CHIPSTAR BASIC - Quick Reference

Paul Kocyla

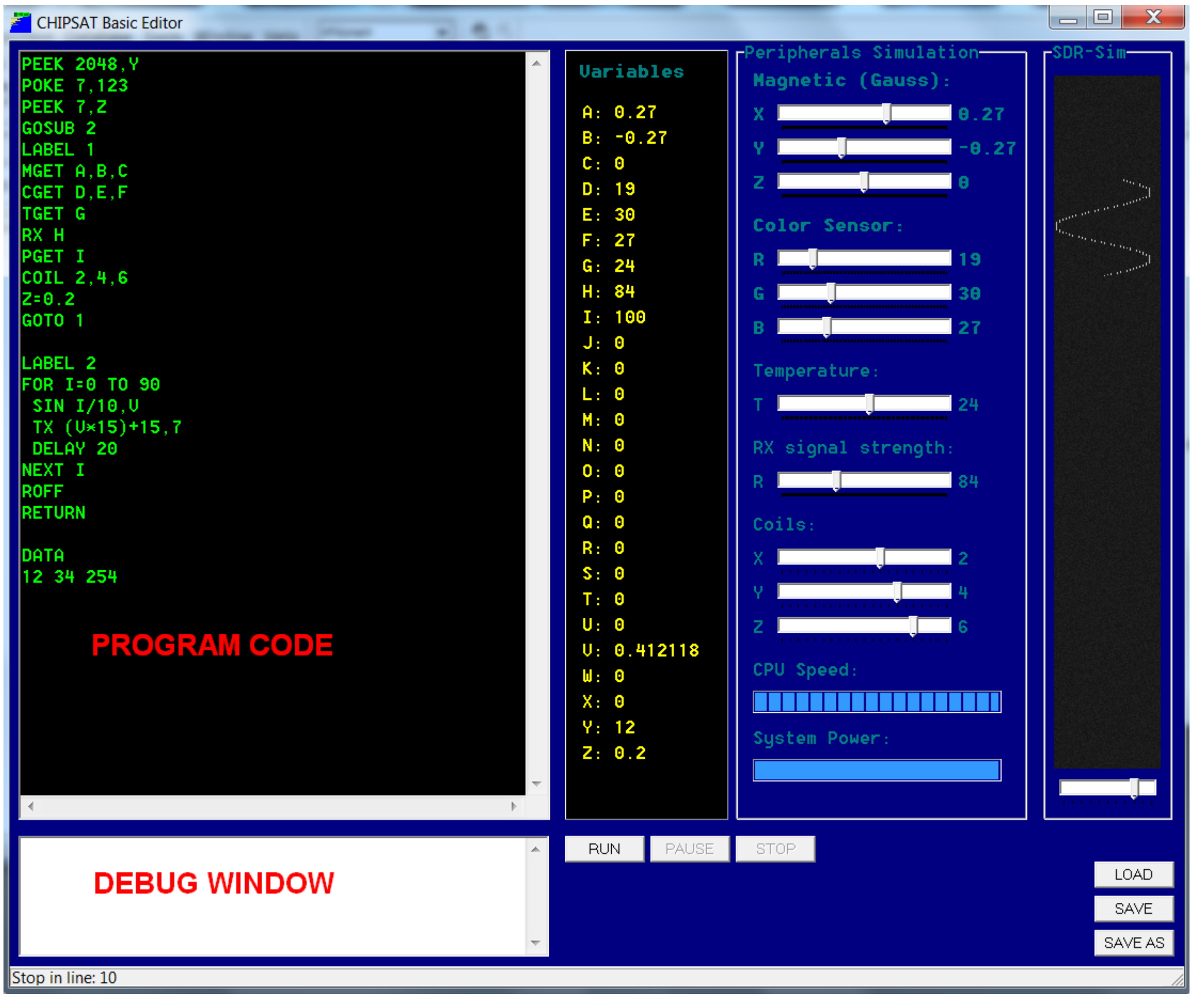
21.01.2015

Contents

1 Environment

BASIC (an acronym for Beginner's All-purpose Symbolic Instruction Code) is a family of general-purpose, high-level programming languages whose design philosophy emphasizes ease of use. The CHIPSTAR BASIC environment offers a graphical user interface to create program code for the Chipstar femto-satellite system. An integrated simulator allows to test the program directly on the PC, without the need to upload it into the Chipstar or the Chipstar Devkit. The environment can export a binary file which can be uploaded into the Chipstar Devkit and tested under real hardware conditions.

