IBM

*Personal Computer*

# BASIC

## Quick Reference

6361724

*Personal Computer*

# BASIC

## Quick Reference

First Edition (May, 1984)

# How to Use This Book

The *BASIC Quick Reference* is intended to remind you of the purpose and syntax of the BASIC commands, statements, functions, and variables.

The entries are categorized by task. An explanation of each example is supplied.

For more detailed information, see the *BASIC Reference.*

# Contents

# General

## ASC

```
1Ø X$="TEST"
2Ø PRINT ASC(X$)
```

The ASCII code of the first character of X$ is printed. For the capital letter T, the value is 84.

## BEEP

```
1Ø IF X<2Ø THEN BEEP
```

If X is less than 2Ø the speaker emits a single beep.

## BLOAD

```
1Ø DEF SEG=&HB8ØØ
2Ø BLOAD "PICTURE",Ø
```

Loads a file called PICTURE that has been previously BSAVEd into memory at offset=Ø of the segment beginning at HB8ØØØ.

## BSAVE

```
1Ø DEF SEG=&HB8ØØ
2Ø BSAVE "PICTURE",Ø,&H4ØØØ
```

Saves an image of the memory area in segment HB8ØØØ starting at offset=Ø and ending at offset=4ØØØ. The image is written to a disk file called PICTURE.

## CALL

```
1Ø DEF SEG=&H8ØØØ
2Ø OFS=Ø
3Ø CALL OFS(A,B$,C)
```

Calls a machine language subroutine that begins at offset=Ø of segment H8ØØØ and passes variables A, B$, and C to the called routine.

## CHAIN

```
1Ø CHAIN "A:PROG1",1Ø,ALL
```

Transfers control to PROG1, which begins execution at line 1Ø, and passes ALL variables to PROG1.

## CLEAR

```
1Ø CLEAR 32768,2ØØØ
```

CLEARs all memory used for data without erasing current program; sets maximum workspace=32768 bytes; sets stack size=2ØØØ bytes.

## CLS

```
1Ø CLS
```

Clears the screen; homes the cursor.

## COLOR (Text Mode)

```
1Ø COLOR 1,Ø
```

IBM monochrome display produces green underlined characters on a black background. Color display produces blue foreground characters on a black background.

## COMMON

```
1ØØ COMMON A, B(), C$
11Ø CHAIN "A:PROG3"
```

Passes variables A, B(), and C$ to PROG3.

## CSRLIN

```
1Ø Y=CSRLIN
```

Stores the value of the current cursor line number.

## DATA

```
1Ø DATA 3.Ø5, 5.89, 23.Ø8
```

Stores a list of constant DATA to be accessed by a READ statement.

## DATE$

```
1Ø DATE$="1Ø/31/83"
2Ø PRINT DATE$
```

Can either set date or retrieve date.

## DEF FN

```
1Ø PI=3.141593
2Ø DEF FNAR(R)=PI*RA2
3Ø INPUT "Radius? ", RAD
4Ø PRINT "Area=";FNAR(RAD)
```

Defines a function written by you to calculate the area of a circle.

## DEF SEG

```
1Ø DEF SEG=&HB8ØØ
```

Defines the segment in memory that is currently being accessed. HB8ØØØ begins the area of the color screen buffer.

## DEFtype

```
1Ø DEFINT A-D
2Ø DEFSING F-L,X
3Ø DEFSTR M-W
```

Declares variable types by the first letter of the variable name.

## DEF USR

```
1Ø CLEAR,&H8ØØØ
2Ø DEF SEG
3Ø DEF USR2=&H8ØØØ
4Ø X=USR2 (Y+2)
```

Defines a machine language subroutine called USR(2) that begins at offset=24ØØØ of the current segment.

## DIM

```
1Ø DIM VAR(2Ø)
```

Allocates space in memory for up to 21 subscripts for the variable VAR.

## END

```
1Ø END
```

Ends program execution.

