

The following material is excerpted from:

## ***The Microcontroller Idea Book***

***Circuits, Programs, & Applications***

***featuring the 8052-BASIC Microcontroller***

by Jan Axelson

copyright 1994, 1997 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](mailto:jaxelson@lvr.com)

World Wide Web: <http://www.lvr.com>

You may distribute this material if you agree to distribute it in full and unchanged and agree to charge no fee for such distribution with the exception of reasonable media charges.

*The author and publisher have used their best efforts in preparing this work and the materials in it. The author built and tested the electronic circuits described, ran and tested the computer programs presented, and reviewed all materials for completeness and accuracy. The author and publisher make no warranty with regard to the circuit schematics, program listings, and other materials in this work. The author and publisher take no responsibility for any damages resulting from any use of the material in this work.*

## 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

<i>expression / expression</i> Divide	C,R
<i>expression ** expression</i> Raises first expression to value of second expression (exponent)	C,R
<i>expression &lt;&gt; expression</i> Inequality test (relational operator)	C,R
<i>expression &lt; expression</i> Less than test (relational operator)	C,R
<i>expression &gt; expression</i> Greater than test (relational operator)	C,R
<i>expression &lt;= expression</i> Less than or equal test (relational operator)	C,R
<i>expression &gt;= expression</i> Greater than or equal test (relational operator)	C,R
<b>?</b> Same as PRINT	
<b>ABS</b> ( <i>expression</i> ) Returns the absolute value of <i>expression</i>	C,R
<i>expression .AND. expression</i> Logical AND	C,R
<b>ASC</b> ( <i>character</i> ) Returns the value of ASCII character	C,R
<b>ATN</b> ( <i>expression</i> ) Returns the arctangent of <i>expression</i>	C,R
<b>BAUD</b> <i>expression</i> Sets the baud rate for LPT (pin 8). For proper operation, XTAL must match the system's crystal frequency.	C,R
<b>CALL</b> <i>integer</i> Calls an assembly-language routine at the specified address in program memory.	C,R

<b>CBY</b> ( <i>expression</i> )	C,R
Retrieves the value at <i>expression</i> in program, or code, memory.	
<b>CHR</b> ( <i>expression</i> )	C,R
Converts <i>expression</i> to its ASCII character.	
<b>CLEAR</b>	C,R
Sets all variables to 0, resets all stacks and interrupts evoked by BASIC.	
<b>CLEAR1</b>	C,R
Clears all interrupts evoked by BASIC. Disables <code>ONTIME</code> , <code>ONEX1</code> .	
<b>CLEAR5</b>	C,R
Resets BASIC-52's stacks. Sets control stack = 0FEh, argument stack = 1FEh, internal stack = value in 3Eh in internal RAM.	
<b>CLOCK0</b>	C,R
Disables the real-time clock.	
<b>CLOCK1</b>	C,R
Enables the real-time clock.	
<b>CONT</b>	C
Continues executing program after <code>STOP</code> or <code>CONTROL+C</code> .	
<b>COS</b> ( <i>expression</i> )	C,R
Returns the cosine of <i>expression</i>	
<b>CR</b>	
PRINT option. Causes a carriage return, but no line feed, on the host display.	
<b>DATA</b> <i>expression</i> [..., <i>expression</i> ]	R
Specifies expressions to be retrieved by a READ statement.	
<b>DBY</b> ( <i>expression</i> )	C,R
Retrieves or assigns a value at <i>expression</i> in internal data memory.	
<b>DIM</b> <i>array name</i> [( <i>size</i> )] [... <i>array name</i> ( <i>size</i> )]	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	

<b>DO:</b> [ <i>program statements</i> ]: <b>UNTIL</b> <i>relational expression</i>	R
Executes all statements between DO and UNTIL until <i>relational expression</i> is true.	
<b>DO:</b> [ <i>program statements</i> ]: <b>WHILE</b> <i>relational expression</i>	R
Executes all statements between DO and WHILE until <i>relational expression</i> is false.	
<b>END</b>	R
Terminates program execution.	
<b>EXP</b> ( <i>expression</i> )	C,R
Raises <i>e</i> (2.7182818) to the power of <i>expression</i>	
<b>FOR</b> <i>counter variable</i> = <i>start-count expression</i>	C,R
<b>TO</b> <i>end-count expression</i> [ <b>STEP</b> <i>count-increment expression</i> ]: [ <i>program statements</i> ]: <b>NEXT</b> [ <i>counter variable</i> ]	
Executes all statements between FOR and NEXT the number of times specified by the counter and step expressions.	
<b>FPROG, FPROG1-FPROG6</b>	C
Like PROG, PROG1-PROG6, but using Intelligent programming algorithm.	
<b>FREE</b>	C,R
Returns the number of bytes of unused external data RAM.	
<b>GET</b>	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.	
<b>GOSUB</b> <i>line number</i>	R
Causes BASIC-52 to transfer program control to a subroutine beginning at <i>line number</i> . A RETURN statement returns control to the line number following the GOSUB statement.	
<b>GOTO</b> <i>line number</i>	C,R
Causes BASIC-52 to jump to <i>line number</i> in the current program.	
<b>IDLE</b>	R
Forces BASIC-52 to wait for ONTIME or ONEX1 interrupt.	