Graphical User Interface (GUI)



```
PEEK 2048,Y
POKE 7,123
PEEK 7,Z
GOSUB 2
LABEL 1
MGET A,B,C
CGET D,E,F
TGET G
RX H
PGET I
COIL 2,4,6
Z=0.2
GOTO 1

LABEL 2
FOR I=0 TO 90
 SIN I/10,V
 TX (V*15)+15,7
 DELAY 20
NEXT I
ROFF
RETURN

DATA
12 34 254
```

```
Variables

A: 0.27
B: -0.27
C: 0
D: 19
E: 30
F: 27
G: 24
H: 84
I: 100
J: 0
K: 0
L: 0
M: 0
N: 0
O: 0
P: 0
Q: 0
R: 0
S: 0
T: 0
U: 0
V: 0.412118
W: 0
X: 0
Y: 12
Z: 0.2
```

```
Magnetic (Gauss):
X 0.27
Y -0.27
Z 0

Color Sensor:
R 19
G 30
B 27

Temperature:
T 24

RX signal strength:
R 84

Coils:
X 2
Y 4
Z 6
```

Stop in line: 10

- The environment is a one-screener. Everything you need to write code for your satellite is available on this screen.
- You write your program in the editor window named “PROGRAM CODE”.
- All program variables (A-Z) are shown in the window “Variables”.
- The button “RUN” starts the simulation of your program code.
- The satellite´s peripherals deliver data which can be used in your program code. The real satellite will provide you the sensor data in form of numbers. In the simulator, you can provide your code a fake data by dragging the sliders in the “Peripherals SImulation” window.
- Errors will be shown in the box named “DEBUG WINDOW”
- The “SDR-Sim” window shows the actual TX frequency/amplitude in a waterfall diagram

2 Syntax

2.0.1 Command Syntax

The simplest BASIC program is just a sequence of commands.

- You can write several commands in one line by separating them by a colon. They will be executed sequentially:

  ```
  COMMAND1 : COMMAND2 : COMMAND3
  ```

- You can use comments in your code by starting a line with "//":

  ```
  // **************************
  // ** This is a description **
  // ** It does nothing but   **
  // ** looking good :)       **
  // **************************
  ```

- You can include binary data into the code which is accessible by the "PEEK" command starting at the memory address 2048. To do so, you insert a line a the end of the program code with the word "DATA". In the following lines you write the binary data as decimals:

  ```
  ...some program code...
  ...program ends here

  DATA
  1 23 45 0 255 7 ...
  ```

2.0.2 Variables

There are 26 floating point variables, named from A to Z which can be used freely. In the CHIPSTAR reduced basic only these single letter variables can be used.

2.0.3 Expressions

You can combine variables and absolute numbers into an expression. The operators are +, -, *, /, % (mod), & (and), | (or). Some examples:

```
// Mean of A and B
C= (A+B)/2
```

```
// Division rest of A and B
C= A % B

// Set bit 7 in variable C
C = C | 128
```

2.0.4 Flow control

A flow controlled program is far more powerful than just a sequence of commands. For example, you want to do different things depending in a signal you receive from the ground station, or you can make a decision depending on a sensor value, like turning on magnetic coils when the magnetic field is in the desrired direction. For this purpose you need to be able to jump to different places in your program. There are only five commands which accomplish all this:

- Labeling: If you need to jump to certain parts of your code, you first need to mark these places. You can do this with the "LABEL" command followed by a number of your choice:

  ```
  LABEL 1
   ...some program code..
   ...
  LABEL 2
   ...more code
  ```

- Unconditoinal Jumps: If you want to jump to a labelled program part immidiatiely, use the "GOTO" command, followed by the label number of your choice. The following example will increase the Variable A in an infinite loop:

  ```
  LABEL 1
   A=A+1
  GOTO 1
  ```

- Subroutines: If you want to use a program part frequently, you can jump to that part and automatically return back. To jump and return, you use the "GOSUB" command followed by the label number. GOSUB means "GO to SUBroutine". To return from the subroutine, write the "RETURN" command. This will make the program return to the next command after the last GOSUB. The following program will set the variable B to 49 and the variable C to 9:

  ```
  A=7 : GOSUB 2 : B=A
  A=3 : GOSUB 2 : C=A
  END

  //Compute square of A
  LABEL 2
   A=A*A
  RETURN A,
  ```

- Loops: If you want to repeat a command sequence several times, you can use the "FOR...TO...NEXT" command. It will count up a chosen variable and execute the following command sequence until the variable reaches another chosen value. The following example sums up the numbers from 1 to 10, resulting in S=1+2+3+4+5+6+7+8+9+10=55

```
S=0
FOR A=1 TO 10
  S=S+A
NEXT A
```