## ENVIRON

```
1Ø ENVIRON "PATH=C:\"
```

Sets a path parameter in the BASIC environment table to the root directory on the C: drive.

## ENVIRON$

```
1Ø PRINT ENVIRON$("PATH")
```

Displays the current contents of the PATH parameter in the BASIC environment table.

## EOF

```
1Ø IF EOF THEN END
```

Checks for an end-of-file condition when reading from a sequential file.

## ERASE

```
1Ø ERASE TEST1,TEST2
```

Erases arrays TEST1 and TEST2 from memory.

## ERDEV

```
1Ø PRINT ERDEV
```

Displays a number containing a device error code.

## ERDEV$

```
1Ø PRINT ERDEV$
```

Displays the name of the device causing an error.

## ERR

```
1Ø IF ERR=27 THEN GOTO 1ØØ
```

If error code=27 (Out of paper) then GOTO the error-handling routine.

## ERL

```
1Ø IF ERL=25Ø THEN RESUME
```

If the error occurred in line 25Ø, ignore the error.

## ERROR

```
2Ø ON ERROR GOTO 4Ø
3Ø IF B>5ØØ THEN ERROR 21Ø
4Ø IF ERR=21Ø THEN END
```

Defines a new error code to be handled by your program.

## FOR ... NEXT

```
1Ø FOR I=1 TO 5ØØ STEP 2
2Ø NEXT I
```

Advances a counter two steps at a time each time NEXT is executed until the count reaches 5ØØ.

## FRE

```
PRINT FRE(Ø)
```

Displays the number of unused bytes of memory.

## GOSUB ... RETURN

```
2Ø GOSUB 4Ø
3Ø PRINT"CHECK":END
4Ø PRINT"DOUBLE":RETURN
```

Goes to a subroutine at line 4Ø; then returns to the statement following the GOSUB.

## GOTO

```
1Ø GOTO 3Ø
2Ø PRINT"ERROR"
3Ø PRINT"CONTINUE"
4Ø PRINT"PROGRAM"
```

Goes to line 3Ø and continues execution from there.

## IF

```
1Ø IF A=2Ø THEN GOTO 3Ø ELSE STOP
```

If A=2Ø is true, then branch to line 3Ø; otherwise, stop execution.

## INKEY$

```
1Ø PRINT"PRESS ANY KEY"
2Ø A$=INKEY$
3Ø IF A$="" THEN 1Ø
```

Reads a character from the keyboard and assigns it to a variable called A$. If no key is pressed, return to line 1Ø and loop until a key is pressed.

## INP

```
1Ø A=INP(255)
```

Reads input from machine port number 255 and assigns it to variable A.

## INPUT

```
1Ø INPUT "WHAT RADIUS";R
```

Waits for keyboard input; displays the quoted phrase; assigns the keyboard input to variable R.

## INPUT$

```
1Ø X$=INPUT$(5)
```

Reads a string of 5 characters from the keyboard and assigns them to X$.

## IOCTL

```
1Ø OPEN "O", #1,"LPT1:"
2Ø IOCTL #1, "PL56"
```

Passes control data (PL56) that sets page length to open device #1.

## IOCTL$

```
1Ø IF IOCTL$(1)="56" THEN PRINT "OK"
```

Reads the control data string on open device #1.

## KEY

```
10 KEY 6, "PRINT"
```

Programs function key F6 to display the word PRINT.

## KEY(n)

```
10 KEY(10) STOP
```

Disables trapping of function key 10.

## LET

```
10 LET Y=35
```

Assigns the value 35 to the variable Y.

## LINE INPUT

```
10 LINE INPUT "Address? ";C$
```

Reads an entire line of input, including delimiters, into variable C$.

## LOCATE

```
10 LOCATE 10,15
```

Positions the cursor at row 10, column 15.

## LPOS

```
10 PRINT LPOS(0)
```

Displays the position of the print head in the print buffer.

## LPRINT

```
10 LPRINT "TESTING"
```

Prints the word TESTING on the printer.

