expression>)
Provides the absolute value of a number.

**AND**
<operand 1> AND <operand 2>
Tests multiple relational expressions.

**ASC**
ASC("<character>")
Provides the ASCII value of the specfied character.

**ATN**
ATN(<numeric expression>)
Provides the inverse tangent of the specified angle.

## B

**BEEP**
BEEP
Generates the "BEEP" sounc.

**BLOAD**
BLOAD"{<external device name>:}<file name>'
Loads a machine language file into the memory.

**BLOAD?**
BLOAD?"{<external device name> }<file name>"
Compares/verifies a machine language program in the memcry with another saved on cassette tape.

**BSAVE**
BSAVE "{<external device name>:}<file name>",
<start address>, <length>{ ,<execute start location>}
Saves a machine language program from the memory into a designated fle.

## C

**CDBL**
CDBL (<numeric expression>)
Converts integers or single precision real numbers to doube precision real numbers.

**CHR$**
CHR$ (<numeric expression>)
Changes a single value ASCII code to its matching string character.

**CINT**
CINT (<numeric expression>)
Converts single or double precision real numbers to integers.

**CLEAR**
CLEAR {<string area size>} {,<maximum memory used in BASIC>}
Initializes all variables, establishes the size of a string region and sets the memory bouncary.

**CLOAD**
CLOAD"<file name>"
Loads a recorded program from cassette tape into the memory.

**CLOAD?**
CLOAD? "<file name>"
Compares/verifies the program currently in memory with another program saved on cassette tape.

**CLOSE**
CLOSE {{#}<file number>}{, {#}<file number>} ...
Terminates input/output between a BASIC program and the data file(s).

**CLS**
CLS
Erases the display from the screen.

**COM**

COM [ON / OFF / STOP]

This command establishes, prohibits, or gives information about interruption by a data transmission circuit.

**CONT**

CONT

Restarts the execution of an interrupted program, for example, one stopped by STOP.

**COS**

COS (<numeric expression>)

Returns the cosine of an angle.

**CSAVE**

CSAVE "<file name>"

Saves a program currently in the memory onto cassette tape.

**CSRLIN**

CSRLIN

Returns the line number of the current cursor position.

# D

**DATA**

DATA <constant> {,<constant>} ...

A statement used to define information for the READ statement.

**DATE$**

DATE$ = "<year>/<month>/<day>"

A function used to set the year, month and day.

**DEFINT/SNG/DBL/STR**

DEF [INT / SNG / DBL / STR] <character range>

Defines the format of a variable.

**DIM**

DIM <variable name> (<maximum subscript value> {,<maximum subscript value>... })

Allocates memory space for storing an array.

# E

**EDIT**

EDIT {<line in which to start editing>}
{-<line in which to stop editing>}

Shifts from BASIC into TEXT mode for program editing.

**END**

END

Used to terminate program execution.

**EOF**

EOF (<file number>)

Determines if the end of a sequential file has been reached.

**EQV**

<operand 1> EQV <operand 2>

A logical operator that tests multiple relations

**ERL**

ERL

Used for displaying the line location of an error.

**ERR**

ERR

Provides the error code when an error occurs.

**ERROR**

ERROR <error code>

A statement used to simulate the occurrence of an existing error.

**EXEC**

EXEC <initial location>

Transfers control to a machine language subroutine in the memory.

**EXP**

EXP ([<arithmetic expression> / <numeric constant> / <numeric variable>])

Returns the value of "e" raised to the specified power in single precision format.

# F

**FILES**

FILES

Displays all the names of the files in the RAM.

**FIX**

FIX (<numeric expression>)
Returns the integer portion of a number.

**FOR... TO ... STEP ~ NEXT**

FOR<variable name> = <initial value> TO <final value>
{STEP<increment>}
.
.
.
NEXT {<variable name>{, <variable name list>}}
Repeats a series of instructions the cesignated number of times.

**FRE**

FRE (<expression>)
Returns the amount of unused memory that is available.

# G

**GOSUB ~ RETURN**

GOSUB <line number>
GOSUB transfers control to the specfied line number.

