

AttoBasic Byte-Wide Basic for ATtiny2313 / AT90S2313

A very small Basic interpreter for a very small chip, for limited debugging, monitor and control use.

This interpreter uses on-chip RAM only.

LIST OF COMMANDS, FUNCTIONS, AND OPERATORS

-commands and functions-

NEW	NEW PROGRAM	EX: NEW
LIST	LIST PROGRAM	EX: LIST
PRINT	PRINT VALUE TO SCREEN	EX: PRINT A
PRX	PRINT HEX	EX: PRX 100 results in the output: 64
PRB	PRINT BINARY	EX: PIB INB prints PINB in binary
\$	CONVERT TWO FOLLOWING CHARACTERS FROM ASCII	EX:
A:=\$31		
KEY	GET KEY FROM TERMINAL	EX: A := KEY ; or KEY (return) to pause.
EMIT	EMIT VALUE AS ASCII CHARACTER TO TERMINAL	EX: EMIT \$20
RUN	RUN PROGRAM	EX: RUN
IF-THEN	CONTROL STRUCTURE	EX: IF A=31 THEN GOTO 100
FOR-TO-NEXT	LOOPING STRUCTURE	EX: see below
GOSUB-RETURN	PROGRAM FLOW CONTROL	EX: see below
GOTO	PROGRAM FLOW	EX: GOTO 100
SIZE	PRINT REMAINING BYTES OF PROGRAM SPACE TO SCREEN	EX: SIZE
END	STOP EXECUTION OF PROGRAM	EX: END
PEEK	READ VALUE OF MEMORY	EX: PRX PEEK A,B
POKE	WRITE VALUE OF MEMORY	EX: POKE A,\$31; POKE VALUE,destination
<backspace>	DESTRUCTIVE BACKSPACE DURING LINE EDITING	
SAVE	SAVE PROGRAM AND VARIABLES TO EEPROM	EX: SAVE
LOAD	LOAD PROGRAM AND VARIABLES FROM EEPROM	EX: LOAD

-operator/relational-

:=	set equal to, LET instruction not needed)
=	used for evaluation as in IF a = b THEN...)
<>	not equal to
>	is greater than
<	is less than
-	subtraction, 8 bit unsigned
+	addition, 8 bit, unsigned
AND	logical AND between two 8 bit values
OR	logical OR between two 8 bit values
EXOR	logical Exclusive OR between two 8 bit values

-I/O-

OPB	OUTPUT PORT B	EX: OPB \$00
OPD	OUTPUT PORT D	EX: OPD \$00
ODB	OUTPUT DATA DIRECTION REGISTER B	EX: ODB \$FF

ODD	OUTPUT DATA DIRECTION REGISTER D	EX: ODD \$FF
INB	INPUT PIN B	EX: A:= INB
IND	INPUT PIN D	EX: B:= IND
SBB	SET BIT IN B	EX: SBB 2
CBB	CLEAR BIT IN B	EX: CBB 2
SBD	SET BIT IN D	EX: SBD 2
CBD	CLEAR BIT IN D	EX: CBD 2
ACO	ANALOG COMPARITOR OUTPUT EX: IF ACO THEN PRINT A.	
PWM8	PULSE WIDTH MODULATION 8 BIT EX: PWM 17	
PWE	PWM EXTENDED 10 BIT PWM ED: PWE 2,00 result in a	
50% cycle		
PWO	PWM OFF	

ABOUT COMMANDS

The If-Then structures may be set up with either of two alternate structures.

1. All on one line, such as "IF A = 2 THEN GOTO 10, or
2. if the "THEN" is omitted and the then is on a second line, as in:

```
" 10 IF A = B
20 goto 100
30 <next statement>"
```

where the line following the IF statement only executes if the statement is satisfied.

The relational operators, =, >, <, and <> return 0 if the test is true, and a nonzero value if the test is false. This derives from the equals evaluation being done by simple subtraction of the two values, thus when equal, the subtraction yields zero.

For example,

```
"print 7=2" results in the value 0 being displayed.
"print 7>2" results in the value 1 being displayed.
"print 2>7" results in the value 0 (a nonzero value) being
displayed.
```

The order of execution of line with more than one command or function on the line may at times differ from other implementations of basic. An argument will take the results or arguments immediately following the command, as the following example illustrates.

LIST

```
10 POKE 100,$60
20 POKE 200,$62
30 PRINT PEEK $60 + 2
40 PRINT 2 + PEEK $60
```

Free pages:1 chars:167

```
>RUN
200
102
-----
```

LOOPS AND GOSUBS

There is no stack for loops and gosubs -only one return address is stored for each, so while it is ok to put a gosub within a for-next loop, nesting loops or gosubs is not supported.

MEMORY LIMITATIONS

Four variables are allowed: A,B,C, and D. All values and arithmetic and logic operations are unsigned 8 bit. Program space is 71 characters. The line buffer is 19 characters long (20 counting the carriage return at the end). Because of limited RAM, only one simple instruction per line will work in most cases. More complex lines (such as: A:= \$16 AND B) will result in a stack error.

