The following material is excerpted from:

# *The Microcontroller Idea Book*

## *Circuits, Programs, & Applications*

## *featuring the 8052-BASIC Microcontroller*

by Jan Axelson



ISBN 0-9650819-0-7
Published by Lakeview Research
Distribution by International Thomson Publishing (ITP) in arrangement with
Peer-to-Peer Communications.

For more information, contact:

Lakeview Research
2209 Winnebago St.
Madison, WI 53704
USA

Phone: 608-241-5824
Fax: 608-241-5848
Email: jaxelson@lvr.com
World Wide Web: http://www.lvr.com

# Quick Reference to BASIC-52

This quick reference to the BASIC-52 programming language lists the keywords alphabetically, along with brief descriptions of function and use.

## Conventions

The reference uses the following typographic conventions:

**KEYWORDS** (boldface uppercase)
BASIC-52 keywords

*placeholders* (italics)
Variables, expressions, constants, or other information that you must supply

[*optional items*] (enclosed in square brackets)
Items that are not required

*repeating elements*... (followed by ellipsis (three dots))
You may add more items with the same form as the preceding item.

C = command mode
R = run mode

*variable* **=** *expression* C,R
Assigns a value to a variable

*expression* **=** *expression* C,R
Equivalence test (relational operator)

*expression* **+** *expression* C,R
Add

*expression* **-** *expression* C,R
Subtract

*expression* ***** *expression* C,R
Multiply

*expression* **/** *expression* C,R
Divide

*expression* ****** *expression* C,R
Raises first expression to value of second expression (exponent)

*expression* **<>** *expression* C,R
Inequality test (relational operator)

*expression* **<** *expression* C,R
Less than test (relational operator)

*expression* **>** *expression* C,R
Greater than test (relational operator)

*expression* <= expression C,R
Less than or equal test (relational operator)

*expression* >**=** *expression* C,R
Greater than or equal test (relational operator)

**?**
Same as PRINT

**ABS** (*expression*) C,R
Returns the absolute value of *expression*

*expression* **.AND.** *expression* C,R
Logical AND

**ASC**(*character*) C,R
Returns the value of ASCII character

**ATN**(*expression*) C,R
Returns the arctangent of *expression*

**BAUD** *expression* C,R
Sets the baud rate for LPT (pin 8). For proper operation, XTAL must match the system's crystal frequency.

**CALL** *integer* C,R
Calls an assembly-language routine at the specified address in program memory.

**CBY**(*expression*) C,R
Retrieves the value at *expression* in program, or code, memory.

**CHR**(*expression*) C,R
Converts *expression* to its ASCII character.

**CLEAR** C,R
Sets all variables to 0, resets all stacks and interrupts evoked by BASIC.

**CLEARI** C,R
Clears all interrupts evoked by BASIC. Disables `ONTIME, ONEX1.`

**CLEARS** C,R
Resets BASIC-52's stacks. Sets control stack = 0FEh, argument stack = 1FEh, internal stack = value in 3Eh in internal RAM.

**CLOCK0** C,R
Disables the real-time clock.

**CLOCK1** C,R
Enables the real-time clock.

**CONT** C
Continues executing program after `STOP` or CONTROL+C.

**COS**(*expression*) C,R
Returns the cosine of *expression*

**CR**
`PRINT` option. Causes a carriage return, but no line feed, on the host display.

**DATA** *expression* [,...,*expression*] R
Specifies expressions to be retrieved by a `READ` statement.

**DBY**(*expression*) C,R
Retrieves or assigns a value at *expression* in internal data memory.

**DIM** *array name* [(*size*)] [,...*array name*(*size)*] C,R
Reserves storage for an array. Default size is 11 (0-10). Size limits are 0-254.
Example:

```
DIM B(100)
```

Reserves storage for 100-element array B

**DO:** [*program statements*]**: UNTIL** *relational expression* R
Executes all statements between DO and UNTIL until *relational expression* is true.