**GOTO**

[GOTO / GO TO] <line number>
Unconditional branch to the designated line number

# I

**IF...THEN...ELSE**
**IF...GOTO...ELSE**

IF <expression> [THEN <then clause> / GOTO <goto clause>]
{ELSE <else clause>}
Controls program execution functions based on conditions established by the evaluation of the <expression>.

**IMP**

<operator 1> IMP <operator2>
A logical operator used to test multiple implied relations.

**INKEY$**

INKEY$
Used to check if a character has been entered from the keybcard.

**INPUT**

INPUT {"<prompt statement>";}<variable 1>
{,<variable 2>}...
Allows data to be entered through the keyboard during program execution. The prompt statement is optional.

**INPUT$**

INPUT$ ([<integer constant> / <integer variable>] {,{#} <file number>})
Reads a character string of specified length, either from a designated file or from the keyboard

**INPUT#**

INPUT #<file number>,<variable 1>{,<variable 2>}...
Used to read data from an opened file into variables contained in the statement.

**INSTR**

INSTR ({<numeric expression>,}<character string 1>, <character string 2>)
Searches for a character string within a string and returns its position.

**INT**

INT(<numeric expression>)
INT rounds numbers to their integer value.

# K

**KEY**

KEY <key number>, "<character string>'
Defines the character string of the programmable function keys.

**KILL**

KILL "<file name. file type extension>"
The KILL command deletes a file.

# L

**LEFT$**

LEFT$ (<character sting>,<numeric expression>)
Returns the designated left portion of a string.

**LEN**

LEN ([<character string> / <character variable>])
Returns the number of characters contained in a string.

**LET**

{LET} <variable name> = <value>
LET assigns values to variable names

**LINE INPUT**

LINE INPUT {"<prompt string>";} <string variable>
Used to allow the input of an entire line of data.

**LIST/LLIST**

[LIST / LLIST] {<line number 1>} {– <line number 2>}
LIST displays the lines of a program on the screen. LLIST prints the lines of a program on the printer.

**LOAD**

LOAD "{<external device name>:} <file name>"{,R}
Loads a program file into memory.

**LOCATE**

LOCATE <horizontal coordinate>, <vertical coordinate>
This command designates the location of the cursor.

**LOG**

LOG (<numeric expression>)
Returns the natural logarithm of a number.

**LPOS**

LPOS (<numeric expression>)
Determines the current column position of the printer head.

# M

**MAXFILES**

MAXFILES = <number of file(s)>
Establishes the maximum number of files that can be opened.

**MENU**

MENU
Terminates BASIC and returns to MENU.

**MERGE**

MERGE "{<external device name>:} <file name>"
Used to merge two programs together.

**MID$**

MID$ (<character string>, <numeric expression 1>
{, <numeric expression 2>})
Returns a specified number of characters from the specified position within a string.

**MOD**

<numeric expression 1> MOD <numeric expression 2>
Provides the remainder of an arithmetic expression.

**MOTOR**

MOTOR <switch>
Controls the ON and OFF functions of the motor that drives the cassette recorder

# N

**NAME**

NAME "<old file name>"AS "<new file name>"
Renames files in the RAM.

**NEW**

NEW
This command clears the RAM of the current program and variables. It is used before writing a new program.

**NOT**

NOT <operand>
A logical operator used to test multiple relations.

# O

**ON...GOTO/ON...GOSUB**

ON <numeric variable> [GOTO / GOSUB] <line number>
{, <line number list>}
Branches to one of several specified lines/subroutines based on the evaluation of the statement.

**ON COM GOSUB**

ON COM GOSUB <line number>
Branches to the designated line number of a routine used to perform communications interrupt processing.

**ON ERROR GOTO ~ RESUME**

ON ERROR GOTO [<line number> / <0>]
Specifies an error subroutine used for trappable errors.

**OPEN**

OPEN "{<external device name>:} <file name>" for
[input / output / append] as {#} <file number>
Opens a file for input or output.

**OPEN"COM"**

OPEN"COM:{<CPBSXS>} "for [input / output] as {#}
<file number>

Opens up the RS-232C circuit.

**OR**

<operand 1> OR <operand 2>

A logical operator used to test multiple relations.