## LPRINT USING

```
10 LPRINT USING "###.##";456.7832
```

Prints the expression 456.7832 on the printer using the quoted format. Printed copy reads 456.78.

## MERGE

```
10 MERGE "B:PROG2"
```

Merges the lines of PROG2 with the lines of the current program in memory. Duplicate line numbers are replaced by the lines in PROG2.

## MOTOR

```
1Ø MOTOR 1
```

Turns on the motor of the cassette player.

## ON ERROR

```
1Ø ON ERROR GOTO 5ØØ
```

When any error occurs, branch to line 5ØØ.

## ON...GOSUB

```
1Ø ON NUMBER GOSUB 22Ø, 45Ø, 13Ø
```

If NUMBER=1, GOSUB 22Ø; If NUMBER=2, GOSUB 45Ø; If NUMBER=3, GOSUB 13Ø.

## ON... GOTO

```
1Ø ON NUMBER GOTO 22Ø, 45Ø, 13Ø
```

Similar to ON GOSUB.

## ON KEY(n)

```
1Ø ON KEY(1Ø) GOSUB 1ØØØ
```

When function key 1Ø is pressed, branch to line 1ØØØ.

## ON PEN

```
1Ø ON PEN GOSUB 2ØØØ
```

When light pen is detected, branch to line 2ØØØ.

## ON PLAY(n)

```
1Ø ON PLAY(5) GOSUB 5ØØ
```

When music is playing in background and 5 notes remain in the buffer, branch to line 5ØØ.

## ON STRIG(n)

```
1Ø ON STRIG(2) GOSUB 1ØØØ
```

When joystick trigger button is pressed, branch to line 1ØØØ.

## ON TIMER

```
1Ø ON TIMER(3Ø) GOSUB 5ØØ
```

When 3Ø seconds have elapsed, branch to line 5ØØ.

## OPTION BASE

```
1Ø OPTION BASE 1
```

Sets the subscript of the lowest-numbered array element as 1.

## OUT

```
1Ø OUT 98Ø,2
```

Sends a value of 2 to machine port 98Ø.

## PEEK

```
1Ø PEEK (&H41Ø)
```

Reads the value stored in memory location H41Ø.

## PEN

```
1Ø PRINT PEN(6)
```

Displays which row light pen was on when activated.

## PLAY

```
1Ø PLAY "XA$;"
```

Plays music as instructed by contents of string XA$.

## PLAY(n)

```
1Ø PRINT PLAY(Ø)
```

Displays the number of notes remaining in the music background buffer.

## POKE

```
1Ø SCREEN 1
2Ø DEF SEG
3Ø POKE &HFE,2
```

Writes the value 2 into memory location HFE.

## POS

```
1Ø IF POS(Ø)>6Ø
   THEN PRINT CHR$(13)
```

If the cursor column position is beyond 6Ø, then perform a carriage return.

## PRINT

```
1Ø PRINT "ANYTHING"
```

Displays the word ANYTHING on the screen.

## PRINT USING

```
1Ø PRINT USING "###.###";456.2341
```

Displays the expression 456.2341 on the screen using the quoted format. Screen displays 456.23.

## RANDOMIZE

```
1Ø RANDOMIZE (2ØØØØ)
```

Reseeds the random number generator with the number 2ØØØØ to produce a new sequence of numbers.

## READ

```
1Ø READ A(I)
```

Reads a value from a DATA statement and assigns it to the variable A(I).

## RESTORE

```
1Ø RESTORE
```

Causes the next READ statement to begin reading at the first DATA statement in the program.

## RESUME

```
1Ø RESUME 12Ø
```

Following an error recovery, causes execution to resume at line 12Ø.

## RETURN

```
1Ø RETURN
```

Causes program to return to the line following the GOSUB that initiated the branch to this subroutine.

## RND

```
1Ø Y=INT(RND*7)
```

Yields random integers in the range Ø to 6.

