Cobalt Basic v1

User Manual

by Leonardo Leoni

```
Why the Basic language?
-----------------------
The Basic language was the first language to be "user-friendly", easy to learn and
to understand. It uses an interpreter, so you can write a command after the prompt
and run it immediately. It was the first simple computer language and became very
common in the home computer sector during the 80's.


What is the Cobalt Basic language?
----------------------------------
Cobalt Basic is one version of the Tiny Basic language developed initially as Palo
Alto Tiny Basic by Li-Chen Wang (1976) and it ran on either the 8080 or Z80
processor. Nowadays there are some very similar versions that run on ATmega328,
some are the TOYOSHIKI Tiny BASIC for Arduino, the Tiny Basic 68k by Gordon
Brandly, the Arduino Basic by Michael Field and the Tiny Basic Plus by Scott
Lawrence. Many credits to those guys.
Cobalt Basic is nothing more that a simpler version, adapted to stay into a little
atmega328 together with the display and keyboard libraries, leaving also the space
for your program to be put in the memory.


What can Cobalt Basic do?
-------------------------
Ok, but again: why the Basic Langauge? Because it's the perfect language if you
don't know anything about programming, because it's simple but complete. Think
about that: with this computer language you can print words on the screen, give
value to variables, make simple mathematical calculation, use subroutines, repeat
commands also infinitely and do much more.


Does it include a standard keyboard?
------------------------------------
Only two words about the keyboard. It's important to understand the keyboard
layout, a QWERTY one but with several particularities, so take two minutes to give
a look to the Cobalt 3 keyboard guide before using it.

Characters typed with the keyboard are lowercase, but the Cobalt Basic commands are
stored in memory uppercase, so you can see the difference from what you type and
what is stored. It's useful when you subsstitute a row of the program with another
one. Type LIST command to see your program uppercase.
Example:

>10 prit "dog"
>run

Syntax Error
>10 print "dog"
>list
10 PRINT "dog"
>run
dog
```

```
OK
>_
```

"Syntax Error" is the error message when there's a mistake in your program.

So, what else, let's start with Cobalt Basic! Let's look together at the commands available!

```
Summary
-------
Relational operators
Arithmetic operations
NEW - clears the last program
RUN - executes the current program
LIST - prints the current program
LET - sets the value of a variable
IF - if expression is true, elaborates an istruction
GOTO - continues execution at a given line number
GOSUB - goes to a subroutine at a given line number
RETURN - returns from a subroutine
REM - comments
FOR - loop block, uses TO and STEP
TO - the end value that stops the for loop
STEP - adds a different value in a for loop
NEXT - end of for block
INPUT - gives to the user the chance of insert a value for a variable
PRINT - displays something on the screen, also "?"
STOP - ends the execution of the current program
BYE - exits, a soft reboot of the Cobalt device
```

```
Relational operators
--------------------
```
The Cobalt Basic v1 has six relational operators used to compare two operands, obtaining a true/false result.

```
= equality
< less than
> greater than
<= less or equal
>= greater or equal
<> not equal
```

```
Arithmetic operations
---------------------
```
The Cobalt Basic v1 can do four arithmetic operations. It can add, substruct, multiply and divide.

```
+ add
- substruct
* multiply
/ divide
```

```
NEW
---
```
Use this command at the Cobalt prompt when you want to clear the memory and start a new program. It cancels an old program, lines, or commands, clearing all.

```
>new
>list
OK


RUN
---
This is the command you need to launch the program you wrote.
A program is made of several rows, each one has an instruction to execute.
At the beginning of each row you can write the number of the row. If you do it, the
row will be part of the current program.
The numbers are for ordering the statements of the program.
Without the number at the beginning of the row, the instruction will be executed
immediately from the Cobalt Basic interpreter.

>run

See the example in the LET statement.


LIST
----
The statement is used to view the program you are typing or you finished.

>list

See the example in the STEP statement.


LET
---
This instruction assigns a value to a variable.
The value could be a number or another variable.

>10 let p=1+7
>20 print p
>run
8
OK


IF
--
It checks if an expression is true, and if so, an inline instructions will be
executed.

The relational operators are:
= equality
< less than
> greater than
<= less or equal
>= greater or equal
<> not equal

Examples:

>10 a=3
>20 if a > 2 print "hi"
>run
hi
OK
```

```
GOTO
----
The GOTO statement has the line number where the program execution will continue,
jumping to that line.
Example:

>10 a=2
>20 a=a+1
>30 print a
>40 if a < 10 print a
>50 if a < 5 goto 20
run
3
3
4
4
5
5
OK


GOSUB
-----
The GOSUB statement has the line number where the program execution will continue,
similar to GOTO, with a difference: the block that will be executed (subroutine)
ends with the keyword RETURN.
So in this case, the excution will jump back to the line after the GOSUB statement.
Consider also the use of STOP to exit the program execution.

>10 z = 2
>20 z = z + 1
>30 gosub 300
>40 if z < 5 goto 20
>50 stop
>300 rem subroutine for printing
>310 print z
>320 return
>run
3
4
5
OK


RETURN
------
The RETURN keyword indicates the end of a subroutine called by a GOSUB.


REM
---
After a line number, the word REM is used to write a comment, in this way:

>20 rem Program about financial calculation


FOR
---
The FOR keyword is used for repeating a block of istructions. How does it work?
```

Let's look at the example below.

```
>10 for t=2 to 5
>20 print t
>30 next t
>run
2
3
4
5
OK
```

After the FOR keyword, "t=2" is the initial value of "t", while "to 5" means the
FOR block will be repeated until the value of "t" will be "5".
In the FOR block there's the command "print t", that means the value of "t" will be
displayed.
The row with "next t" indicates the end of the loop.

TO
--
In a FOR loop, after the starting value, the block will be repeated until the end
value, showed after the "to" keyword.
See the example in the FOR statement.

STEP

Substituting the row number 10, this is what will happen:

```
>10 for t=2 to 10 step 2
>list
10 FOR T=2 TO 5
20 PRINT T
30 NEXT T
>run
2
4
6
8
10
OK
```

That's because adding "step 2" means add "2" to the value of "t";

NEXT

The statement that marks the end of a FOR loop. It means "next round";
Example:

```
>30 next t
```

INPUT

With this command a usser can assign a number to a variable.
Below, when I run the program, I write "2" after the "?", and the program assigns
it to the variable a. Then it prints its value.
See also the use of the command LIST.

```
>10 input a
>20 print a
>list
10 INPUT A
20 PRINT A
OK
>run
?2
2
OK
>_
```

PRINT

Used to make the program write something on the screen. After the execution,the system shows the prompt and the cursor.

```
>print "hello"
hello
>_
```

If you add a semicolon at the end, the Cobalt Basic interpreter writes the commands inline, so you have:

```
>print "hello";
hello>_
```

STOP

This command ends the execution of the program.
See the example in the GOSUB statement.

BYE

This is a system command, you can use it if you need a soft reboot of the system with the reset of the memory.

```
>bye
1226 bytes free.
OK
>_
```

Thanks for reading.

Cheers,

Leonardo