- Conditions: You can choose what to do depending on a certain condition. You can use the "IF...THEN...ELSE" statement. The condition is a logical expression which is either true or false.

```
// Set B to absolute value of A
IF A<0 THEN B=0-A ELSE B=A
```

- Delays: You can delay the program flow with the "DELAY" command, followed by a value representing the delay time in 100 microsenconds. The example delays the program for 1 second.

```
DELAY 10000
```

- CPU Speed / Power-Save: If you want to save power, you can make the system run slower. Use the "SPEED" command followed by the speed value. This value can be in the range of 1 to 4, where 1 is slow and 4 is fast. Value 0 has a special function: Radio control and sensor readings don´t work in this mode, but it has the lowest power consumption. At startup, the program is running in speed mode 4. The example slows down the system for one second to save power and speeds it up to maximum power again.

```
SPEED 1 : DELAY 10000 : SPEED 4
```

2.0.5 Commands for Sensors/Actors

- Magnetic Field: Use the "MGET" command to read the magnetic field in Gauss in all three dimensions. MGET is followed by three variables, separated by a colon. The example stores the actual magnetic field values of the X, Y and Z dimension into the variables A, B and C

```
MGET A,B,C
```

- Color Sensor: Use the "CGET" command to read the color light sensor values. CGET is followed by three variables, separated by a colon. The example stores the red, green and blue portion of the light to the variables R, G and B.

```
CGET R,G,B
```

- System Power: Use the "PGET" command to read the remaining system power. PGET is followed by a variable which will contain the system remaining system power in percent. The example stores the system power into the variable A.

  ```
  PGET A
  ```

- Temperature: Use the "TGET" command to read the chipsat´s temperature in °C. TGET is followed by a variable which will contain the temperature value. The example stores the temperature into the variable A.

  ```
  TGET A
  ```

- Electromagnetic Coils: Use the "COIL" command to generate a magnetic field. COIL is followed by three values, representing the magnetic field strength in the X, Y and Z dimension. The values can be in the range from -10 to +10. (Note: The coils´ fields are refreshed every 60Hz)

  ```
  COIL 2,−4,6
  ```

2.0.6 Commands for Radio Transceiver

- Transmitting: Use the "TX" command to control the transmitter. TX is followed by two expressions, the first defines the frequency deviation and the second the transmitting power. The frequency deviation can be in the range from 0 to 31 and the power can be in the range from 0 to 7. The example transmits with full power on the coordinated CHIPSTAR frequency. Note that it takes a few millisenconds for the transceiver to reach the desired power after beeing turned on.

  ```
  TX 16,7
  ```

- Receiving: Use the "RX" command to get the received signal strength. RX is followed by a variable. The example reads the actual signal strength into the variable R. Note that the transceiver needs a few milliseconds to switch from TX to RX mode. Reading the signal directly after a transmission will give a false value.

  ```
  RX R
  ```

- Turn Off: Use the command "ROFF" to turn off the radio.

  ```
  ROFF
  ```

2.0.7 Commands for Memory Access

- Writing to Memory: You have a 2kB RAM memory for write and read access. For wrting, use the command "POKE" followed by the memory address and the byte value. These can be either absolute values or variables. The example writes the value 7 into the memory address 1234.

  ```
  C=1234 : POKE C,7
  ```

- Reading from Memory: Use the command “PEEK” followed by the memory address and the target variable to read a value from memory into that variable. An address higher than 2047 (2kB) will read the data that you specified in the program code with the “DATA” command. The example stores the value in the memory address 1234 into the variable A and the first DATA-section value into the variable B:

  ```
  PEEK 1234,A
  PEEK 2048,B
  ```

2.0.8 Math Functions

Math functions are used by writing the specific command followed by the function parameters and the target variable. The result will be stored in the target variable. For example, to store the sine value of 2 into the variable a, use:

```
SIN 2,A
```

The following math functions are available:

- ACOS <expression> , <variable>
- ASIN <expression> , <variable>
- ATAN <expression> , <variable>
- TPATAN <expression> , <expression>, <variable> (Two parameter ATAN)
- CEIL <expression> , <variable>
- COS <expression> , <variable>
- COSH <expression> , <variable>
- EXP <expression> , <expression>, <variable>
- FABS <expression> , <variable>
- FLOOR <expression> , <variable>
- FMOD <expression> , <expression>, <variable>
- LOG <expression> , <variable>
- LOGDEC <expression> , <variable>
- POW <expression> , <expression>, <variable>
- SIN <expression> , <variable>
- SINH <expression> , <variable>

- SQRT <expression> , <variable>
- TAN <expression> , <variable>
- TANH <expression> , <variable>