## SCREEN

```
1Ø PRINT SCREEN(5,1Ø)
```

Displays the ASCII code for the character on the screen at row 5, column 1Ø.

## SHELL

```
1Ø SHELL
```

Loads DOS from BASIC; current program remains in memory.

## SOUND

```
1Ø SOUND 22Ø.ØØØ, 18
```

Produces a sound of 22Ø cps for a duration of 18 clock ticks (1 second).

### SPC

```
1Ø PRINT SPC(2Ø)
```

Prints 2Ø spaces.

### STICK

```
1Ø P=STICK(Ø)
```

Returns x-coordinate of joystick A and assigns it to variable P.

### STRIG

```
1Ø STRIG ON
```

Enables status of joystick buttons to be read.

### STRIG(n)

```
1Ø STRIG(2) ON
```

Enables trapping of joystick button B1 by the ON STRIG(n) statement.

### SWAP

```
1Ø SWAP X$,Y$
```

Exchanges the values of variables X$ and Y$.

### SYSTEM

```
1Ø SYSTEM
```

Returns to DOS.

### TAB

```
1Ø PRINT TAB(25)
```

Moves cursor to position 25.

### TIME$

```
1Ø TIME$="1Ø:23:ØØ"
2Ø PRINT TIME$
```

Either sets or retrieves the current time.

### TIMER

```
1Ø PRINT TIMER
```

Displays the number of seconds elapsed since midnight or since system reset.

## USR

```
1Ø DEF USRØ=&HFØØØ
2Ø C=USRØ(B/2)
```

Calls the machine language subroutine USRØ and supplies the argument (B/2).

## VARPTR

```
1Ø PRINT VARPTR(X)
```

Displays the memory location of the variable X.

## VARPTR$

```
1Ø PLAY "X"+VARPTR$(A$)
```

Returns a 3-byte string of the memory location of the variable A$. Is equivalent to PLAY "XA$;".

## WAIT

```
1Ø WAIT 32,2
```

Causes program to wait until port 32 receives a bit-value of 1 in the second bit position.

## WHILE and WEND

```
1Ø X=Ø
2Ø WHILE X=Ø
3Ø INPUT X
4Ø S=S+X
5Ø WEND
6Ø PRINT "SUM=";S
```

Causes the statements between WHILE and WEND to loop until a value for X is input.

## WIDTH

```
1Ø WIDTH 4Ø
```

Sets the screen width to 4Ø characters per line.

## WRITE

```
1Ø A=8Ø:B=9Ø:C$="THE END"
2Ø WRITE A,B,C$
```

Similar to PRINT, but inserts commas and quotes. Example will display 8Ø,9Ø,"THE END"

# Communications

### COM(n)

```
10 COM(1) ON
```

Enables trapping of communications adapter #1.

### ON COM(n)

```
10 ON COM(1) GOSUB 1000
```

When activity is detected in communications adapter #1 branch to line 1000.

### OPEN "COM...

```
10 OPEN "COM:" AS 1
```

Opens communications adapter #1 for communications.

# Conversions

### CDBL

```
10 PRINT CDBL(A)
```

Converts the value of A to a double-precision number.

### CHR$

```
10 PRINT CHR$(66)
```

Converts the value 66 to the equivalent ASCII code character which is a capital letter B.

### CINT

```
10 PRINT CINT(45.67)
```

Converts the value 45.67 to the integer value 46.

### CSNG

```
10 PRINT CSNG(45.34536789)
```

Converts the double-precision value 45.34536789 to the single-precision value 45.3453.

### CVD

```
10 Y=CVD(N$)
```

Converts the 8-byte string variable N$ to a double-precision numeric variable.

### CVI

```
1Ø X=CVI(X$)
```

Converts the 2-byte string variable X$ to an integer variable.

### CVS

```
1Ø Y=CVS(Y$)
```

Converts the 4-byte string variable Y$ to a single-precision numeric variable.