**OUT**

OUT<port number>,<data>

Sends data to a designated output port.

# P

**PEEK**

PEEK (<address>)

Loads the contents of a designated memory location.

**POKE**

POKE <address>,<data>

Writes data to the specified memory address.

**POS**

POS (<expression>)

Determines the current cursor column position.

**POWER**

POWER [<timer> / OFF / CONT] {,RESUME}

Sets or resets the automatic shut-off function of the PC-8300.

**PRESET**

PRESET (<horizontal coordinates>, <vertical coordinate> {,<function code>})

Resets dots on the LCD screen at the designated coordinates.

**PRINT/LPRINT**

[PRINT / LPRINT] {"} {<expression>...} {"}

PRINT outputs information to the display screen. LPRINT outputs information to a parallel printer.

**PRINT USING/LPRINT USING**

[PRINT / LPRINT] USING <formatting string>;<numeric expression>
{[, / ;] <numeric expression list>}

PRINT USING outputs data formatted as specified to the display screen. LPRINT USING outputs data formatted as specified to the parallel printer.

**PSET**

PSET (<horizontal coordinate>, <vertical coordinate> {,<function code>})

Sets dots on the LCD screen at the designated coordinates.

# R

**READ**

READ <variable list>

Used to read a value from a data statement and assign data to a variable.

**REM**

[REM / '] {<remark>}

Used to put non-executable lines such as remarks or comments in a program.

**RENUM**

RENUM {<new line number>} {,<old line number>} {,<increment>}

Renumbers the lines of a program.

**RESTORE**

RESTORE {<line number>}

Sets the data pointer back to the start of the data; used when the data needs to be read from its start again.

**RESUME**

RESUME [<0> / <NEXT> / <line number>]

Used to resume program execution after performing an error processing routine.

**RETURN**

RETURN {<line number>}

Returns control back to the main program, after execution of the subroutine which contains this command. Control is returned to the first statement which follows the GOSUB statement in the main program.

**RIGHT$**
RIGHT$ (<character string>,<numeric expression>)
Returns the designated right portion of a string.

**RND**
RND (<numeric expression>)
Generates a random number with a value between 0 and 1.

**RUN**
RUN
This statement will cause a program that is in memory to be executed.

RUN [<line number>]
This statement will cause a program that is in memory to be executed, starting at the specified line number.

RUN "[<device name>:] <program name>" [,R]
This statement will cause a program to be loaded into memory from an external device, and to be executed.

# S

**SAVE**
SAVE "[<external device name>:] <file name>" [,A]
Stores a program currently in the memory into RAM or to an external device.

**SCREEN**
SCREEN 0, <function key display switch>
This statement establishes the display mode.

**SGN**
SGN (<numeric expression>)
Determines the sign (+ or −) of a number.

**SIN**
SIN (<numeric expression>)
Returns the sine of an angle.

**SOUND**
SOUND <tone>,<length>
This command produces a sound, as designated.

**SPACE$**
SPACE$ (<numeric expression>)
Used in spacing output for reports and forms.

**SQR**
SQR (<numeric expression>)
Returns the square root of a number.

**STOP**
STOP
Halts program execution, but leaves files intact.

**STR$**
STR$ (<numeric expression>)
Converts a numeric value to a numeric string.

**STRING$**
STRING$ (<numeric expression> [<character string> / <ASCII code>])
Repeats the designated string the specified number of times.