**DO:** [*program statements*]**: WHILE** *relational expression* R
Executes all statements between DO and WHILE until *relational expression* is false.

**END** R
Terminates program execution.

**EXP** (*expression*) C,R
Raises *e* (2.7182818) to the power of *expression*

**FOR** *counter variable* **=** *start-count expression* C,R
**TO** *end-count expression* [
**STEP** *count-increment expression*]**:** [*program statements*]**:**
**NEXT** [*counter variable*]
Executes all statements between FOR and NEXT the number of times specified by the counter and step expressions.

**FPROG**, **FPROG1**-**FPROG6** C
Like PROG, PROG1-PROG6, but using Intelligent programming algorithm.

**FREE** C,R
Returns the number of bytes of unused external data RAM.

**GET** R
Contains the ASCII code of a character received from the host computer's keyboard. After a program reads the value of GET (For example, G=GET), GET returns to 0 until a new character arrives.

**GOSUB** *line number* R
Causes BASIC-52 to transfer program control to a subroutine beginning at *line number.* A RETURN statement returns control to the line number following the GOSUB statement.

**GOTO** *line number* C,R
Causes BASIC-52 to jump to *line number* in the current program.

**IDLE** R
Forces BASIC-52 to wait for ONTIME or ONEX1 interrupt.

**IE** C,R
Retrieves or assigns a value to the 8052's special function register IE.

**IF** *relational expression* R
**THEN** *program statements*
[**ELSE**] [*program statements*]
If *relational expression* is true, executes program statements following THEN. If *relational expression* is false, executes program statements following ELSE, if used.

**INPUT** [**"***Prompt message***"**][,] *variable* [,*variable*] [,...*variable*] R
Displays a question mark and optional prompt message on the host computer and waits for keyboard input. Stores input in *variable*(s). A comma before the first variable suppresses the question mark.

**INT**(*expression*) C,R
Returns integer portion of *expression*.

**IP** C,R
Retrieves or assigns a value to the 8052's special function register IP.

**LD@** *expression* C,R
Retrieves a 6-byte floating-point number and places it on the argument stack. *Expression* points to the most significant byte of the number.

**LEN** C,R
Returns the number of bytes in the current program

[**LET**] *variable* **=** *expression* C,R
Assigns a variable to the value of *expression*. Use of LET is optional.

**LIST**[*line number*][-*line number*] C,R
Displays the current program on the host computer.

**LIST#** [*line number*][-*line number*] C,R
Writes the current program to LPT (pin 8).

**LIST@** [*line number*][-*line number*] C,R
Writes the current program to a user-written assembly-language output driver at 40C3h. Setting bit 7 of internal data memory location 27H enables the driver.

**LOG**(*expression*) C,R
Returns natural logarithm of *expression*.

**MTOP** [**=***highest address in RAM program space*] C,R
Assigns or reads the highest address BASIC-52 will use to store variables, strings, and RAM programs. Usually 7FFFh or lower, since EPROM space begins at 8000h.

**NEW** C
Erases current program in RAM; clears all variables.

**NOT** (*expression*) C,R
Returns 1's complement (inverse) of *expression*.

**NULL** [*integer*] C
Sets the number (0-255) of NULL characters (ASCII 00) that BASIC-52 sends automatically after a carriage return. Only very slow printers or terminals need these extra nulls.

**ON** *expression* **GOSUB** *line number* [,*line number*] [,...,*line number*] R
Transfers program control to a subroutine beginning at one of the line numbers in the list. The value of *expression* matches the position of the line number selected, with the first line number at position 0.