### HEX$

```
1Ø H$=HEX$(16)
```

Converts the decimal value 16 to the hexadecimal value 1Ø.

### MKD$

```
1Ø D$=MKD$(AMT)
```

Converts the value of the single-precision variable AMT to a string variable.

### MKI$

```
1Ø R$=MKI$(STEP)
```

Converts the value of the integer variable STEP to a string variable.

### MKS$

```
1Ø K$=MKS$(BALANCE)
```

Converts the single-precision variable BALANCE to a string variable.

### OCT$

```
1Ø PRINT OCT$(24)
```

Converts the decimal value (24) to the equivalent octal value (3Ø).

# Development

### AUTO

```
1Ø AUTO 1ØØ,5Ø
```

Automatically numbers each new program line using steps of 5Ø and beginning with line 1ØØ.

## CONT

```
1Ø CONT
```

Continues program execution after a pause.

## DELETE

```
1Ø DELETE 4Ø-1ØØ
```

Erases lines 4Ø-1ØØ.

## EDIT

```
1Ø EDIT 35
```

Displays line 35 for editing.

## LIST

```
1Ø LIST 2Ø-2ØØ
```

Displays a listing of lines 2Ø-2ØØ of the program in memory.

## LLIST

```
1Ø LLIST
```

Prints a complete program listing.

## LOAD

```
1Ø LOAD "PROG3"
```

Loads the program PROG3 into memory.

## NAME

```
1Ø NAME "ACCTS.BAS" AS "LEDGER.BAS"
```

Renames a file.

## NEW

```
1Ø NEW
```

Clears current program and all variables from memory

## REM

```
1Ø REM ignore this statement
```

Inserts a nonexecutable remark.

## RENUM

```
1Ø RENUM 1ØØØ,1Ø
```

Renumber the entire program to start with a line number of 1ØØØ with each line incremented by 1Ø.

### RUN

```
1Ø RUN
```

Executes the current program in memory.

### SAVE

```
1Ø SAVE "PROG4",A
```

Saves the program in memory using the name PROG4 as an ASCII file.

### STOP

```
1Ø STOP
```

Halts program execution.

### TRON and TROFF

```
1Ø TRON
2Ø REM code to be traced
3Ø TROFF
```

Turns program trace on and off.

## Files

### CHDIR

```
1Ø CHDIR "\"
```

Changes directory to the root directory.

### CLOSE

```
1Ø CLOSE
```

Closes all open files and devices.

### FIELD

```
1Ø FIELD 1, 2Ø AS N$,
   3Ø AS A$
```

Allocates space for variables in random file buffer #1.

### FILES

```
1Ø FILES "B:*.*"
```

Displays all files on drive B.

### GET

```
10 OPEN "A:CUST" AS #1
20 GET 1
```

Reads a record from random file #1.

### INPUT#

```
10 OPEN "O",#1,"DATA"
20 INPUT#1,INFO$
```

Reads the variable INFO$ from the open file.

### KILL

```
10 KILL"B:DATA.BAS"
```

Erases the file on drive B called DATA.BAS.

### LINE INPUT#

```
10 LINE INPUT#1, ADDRESS$
```

Reads an entire line from the open file and assigns it to the variable ADDRESS$.

### LOC

```
10 PRINT LOC(1)
```

Displays the current position in the file opened as #1.

### LOF

```
10 PRINT LOF(1)
```

Displays length of the file opened as #1.

### LSET and RSET

```
10 LSET N$=NA$
```

Moves the contents of NA$ into the random file buffer named N$ and left-justifies that field.

### MKDIR

```
10 MKDIR "SALES"
```

Creates a directory called SALES.

### OPEN

```
10 OPEN "O",#1,"DATA"
```

Opens a device or file called DATA to receive output as file #1.

### PRINT #

```
1Ø PRINT #1,DATE$;TIME$
```

Writes the variables DATE$ and TIME$ to the sequential file open as #1.