# T

**TAB**
TAB (<numeric expression>)
Used to horizontally space data to be printed or displayed.

**TAN**
TAN (<numeric expression>)
Returns the tangent of an angle.

**TIME$**
TIME$ = "<hour>:<minute>:<second>"
Used to set the current time in the format "hh:mm:ss"

# V

**VAL**
VAL (<numeric string>)
Returns the numeric value of a numeric string.

# X

**XOR**
<operand 1> XOR <operand 2>
A logical operator used to test multiple relations.

## RESERVED WORDS

ABS
AND
ASC
ATN
BEEP
BLOAD
BLOAD?
BSAVE
CDBL
CHR$
CINT
CLEAR
CLOAD
CLOAD?
CLOSE
CLS
COM
CONT
COS
CSAVE
CSNG
CSRLIN
DATA
DATE$
DEFINT
DEFDBL
DEFSNG
DEFSTR
DIM
EDIT
ELSE
END
EOF
EQV
ERL
ERR
ERROR
EXEC
EXP
FILES
FIX
FOR
FRE
GOSUB
GOTO
IF
IMP
INKEY$
INP
INPUT
INPUT$
INPUT#
INSTR
INT
KEY
KILL
LEFT$
LEN
LET
LINE
LIST
LLIST
LOAD
LOCATE
LOG
LPOS
LPRINT
MAXFILES
MENU
MERGE
MID$
MOD
MOTOR
NAME
NEW
NEXT
NOT
OFF
ON
OPEN
OR
OUT
PEEK
POKE
POS
POWER
PRESET
PRINT
PSET
READ
REM
RENUM
RESTORE
RESUME
RETURN
RIGHT$
RND
RUN

## RESERVED WORDS (continued)

SAVE
SCREEN
SGN
SIN
SOUND
SPACE$
SQR
STEP
STOP
STR$
STRING$
TAB
TAN
THEN
TIME$
TO
USING
VAL
XOR

# ERROR CODES

| Error Message | Code | N-BASIC Message | Meaning |
|---|---|---|---|
| ?AC Error | 53 | File Already Open | That file is already open. |
| ?BN Error | 51 | Bad file Number | The file number is incorrect. |
| ?BC Error | 23 | Communcation buffer overflow (Buffer Overflow) | The nput buffer has overlowed. |
| ?BS Error | 9 | Subscript out of range (Bad Subscript) | Array subscript is incorrect. |
| ?CF Error | 58 | File not open (Closed File) | The file is not yet open. |
| ?CN Error | 17 | Can't ContiNue | Program execution cannot be resumed by means of a CONT command. |
| ?DD Error | 10 | Duplicate Definition | The same array has been declared twice. |
| ?DS Error | 56 | Direct Statement in file | An ASCII format won't load. |
| ?DL Error | 25 | Device Unavailable | The designated device is not accessible. |
| ?EF Error | 54 | Input past end (End of File) | No more data in the file. |
| ?FC Error | 5 | Illegal Function Call | Attempts to use Commands or Funcions are incorrect. |
| ?FF Error | 52 | File not Found | The designated file cannot be located. |
| ?FL Error | 57 | Filing Limit | There are too many files. |
| ?ID Error | 12 | Illegal Direct | The specified command cannot be used in the direct mode. |
| ?IE Error | 50 | Internal Error | An error within BASIC. |
| ?IO Error | 24 | I/O error | An error during input or output. |
| ?LS Error | 15 | String too long (Long String) | Over 255 characters in a string variable. |

| Error Message | Code | N-BASIC Message | Meaning |
|---|---|---|---|
| ?MO Error | 22 | Missing Operand | A required parameter is missing |
| ?NF Error | 1 | NEXT without FOR | There is no FOR statement to match the NEXT statement. |
| ?NM Error | 55 | Bad file name (File Name Mismatch) | The name of the file is inappropriate for the operation attempted. |
| ?NR Error | 19 | No RESUME | There is no RESUME statement present in an error processing routine. |
| ?OD Error | 4 | Out of Data | There is no more data. |
| ?OM Error | 7 | Out of Memory | There is not enough memory. |
| ?OS Error | 14 | Out of String space | The memory region available for string storage is inadequate. |
| ?OV Error | 6 | Overflow | A numeric value is too big. |
| ?PC Error | 59 | PC-8001A Command | This command is for use only on the PC-8001A. |
| ?RG Error | 3 | RETURN without GOSUB | A RETURN statement is present without a matching GOSUB statement. |
| ?RW Error | 20 | RESUME without Error | A RESUME is met before an error processing routine is entered. |
| ?SN Error | 2 | SyNtax error | The grammar of a statement is incorrect. |
| ?ST Error | 16 | String formula Too complex | The string formula is too complex. |
| ?TM Error | 13 | Type Mismatch | The types of variables and integers are inconsistent. |
| ?UE Error | 21 | Unprintable Error | An error that has not been designated in a message occurred. |
| ?UF Error | 18 | Undefined user Function | An undefined user function has been read. |

| Error Message | Code | N-BASIC Message | Meaning |
|---|---|---|---|
| ?UL Error | 8 | Undefined Line number | A designated line has not been defined. |
| ?/0 Error | 11 | Division by Zero | A division by 0 is attempted. |