Examples:

```
X=1
ON X GOTO 100,200,400
```
Transfers program control to a subroutine at line 200 (position 1 in the list)

```
X=0
ON X GOTO 800,300
```
Transfers program control to a subroutine at line 800 (position 0 in the list)

**ON** *expression* **GOTO** *line number* [,*line number*] [,...,*line number*] R
Transfers program control to one of the line numbers in a list of numbers. The value of *expression* matches the position of the line number selected, with the first line number at position 0.
Example:

```
X=0
ON X GOTO 800,300
```
Transfers program control to line 800 (position 0 in the list)

**ONERR** *line number* R
Passes control to *line number* following an arithmetic error. Arithmetic errors include ARITH. OVERFLOW, ARITH. UNDERFLOW, DIVIDE BY ZERO, and BAD ARGUMENT.

**ONEX1** *line number* R
On interrupt 1 (pin 13), BASIC-52 finishes executing the current statement, and then passes control to an interrupt routine beginning at *line number*. The interrupt routine must end with `RETI`.

**ONTIME** *number of seconds, line number* R
When `TIME =` *number of seconds*, BASIC-52 passes control to an interrupt routine beginning at *line number.* The interrupt routine must end with `RETI.` `CLOCK1` starts the timer.

*expression* **.OR.** *expression* C,R
Logical OR

**P.**
same as `PRINT`

**PCON** C,R
Retrieves or assigns a value to the 8052's special function register PCON.

**PGM** C,R
Programs an EPROM, EEPROM, or NV RAM with data from memory. The following data must be stored in internal data memory in the locations listed:

| | |
|---|---|
| 1Bh,19h | High byte, low byte of first address of data to program |
| 1Ah,18h | High byte, low byte of first address to be programmed - 1 |
| 1Fh,1Eh | High byte, low byte indicating number of bytes to program |
| 40h,41h | High byte, low byte indicating width of programming pulse.<br>High byte = ((65536 - pulse width in seconds * XTAL/12) / 256.<br>Low byte = ((65536 - pulse width in seconds * XTAL/12) .AND. 0FFh. |
| 26h | For Intelligent programming, set bit 3.<br>For 50-millisecond programming, clear bit 3. |

**PH0.** C,R
Same as `PRINT,` but displays values in hexadecimal format. Uses two digits to display values less than 0FFh.

**PH0.#** C,R
Same as `PRINT#,` but displays values in `PH0.` hexadecimal format

**PH0.@** C,R
Same as PRINT@, but outputs values in PH0. hexadecimal format.

**PH1.** C,R
Same as PRINT, but displays values in hexadecimal format. Always displays four digits.

**PH1.#** C,R
Same as PRINT#, but displays values in PH1. hexadecimal format.

**PH1.@** C,R
Same as PRINT@, but outputs values in PH1. hexadecimal format.

**PI** C,R
Constant equal to 3.1415926.

**POP** *variable* [,...*variable*] C,R
Assigns the value of the top of the argument stack to *variable.*

**PORT1** C,R
Retrieves or assigns a value to PORT1 (pins 1-8).

**PRINT** [*expression*] [,...*expression*] [,] C,R
Displays the value of *expression*(s) on the host computer. A comma at the end of the statement suppresses the CARRIAGE RETURN/LINEFEED. Values are separated by two spaces. Additional PRINT options are CR, SPC, TAB, USING.

**PRINT#** C,R
Same as PRINT, but outputs to LPT (pin 8). BAUD and XTAL values affect the PRINT# rate.

**PRINT@** C,R
Same as PRINT, but outputs to a user-defined output driver. Requires an assembly-language output routine at 403Ch in external program memory. Setting bit 7 of internal data memory location 24h enables the output routine.

**PROG** C
Stores the current RAM program in the EPROM space.

**PROG1** C
Saves the serial-port baud rate. On power-up or reset, BASIC-52 boots without having to receive a space character. The terminal's baud rate must match the stored value.