ENTERING PROGRAMS

Any line that has a numeral in the first column will be stored in the program memory if there is room for it when the return character is received.

Any line that does not have a numeral in the first column will be interpreted when the return character is received.

Lines may be edited before the return character is entered by using the backspace key. Once a line is entered, its part of the program until the NEW command is executed. There are no other provisions for editing.

Line numbers are merely labels for GOTO and GOSUB commands and need not be in any particular sequence or order. Line numbers above 255 may be entered but may not work properly with GOTO and GOSUB statements.

Commands may be entered in either upper-case or lower-case as they are all converted to upper case when read by the interpreter.

The first three letters of a command may be used in place of the entire command in the case of commands longer than three characters. Thus, "PRI" can substitute for "PRINT". Commands and numbers need to be delimited so the parser can tell them apart. Two formal delimiters supported are the space character and the comma. Numbers and letters are informally delimited and so do not need formal delimiters. Thus, the "\$" actually an operator and needs to be formally delimited from other operators such as +, -, =, etc.

The "TO", as in "FOR A = 1 to 9" may be replaced with one or more spaces, thus the line: "FOR A = 1 9" is equivalent. The "TO" command is included as a convenience.

A RETURN command without a corresponding GOSUB has the same effect as END in that execution will stop.

The following examples may clarify some of the preceding.

OK: A=255
 NOT OK: A=256

OK: PRINT \$0F
 NOT OK: PRINT \$F (Result is garbage because \$ expects two chars and does not check.)

OK: FOR A=
 NOT OK: FORA= (can't tell the difference between the "FOR" and the "A")

OK: A:= \$10
 A:= \$10+10
 A:= A+ \$10
 B:= \$31
 PRX A+B
 PRX \$2A
 PRX \$10 + 10
 NOT OK: PRX \$10 + \$10 because of limited stack depth.

OK: A= \$3A
 NOT OK: A=\$3A (need a space between "=" and "\$")

OK; PRINT A
 NOT OK: PRINT A,B (print will only print the last value on a line)

ERROR CODES

The interpreter detects and flags many kinds of errors with error codes. To conserve FLASH space, only error numbers are printed to the terminal.
 Here is the key:

- 0: Unknown command or operator.
- 1: Encountered a value that is not between 0 and 255.
- 2: Program memory is full.
- 3: Encountered a character that is not a recognized type.
- 4:(internal error -buffer limit exceeded)
- 5: Inappropriate variable name encountered.
- 6: Push data stack error -too many arguments on one line.
- 7: Pop data stack error. Function tried to use a value that was not available.
- 8: Machine stack error Functions and/or calls nested too deeply or too complex
- 9: GOTO or GOSUB destination line was not found.

SOME EXAMPLE PROGRAMS

Take note that in these examples, every possible trick to conserve program memory is used. Minimum use is made of delimiters and commands are represented by their three letter abbreviations, which are the first three letters of the command. Thus "nex" means "next" and "got" means "goto" in the first example.

The tradeoff is between readability and program memory.

Dump memory example: The program below dumps the first 8 bytes of memory, followed by the constant 255, then waits for a character from the terminal before doing it again.

```
-----  
      Atto Basic V0.2. Copyright 2002 Richard  
Cappels,projects@cappels.org
```

```
>load
```

```
>list
```

```
1for a=0 8  
2prx pee a  
3nex  
4pri255  
5key  
6got1
```

```
25  Free
```

```
>run  
$02  
$10  
$23  
$FF  
$FE  
$79  
$00  
$AB  
$00  
255
```

```
-----  
Strobe all bits on port b when carriage return is received,  
ignore all other keys
```

```
-----  
      Atto Basic V0.2. Copyright 2002 Richard  
Cappels,projects@cappels.org
```

```
>load
```

```
>list

1opb0
2a:=key
3if a=13
4gos8
5got1
8opb $FF
9ret
```

```
22 Free
```

```
>
```

Monitor bits on port b example.
Read port B and print in binary to terminal.

Atto Basic V0.2. Copyright 2002 Richard
Cappels,projects@cappels.org

```
>load
```

```
>list
```

```
1prb inb
2got1
```

```
56 Free
```

```
>run
11100000
11100000
11100000
11100000
```

Move lower 3 bits of port B to port D example

Atto Basic V0.2. Copyright 2002 Richard
Cappels,projects@cappels.org

```
>load
```

```
>list
```

```
1 a:= ind
2 b:= inb
```

```
3 c:=a and 7
4 opd c or b
```

25 Free

>

Sample PORTB, bit zero and increments counter each time bit
zero is
found to be high and sends current total to terminal

Atto Basic V0.2. Copyright 2002 Richard
Cappels,projects@cappels.org

>load

>list

```
1a:=1 and inb
2if a=1
3gos 8
4got1
8B:=B+1
9pri B
10 ret
```

14 Free

>B:=0

>run

```
1
2
3
4
5
6
7
8
9
10
11
12
```