# CONTROL CODES

| OPERATION | CHARACTER CODE | FUNCTION |
|---|---|---|
| CTRL C or STOP | 3 | Interrupts program execution |
| CTRL E | 5 | Deletes after the cursor position to the end of the file |
| CTRL G | 7 | Sounds bell |
| CTRL H or DEL BS | 8 | Deletes one character to the left of the cursor |
| CTRL I or TAB | 9 | Moves the cursor to the next tab setting |
| CTRL K | 11 | Moves the cursor to the home position |
| CTRL L | 12 | Clears the screen |
| CTRL M or ↵ | 13 | Moves the cursor to the biginning of a new line |
| CTRL N | 14 | Shift OUT* |
| CTRL O | 15 | Shift IN* |
| CTRL Q | 17 | Authorizes reopening on transmissions (XON)* |
| CTRL S | 19 | Requests an interrupt of transmissions (XOFF)* |
| ESC | 27 | Begins an ESCape sequence |
| ◁ | 28 | Moves the cursor one character to the right |
| ▷ | 29 | Moves the cursor one character to the left |
| ▽ | 30 | Moves the cursor up one line |
| △ | 31 | Moves the cursor down one line |

# ESCAPE SEQUENCES

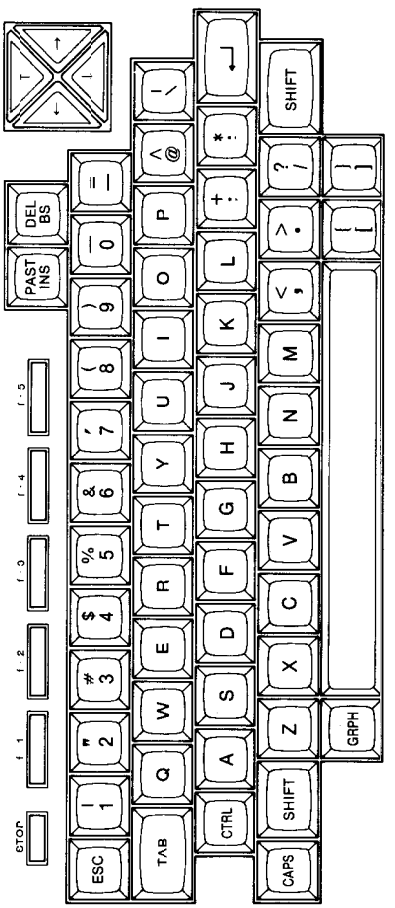
| ESC + | CHARACTER CODE | FUNCTION |
|---|---|---|
| A | 27,65 | Moves the cursor one line up |
| B | 27,66 | Moves the cursor one one line down |
| C | 27,67 | Moves the cursor one character (one column) to the right |
| D | 27,68 | Moves the cursor one character (one column) to the left |
| E | 27,29 | Clears screen and moves the cursor to the top left corner of the screen (the home position) |
| J | 27,74 | Erases characters from the cursor position to the end of the display |
| K | 27,75 | Erases characters from the cursor position to the right end of the current line |
| L | 27,76 | Insert a line |
| M | 27,77 | Deletes the line where the cursor is located |
| T | 27,84 | Displays Function Keys |
| U | 27,85 | Erases Function Keys display |
| V | 27,86 | Inhibits scrolling (freezes the display) |
| W | 27,87 | Scrolling is permitted |
| Y<y><x> | * | Moves the cursor to the designated location |
| j | 27,106 | Clears the screen |
| p | 27,112 | Changes the screen to reverse display |
| q | 27,113 | Restores display to normal (back from reverse display) |

* ESC + Y<y><x>
The cursor position is designated by the vertical and horizontal coordinates (y and x) respectively.

Characters and capital letters beginning at ASCII code 32 are used in the coordinate designation. A space corresponds to 0, the excaimation point (!) corresponds to 1, etc. Refer to an ASCII code chart for the complete list.