**PROG2** C
Like PROG1, but on power-up or reset, BASIC-52 also begins executing the first program in the EPROM space.

**PROG3** C
Like PROG1, but also saves MTOP. On power-up or reset, BASIC-52 clears memory only to MTOP.

**PROG4** C
Like PROG2, but also saves MTOP. On power-up or reset, BASIC-52 clears memory only to MTOP.

**PROG5** C
Like PROG3, but also reads 5Fh in external data memory on power-up or reset. If 5Fh contains 0A5h, BASIC-52 doesn't clear external data memory. If data memory location 5Eh contains 34h, BASIC-52 will automatically begin executing a program in external data memory.

**PROG6** C
Like PROG5, but if external data memory location contains 5Fh, BASIC-52 calls a user-written assembly-language reset routine beginning at program memory 4039h.

**PUSH** *expression* [,...*expression*] C,R
Places the values of *expression*(s) sequentially on BASIC-52's argument stack.

**PWM** *expression1, expression2, expression3* C,R
Outputs a pulse-width modulated (PWM) sequence of pulses on pin 3. *Expression1* is the width of each high pulse, expressed in clock cycles. *Expression2* is the width of each low pulse, expressed in clock cycles. *Expression3* is the number of PWM cycles output. One clock cycle = 12/XTAL. One PWM cycle = one high pulse plus one low pulse. *Expression1* and *Expression2* must each be at least 25. Maximum for each *Expression* is 65535.

**RAM** C
Selects the current program in the RAM space.

**RCAP2** C,R
Retrieves or assigns a value to the 8052's special function registers RCAP2H and RCAP2L.

**READ** *variable* [*,...,variable*] R
Retrieves the expressions in a DATA statement and assigns each expression to a variable.

**REM** C,R
Introduces a comment, or remark. BASIC-52 ignores all text after REM in a program line.

**RESTORE** R
Resets READ pointer to the first expression in the DATA statement.

**RETI** R
Returns program control to the line number following the most recently executed ONEX1 or ONTIME statement.

**RETURN** R
Returns program control to the line number following the most recently executed GOSUB statement.

**RND** C,R
Returns a pseudo-random number between 0 and 1 inclusive.

**ROM** [*program number*] C
Selects a program in the EPROM space (beginning at 8000h). Default program number is 1.

**RROM** [*program number*] C,R
Changes to ROM mode and runs the specified program. Default program number is 1.

**RUN** R
Executes the current program. Clears all variables.

**SGN** (*expression*) C,R
Returns +1 if *expression* >=0, zero if *expression* = 0, and -1 if *expression* <0.

**SIN**(*expression*) C,R
Returns the sine of *expression*

**SPC**(*expression*)
PRINT option. Causes the display to place *expression* additional spaces (besides the minimum two) between values in a PRINT statement.

Example:

```
PRINT “hello”,SPC(3),"good-by"
hello     good-by
```

**SQR**(*expression*) C,R
Returns square root of expression.

**ST@** *expression* C,R
Copies a 6-byte floating-point number from the argument stack to external data memory. *Expression* points to the most significant byte of the number.

**STOP**
Halts program execution.

**STRING** *expressions, expression2* C,R
Allocates memory for strings (variables each consisting of a series of text characters).
*Expression1* = (*Expression2* * number of strings) + 1.
*Expression2* = maximum number of bytes (characters) per string + 1. Executing STRING clears all variables. Maximum number of strings is 255.
Examples:

```
STRING 91,9
```
reserves space for ten 8-character strings

```
STRING 9,4
```
reserves space for two 3-character strings

**T2CON** C,R
Retrieves or assigns a value to the 8052’s special function register T2CON.

**TAB**(*expression*),
PRINT option. Specifies the position (number of spaces) to begin displaying the next value in the PRINT statement.
Example:

```
PRINT TAB(5) “hello”
     hello
```

```
PRINT TAB(2) “hello”
  hello
```

