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
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  
Assigns a value to a variable

equation = expression  
Equivalence test (relational operator)

expression + expression  
Add

expression - expression  
Subtract

expression * expression  
Multiply
expression / expression
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.
Chapter 5

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

**CHR**(expression)
Converts *expression* to its ASCII character.

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

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

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

**CLOCK0**
Disables the real-time clock.

**CLOCK1**
Enables the real-time clock.

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

**COS**(expression)
Returns the cosine of *expression*

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

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

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

**DIM** array name [(size)] [,....array name(size)]
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
Executes all statements between DO and UNTIL until relational expression is true.

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

END
Terminates program execution.

EXP (expression)
Raises $e (2.7182818)$ to the power of expression

FOR counter variable = start-count expression
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
Like PROG, PROG1-PROG6, but using Intelligent programming algorithm.

FREE
Returns the number of bytes of unused external data RAM.

GET
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
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
Causes BASIC-52 to jump to line number in the current program.

IDLE
Forces BASIC-52 to wait for ONTIME or ONEX1 interrupt.
IE
Retrieves or assigns a value to the 8052’s special function register IE.

IF relational expression
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.

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 7FFH 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.
    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.

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

### PH1.
Same as PRINT, but displays values in hexadecimal format. Always displays four digits.

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

### PH1.@
Same as PRINT@, but outputs values in PH1. hexadecimal format.

### PI
Constant equal to 3.1415926.

### POP variable [...variable]
Assigns the value of the top of the argument stack to variable.

### PORT1
Retrieves or assigns a value to PORT1 (pins 1-8).

### PRINT [expression] [...expression] [,.]
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#
Same as PRINT, but outputs to LPT (pin 8). BAUD and XTAL values affect the PRINT# rate.

### PRINT@
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
Stores the current RAM program in the EPROM space.
Chapter 5

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

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

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

**PROG5**
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**
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]*
Places the values of *expression*(s) sequentially on BASIC-52’s argument stack.

**PWM** *expression1, expression2, expression3*
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**
Selects the current program in the RAM space.
RCAP2
Retrieves or assigns a value to the 8052’s special function registers RCAP2H and RCAP2L.

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

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

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

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

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

RND
Returns a pseudo-random number between 0 and 1 inclusive.

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

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

RUN
Executes the current program. Clears all variables.

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

SIN(expression)
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
```
PRINT TAB(2) "hello"
   hello

**TAN** (*expression*)

Returns the tangent of *expression*.

**TCON**

Retrieves or assigns a value to the 8052’s special function register TCON .

**TIME**

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

**TIMER0**

Retrieves or assigns a value to the 8052’S special function registers TH0 and TL0.

**TIMER1**

Retrieves or assigns a value to the 8052’s special function registers TH1 and TL1.

**TIMER2**

Retrieves or assigns a value to the 8052’s special function registers TH2 and TL2.

**TMOD**

Retrieves or assigns a value to the 8052’s special function register TMOD.

**U.**

PRINT option. Same as USING.

**UI0**

Restores BASIC-52’s console input driver after using UI1.

**UI1**

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**

Restores BASIC-52’s console output driver after using UI1.

**UO1**

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 \textit{expression} is 8.

Example:

```plaintext
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 ± 0.99999999 to ± 0.1 as decimal fractions. Numbers outside this range display in \textit{USING (FN)} format. \textit{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:

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

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

**XFER**
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**
Logical exclusive OR

**XTAL**
Assigns a value equal to the system’s crystal frequency, for use by BASIC-52 in timing calculations.
The microcontroller is the brain of the Arduino. In Practical AVR Microcontrollers, you’ll learn how to use Designing Embedded Systems with PIC Microcontrollers - Principles. 583 Pages·2007·12.16 MB·25,092 Downloads. 1.6 An introduction to PIC microcontrollers using the 12 Series. 17 65. 4.1 The main idea – what The 8051 Microcontroller and Embedded Useful Microcontroller Related Books. The links from this page take you to useful book suggestions - most of them are book suggestions for electronics or programming in general. These types of books are more useful in the long run as they give you a more solid grounding in fundamental techniques and these are books that I have found the most useful. In the C programming link page you’ll find the best books for C programming - strangely enough the most useful is actually the smallest! Books on C programming. Books on Electronics This is the best all round book on electronics covering a vast amount of ideas and information. Although this is University grade stuff the information is presented in an informal and maths free way (as much as possible).