To move the cursor to the home position (coordinate 0,0), for example, you would input the following string:

ESC Y SPACE SPACE




## Keyboard

# CHARACTER CODE CHART

| Lower 4 bits | DECIMAL | HEX / BINARY | Higher 4 bits: 0 | 16 | 32 | 48 | 64 | 80 | 96 |
|---|---|---|---|---|---|---|---|---|---|
| | | | 0 0000 | 1 0001 | 2 0010 | 3 0011 | 4 0100 | 5 0101 | 6 0110 |
| | 0 | C 0000 | C/@ | C/P | (SPACE) | 0 | @ | P | ` |
| | 1 | 1 0001 | C/A S/← | C/Q C/← | ! | 1 | A | Q | a |
| | 2 | 2 0010 | C/B S/↑ | C/R INS C/→ | " | 2 | B | R | b |
| | 3 | 3 0011 | C/C STOP | C/S | # | 3 | C | S | c |
| | 4 | 4 0100 | C/D | C/T S/↑ | $ | 4 | D | T | d |
| | 5 | 5 0101 | C/E | C/U | % | 5 | E | U | e |
| | 6 | 6 0110 | C/F S/← | C/V | & | 6 | F | V | f |
| | 7 | 7 0111 | C/G | C/W C/↑ | ' | 7 | G | W | g |
| | 8 | 8 1000 | C/H BS | C/X | ( | 8 | H | X | h |
| | 9 | 9 1001 | C/I | C/Y | ) | 9 | I | Y | i |
| | 10 | A 1010 | C/J | C/Z C/↑ | * | : | J | Z | j |
| | 11 | B 1011 | C/K | ESC | + | ; | K | [ | k |
| | 12 | C 1100 | C/L | → | , | < | L | \ | l |
| | 13 | D 1101 | C/M ↵ | ← | — | = | M | ] | m |
| | 14 | E 1110 | C/N | ↑ | . | > | N | ^ | n |
| | 15 | F 1111 | C/O | ↓ | / | ? | O | _ | o |

| Higher 4 bits: 112 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |
|---|---|---|---|---|---|---|---|---|
| | 7 0111 | 8 1000 | 9 1001 | A 1010 | B 1011 | C 1100 | D 1101 | E 1110 | F 1111 |
| p | G/Z ◀ | G/Q | GS/Z | GS/Q | | | | |
| q | G/X ↵ | G/W | GS/X | GS/W | | | | |
| r | G/C ▒ | G/E | GS/C | GS/E | | | | |
| s | G/V | G/R | GS/V | GS/R | | | | |
| t | G/B | G/T | GS/B | GS/T | | | | |
| u | G/N | G/Y | GS/N | GS/Y | | | | |
| v | G/M | G/U | GS/M | GS/U | | | | |
| w | G/L | G/I | GS/L | GS/I | | | | |
| x | G/A | G/O | GS/A | GS/O | | | | |
| y | G/S | G/P | GS/S | GS/P | | | | |
| z | G/D | G/@ | GS/D | GS/@ | | | | |
| { | G/F | G/^ | GS/F | GS/[ | | | | |
| \| | G/G | G/, | GS/G | GS/< | | | | |
| } | G/H | G/- | GS/H | GS/> | | | | |
| ~ | G/J | G/. | GS/J | GS/? | | | | |
| (DEL) | G/K | G/] | GS/K | GS/} | | | | |

**Notes:** 1. C/r means hold CTRL while pressing "r"
2. S/r means hold SHIFT while pressing "r"
3. G/r means hold GRPH while pressing "r"
4. GS/r means hold SHIFT and GRPH while pressing "r"

**Example:** To find the character code of "A", add 1 to DECimal 64, or to HEX 40.
Thus the character code of "A" is 63 DEC or 41 HEX (01000001 BINARY)

**Code:**

- 00H — 1FH: Unique code that cannot be output as characters (See p.21 Control Codes)
- 83H — DFH: User-defined characters (Can be input from the keyboard)
- E0H — FFH: User-defined characters (Can be output by using the CHR$ function)