### PRINT # USING

```
1Ø PRINT #1 USING "###.##";AMOUNT
```

Writes the value of the variable AMOUNT to the sequential file open as #1 using the quoted format.

### PUT

```
1Ø PUT #1
```

Writes a record that has been LSET or RSET into random buffer #1, a random file.

### RESET

```
1Ø RESET
```

Closes all open disk files.

### RMDIR

```
1Ø RMDIR "SALES"
```

Removes the directory called SALES.

### WRITE #

```
1Ø WRITE #1,NAME$,AGE$
```

Writes the variables NAME$ and AGE$ to the sequential file opened as #1. Automatically inserts commas between items and quotes around strings.

## Graphics

### CIRCLE

```
1Ø CIRCLE (16Ø, 1ØØ),5Ø
```

Draws a circle with center at X=16Ø, Y=1ØØ and radius=5Ø.

### COLOR

```
1Ø COLOR 9,Ø
```

Sets background color to light blue and selects palette Ø in SCREEN 1.

### DRAW

```
1Ø SCREEN 1
2Ø DRAW "U2Ø R2Ø D2Ø L2Ø
```

Draws a box 2Ø units wide and high.

### GET

```
1Ø GET (1Ø,1Ø)-(2Ø,2Ø), "PICTURE"
```

Saves the contents of the screen within the rectangle whose opposite corners are (1Ø,1Ø) and (2Ø,2Ø) into an array named PICTURE.

### LINE

```
1Ø LINE (1,1)-(5Ø,5Ø),2,B
```

Draws a box using color attribute 2 whose opposite corners are (1,1) and (5Ø,5Ø).

### PAINT

```
1Ø PAINT (15,15),2
```

Fills the interior of the box in the Line example with color attribute 2 starting at point (15,15).

### PMAP

```
1Ø WINDOW (-1,-1)-(1,1)
2Ø PMAP(1,Ø):PMAP(Ø,1)
```

Translates the center point of the WINDOW from world coordinates of (Ø,Ø) to physical coordinates of (16Ø,1ØØ).

### POINT

```
1Ø C=POINT(15,15)
```

Reads the color attribute of the point at screen location (15,15) and assigns it to variable C.

### PSET and PRESET

```
1Ø PSET (15,15)
2Ø PRESET (15,15)
```

PSET draws a point at coordinates (15,15) in the foreground color. PRESET removes the same point.

### PUT

```
10 PUT (40,40),"PICTURE"
```

Takes the bit image that was saved with GET in the array called PICTURE and puts it on the screen with the upper left corner of the image at location (40,40).

### SCREEN

```
10 SCREEN 1,0
```

Switches screen to medium resolution graphics mode and enables color burst.

### VIEW

```
10 VIEW (5,5)-(100,100)
```

Defines a rectangular section of the screen whose opposite corners are (5,5) and (100,100) as a viewport into which the contents of WINDOW are mapped.

### WINDOW

```
10 WINDOW (-6,-6)-(6,6)
20 CIRCLE (4,4),5,1
```

Produces a circle of radius=5 that fills most of the screen because WINDOW redefined the coordinates of the screen to range from only +6 to −6.

## Math Functions

### ABS

```
10 PRINT ABS(7*(-5))
```

Displays the absolute value (35) of the stated expression.

### ATN

```
10 PRINT ATN(5)
```

Displays the arctangent in radians of the number 5.

### COS

```
10 PRINT COS(3.14)
```

Displays the cosine in radians of an angle equal to 3.14 radians.

EXP

```
1Ø PRINT EXP(1)
```

Displays the value of the number e raised to the first power.

FIX

```
1Ø PRINT FIX(45.67)
```

Displays the integer digits (45) of the number 45.67.

INT

```
1Ø PRINT INT(-2.89)
```

Displays the largest integer (−3) that is less than or equal to (−2.89).

LOG

```
1Ø PRINT LOG(45/7)
```

Displays the natural logarithm (1.86Ø752) of the expression (45/7).