<b>IE</b>	C,R
Retrieves or assigns a value to the 8052's special function register IE.	
<b>IF</b> <i>relational expression</i>	R
<b>THEN</b> <i>program statements</i>	
<b>[ELSE]</b> [ <i>program statements</i> ]	
If <i>relational expression</i> is true, executes program statements following THEN. If <i>relational expression</i> is false, executes program statements following ELSE, if used.	
<b>INPUT</b> [" <i>Prompt message</i> "][,] <i>variable</i> [, <i>variable</i> ] [... <i>variable</i> ]	R
Displays a question mark and optional prompt message on the host computer and waits for keyboard input. Stores input in <i>variable</i> (s). A comma before the first variable suppresses the question mark.	
<b>INT</b> ( <i>expression</i> )	C,R
Returns integer portion of <i>expression</i> .	
<b>IP</b>	C,R
Retrieves or assigns a value to the 8052's special function register IP.	
<b>LD@</b> <i>expression</i>	C,R
Retrieves a 6-byte floating-point number and places it on the argument stack. <i>Expression</i> points to the most significant byte of the number.	
<b>LEN</b>	C,R
Returns the number of bytes in the current program	
<b>[LET]</b> <i>variable</i> = <i>expression</i>	C,R
Assigns a variable to the value of <i>expression</i> . Use of LET is optional.	
<b>LIST</b> [ <i>line number</i> ][- <i>line number</i> ]	C,R
Displays the current program on the host computer.	
<b>LIST#</b> [ <i>line number</i> ][- <i>line number</i> ]	C,R
Writes the current program to LPT (pin 8).	
<b>LIST@</b> [ <i>line number</i> ][- <i>line number</i> ]	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)
```

<b>ONERR</b> <i>line number</i>	R
Passes control to <i>line number</i> following an arithmetic error. Arithmetic errors include ARITH. OVERFLOW, ARITH. UNDERFLOW, DIVIDE BY ZERO, and BAD ARGUMENT.	
<b>ONEX1</b> <i>line number</i>	R
On interrupt 1 (pin 13), BASIC-52 finishes executing the current statement, and then passes control to an interrupt routine beginning at <i>line number</i> . The interrupt routine must end with RETI.	
<b>ONTIME</b> <i>number of seconds, line number</i>	R
When TIME = <i>number of seconds</i> , BASIC-52 passes control to an interrupt routine beginning at <i>line number</i> . The interrupt routine must end with RETI . CLOCK1 starts the timer.	
<i>expression .OR. expression</i>	C,R
Logical OR	
<b>P.</b>	
same as PRINT	
<b>PCON</b>	C,R
Retrieves or assigns a value to the 8052's special function register PCON.	
<b>PGM</b>	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. High byte = ((65536 - pulse width in seconds * XTAL/12) / 256. Low byte = ((65536 - pulse width in seconds * XTAL/12) .AND. 0FFh.
26h	For Intelligent programming, set bit 3. For 50-millisecond programming, clear bit 3.
<b>PH0.</b>	C,R
Same as PRINT , but displays values in hexadecimal format. Uses two digits to display values less than 0FFh.	
<b>PH0.#</b>	C,R
Same as PRINT# , but displays values in PH0 . hexadecimal format	

---

<b>PH0.@</b>	C,R
Same as PRINT@ , but outputs values in PH0 . hexadecimal format.	
<b>PH1.</b>	C,R
Same as PRINT , but displays values in hexadecimal format. Always displays four digits.	
<b>PH1.#</b>	C,R
Same as PRINT# , but displays values in PH1 . hexadecimal format.	
<b>PH1.@</b>	C,R
Same as PRINT@ , but outputs values in PH1 . hexadecimal format.	
<b>PI</b>	C,R
Constant equal to 3.1415926.	
<b>POP</b> <i>variable</i> [... <i>variable</i> ]	C,R
Assigns the value of the top of the argument stack to <i>variable</i> .	
<b>PORT1</b>	C,R
Retrieves or assigns a value to PORT1 (pins 1-8).	
<b>PRINT</b> [ <i>expression</i> ] [... <i>expression</i> ] [,]	C,R
Displays the value of <i>expression</i> (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.	
<b>PRINT#</b>	C,R
Same as PRINT , but outputs to LPT (pin 8). BAUD and XTAL values affect the PRINT# rate.	
<b>PRINT@</b>	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.	
<b>PROG</b>	C
Stores the current RAM program in the EPROM space.	

## Chapter 5

---

<b>PROG1</b>	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.	
<b>PROG2</b>	C
Like PROG1 , but on power-up or reset, BASIC-52 also begins executing the first program in the EPROM space.	
<b>PROG3</b>	C
Like PROG1 , but also saves MTOP. On power-up or reset, BASIC-52 clears memory only to MTOP.	
<b>PROG4</b>	C
Like PROG2 , but also saves MTOP. On power-up or reset, BASIC-52 clears memory only to MTOP.	
<b>PROG5</b>	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.	
<b>PROG6</b>	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.	
<b>PUSH</b> <i>expression</i> [... <i>expression</i> ]	C,R
Places the values of <i>expression</i> (s) sequentially on BASIC-52's argument stack.	
<b>PWM</b> <i>expression1</i> , <i>expression2</i> , <i>expression3</i>	C,R
Outputs a pulse-width modulated (PWM) sequence of pulses on pin 3. <i>Expression1</i> is the width of each high pulse, expressed in clock cycles. <i>Expression2</i> is the width of each low pulse, expressed in clock cycles. <i>Expression3</i> is the number of PWM cycles output. One clock cycle = 12/XTAL. One PWM cycle = one high pulse plus one low pulse. <i>Expression1</i> and <i>Expression2</i> must each be at least 25. Maximum for each <i>Expression</i> is 65535.	
<b>RAM</b>	C
Selects the current program in the RAM space.	

---

<b>RCAP2</b>	C,R
Retrieves or assigns a value to the 8052's special function registers RCAP2H and RCAP2L.	
<b>READ</b> <i>variable</i> [..., <i>variable</i> ]	R
Retrieves the expressions in a DATA statement and assigns each expression to a variable.	
<b>REM</b>	C,R
Introduces a comment, or remark. BASIC-52 ignores all text after REM in a program line.	
<b>RESTORE</b>	R
Resets READ pointer to the first expression in the DATA statement.	
<b>RETI</b>	R
Returns program control to the line number following the most recently executed ONEX1 or ONTIME statement.	
<b>RETURN</b>	R
Returns program control to the line number following the most recently executed GOSUB statement.	
<b>RND</b>	C,R
Returns a pseudo-random number between 0 and 1 inclusive.	
<b>ROM</b> [ <i>program number</i> ]	C
Selects a program in the EPROM space (beginning at 8000h). Default program number is 1.	
<b>RROM</b> [ <i>program number</i> ]	C,R
Changes to ROM mode and runs the specified program. Default program number is 1.	
<b>RUN</b>	R
Executes the current program. Clears all variables.	
<b>SGN</b> ( <i>expression</i> )	C,R
Returns +1 if <i>expression</i> >=0, zero if <i>expression</i> = 0, and -1 if <i>expression</i> <0.	
<b>SIN</b> ( <i>expression</i> )	C,R
Returns the sine of <i>expression</i>	

## Chapter 5

---

### **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*)

Returns square root of expression.

C,R

### **ST@** *expression*

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.

C,R

### **STOP**

Halts program execution.

### **STRING** *expressions, expression2*

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
```