**TAN**(*expression*) C,R
Returns the tangent of *expression*.

**TCON** C,R
Retrieves or assigns a value to the 8052’s special function register TCON.

**TIME** C,R
Retrieves or assigns a value, in seconds, to BASIC-52’s real-time clock.

**TIMER0** C,R
Retrieves or assigns a value to the 8052’S special function registers TH0 and TL0.

**TIMER1** C,R
Retrieves or assigns a value to the 8052’s special function registers TH1 and TL1.

**TIMER2** C,R
Retrieves or assigns a value to the 8052’s special function registers TH2 and TL2.

**TMOD** C,R
Retrieves or assigns a value to the 8052’s special function register TMOD.

**U.**
PRINT option. Same as USING.

**UI0** C,R
Restores BASIC-52’s console input driver after using UI1.

**UI1** C,R
Allows a user-provided assembly-language console (host computer) input routine to replace BASIC-52’s console input driver. External program memory location 4033h must contain a jump to the user’s routine.

**UO0** C,R
Restores BASIC-52’s console output driver after using UI1.

**UO1** C,R
Allows a user-provided assembly-language console (host computer) output routine to replace BASIC-52’s console output driver. External program memory location 4030h must contain a jump to the user’s routine.

**USING** (**F***N*)
PRINT option. Causes BASIC-52 to output numbers in exponential format with *N* significant digits. BASIC-52 always outputs at least 3 significant digits. Maximum *expression* is 8.

Example:

```
PRINT USING(F3),3,4.1,100
3.00 E 0
4.10 E 0
1.00 E 2
```

**USING(0)**
PRINT option. Causes BASIC-52 to output numbers from ±.99999999 to ±0.1 as decimal fractions. Numbers outside this range display in USING(FN) format. USING(0) is the default format.

**USING** (*#*[...*#*][.]*#*[...*#]*)
PRINT option. Causes BASIC-52 to output numbers using decimal fractions, with # representing the number of significant digits before and after the decimal point. Up to eight # characters are allowed.

Example:

```
PRINT USING(###.##),3,4.1,100
  3.00
  4.10
00.00
```

**XBY**(*expression*) C,R
Retrieves or assigns a value in external data memory.

**XFER** C
Copies the current program from the EPROM space (beginning at 8010h for program 1) to RAM (beginning at 200h), and selects RAM mode.

*expression* **.XOR.** *expression* C,R
Logical exclusive OR

**XTAL** C,R
Assigns a value equal to the system's crystal frequency, for use by BASIC-52 in timing calculations.

1 The Microcontroller Idea Book Circuits, Programs & Applications featuring the 8052-BASIC Single-chip Computer Jan Axelson. 2 Table of Contents Chapter 1 Microcontroller Basics 1 What s a Microcontroller? 1 A Little History 2 New Tools 3 Project Steps 4Â  and Measure 153 Sensor Basics 153 Choosing Sensors 154 On/off Sensors 155 Analog Sensors 156 Sensor Examples 163 Level Translating 167 Choosing a Converter 169 Chapter 10 Clocks and Calendars 171 BASIC-52 s Real-time Clock 171 A Watchdog Timekeeper 174 iv. 4 Introduction Introduction This book is a hands-on guide to designing, building, and testing microcontroller-based devices. The 100 best microcontrollers books, such as TinyML, Arduino Workshop, Coding the Arduino and The Z80 Microprocessor.Â As featured on CNN, Forbes and Inc â€“ BookAuthority identifies and rates the best books in the world, based on recommendations by thought leaders and experts. We may earn a commission for purchases made through links in this page. Learn more. Book: PIC Microcontrollers -Programming in C. Table of Contents. Â§ Â§ Chapter 1: World of Microcontrollers.Â  The situation we find ourselves today in the field of microcontrollers has its beginnings in the development of technology of integrated circuits. It enabled us to store hundreds of thousands of transistors into one chip, which was a precondition for the manufacture of microprocessors.Â  Originally, the main idea was to express logical forms through algebraic functions. Such thinking was soon transformed into a practical product which far later evaluated in what today is known as AND, OR and NOT logic circuits. The principle of their operation is known as Boolean algebra.