SGN

```
1Ø PRINT SGN(X)
```

Displays the sign of the variable X.

SIN

```
1Ø PRINT SIN(3.14)
```

Displays the sine in radians of an angle equal to 3.14 radians.

SQR

```
1Ø PRINT SQR(81)
```

Displays the square root (9) of the value 81.

TAN

```
1Ø PRINT TAN(3.14)
```

Print the tangent in radians of an angle equal to 3.14 radians.

# String Functions

### INSTR

```
10 A$="ABCDEFG"
20 B$="B"
PRINT INSTR(A$,B$)
```

Searches for the first occurrence of B$ within A$ and displays the position (2) where the match begins.

### LEFT$

```
10 A$="BASIC EXAMPLE"
20 PRINT LEFT$(A$,2)
```

Displays the leftmost 2 characters (BA) of the string A$.

### LEN

```
10 X$="IBM PC"
20 PRINT LEN(A$)
```

Displays the length (6) of the string X$.

### MID$

```
10 EX$="QRSTUVW"
20 PRINT MID$(EX$,2,3)
```

Starting with the second character, displays the next 3 characters (RST).

### RIGHT$

```
10 SAM$="SAMPLE"
20 PRINT RIGHT$(SAM$,3)
```

Displays the rightmost 3 characters (PLE) of string SA.

### SPACE$

```
10 SP$=SPACE$(25)
20 PRINT SP$;"TEST"
```

Prints a string of 25 spaces; then prints the word TEST.

### STR$

```
10 PRINT LEN(STR$(321))
```

Treats the numeric expression 321 as a string expression and displays the length.

### STRING$

```
10 X$=STRING$(10,45)
20 PRINT X$;"TEST";X$
```

Displays a string consisting of ASCII character 45 (−) repeated 10 times; then prints the word TEST; then displays 10 more dashes.

### VAL

```
10 PRINT VAL("29 EAST AVE.")
```

Displays the numeric value of the given string (29).

# Error Messages

| Number | Message |
|---|---|
| 1 | NEXT without FOR |
| 2 | Syntax error |
| 3 | RETURN without GOSUB |
| 4 | Out of data |
| 5 | Illegal function call |
| 6 | Overflow |
| 7 | Out of memory |
| 8 | Undefined line number |
| 9 | Subscript out of range |
| 10 | Duplicate definition |
| 11 | Division by zero |
| 12 | Illegal direct |
| 13 | Type mismatch |
| 14 | Out of string space |
| 15 | String too long |
| 16 | String formula too complex |
| 17 | Can't continue |
| 18 | Undefined user function |
| 19 | No RESUME |
| 20 | RESUME without error |
| 22 | Missing operand |
| 23 | Line buffer overflow |
| 24 | Device timeout |
| 25 | Device fault |
| 26 | FOR without NEXT |
| 27 | Out of paper |
| 29 | WHILE without WEND |
| 30 | WEND without WHILE |
| 50 | Field overflow |
| 51 | Internal error |
| 52 | Bad file number |
| 53 | File not found |
| 54 | Bad file mode |
| 55 | File already open |
| 57 | Device I/O error |
| 58 | File already exists |
| 61 | Disk full |
| 62 | Input past end |
| 63 | Bad record number |
| 64 | Bad file name |
| 66 | Direct statement in file |
| 67 | Too many files |
| 68 | Device unavailable |
| 69 | Communication buffer overflow |
| 70 | Disk write protect |
| 71 | Disk not ready |
| 72 | Disk media error |
| 73 | Advanced feature |
| 74 | Rename across disks |
| 75 | Path/file access error |
| 76 | Path not found |
| — | Unprintable error |
| — | Incorrect DOS Version |
| — | You cannot SHELL to BASIC |
| — | Can't continue after SHELL |

# Notes

# Notes

IBM United Kingdom International Products Limited
PO Box 41, North Harbour, Portsmouth PO6 3AU, England

Printed in Great Britain by Ben Johnson & Co. Ltd., York.