C,R

### **T2CON**

Retrieves or assigns a value to the 8052's special function register T2CON.

C,R

### **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
```

<b>TAN</b> ( <i>expression</i> )	C,R
Returns the tangent of <i>expression</i> .	
<b>TCON</b>	C,R
Retrieves or assigns a value to the 8052's special function register TCON .	
<b>TIME</b>	C,R
Retrieves or assigns a value, in seconds, to BASIC-52's real-time clock.	
<b>TIMER0</b>	C,R
Retrieves or assigns a value to the 8052'S special function registers TH0 and TL0.	
<b>TIMER1</b>	C,R
Retrieves or assigns a value to the 8052's special function registers TH1 and TL1.	
<b>TIMER2</b>	C,R
Retrieves or assigns a value to the 8052's special function registers TH2 and TL2.	
<b>TMOD</b>	C,R
Retrieves or assigns a value to the 8052's special function register TMOD.	
<b>U.</b>	
PRINT option. Same as USING.	
<b>UI0</b>	C,R
Restores BASIC-52's console input driver after using UI1.	
<b>UI1</b>	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.	
<b>UO0</b>	C,R
Restores BASIC-52's console output driver after using UI1 .	
<b>UO1</b>	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.	

## Chapter 5

---

### **USING (FN)**

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  $\pm.99999999$  to  $\pm 0.1$  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)**

Retrieves or assigns a value in external data memory.

C,R

### **XFER**

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

C

### *expression* **.XOR.** *expression*

Logical exclusive OR

C,R

### **XTAL**

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

C,R