

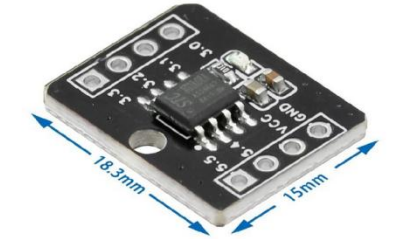
## #nB51 manual

### (very draft version)

The project repository (i.e. binaries, manual, and examples) are available at: <https://github.com/bgolab/nB51/tree/main>

### Wiring

Original breadboard wiring: STC8G1K08A



### Getting started

Flash the latest BASIC hex file (e.g. nB51.hex). Start a terminal emulator e.g. **TeraTerm** (9600,8,N,1). The 8051 will boot.

```
COM5 - Tera Term VT
File Edit Setup Control Window Help
#nB51 v0.1b1 (C) 2025 bg
#n
#for k=1 to 5
#print k, "->", k*k
#next k
#r
1->1
2->4
3->9
4->16
5->25
#
#
#
```

Enter an example BASIC program (you can use Copy & Paste)

```
for k=1 to 5
  print k, ",", k*k
next k
end
```

Use 'c' to see the code, 'r' to run the code. The 'n' clears the program memory.

You can use @N (e.g. @0) to delete Nth line or @N <code> (@0 print 1) to insert the line of code.

Capital and small letters accepted for keywords (i.e. 'CLS' equals 'cls'). For the names small and capital letters are NOT recognized (i.e. 'as' is the same name as 'AS')

*NOTE: If you use TeraTerm and some characters are lost during Copy & Paste operation consider increasing the 'Paste delay per line' in Setup->Additional Settings->Copy and Paste*

### Limits

-VAR / LABEL NAME: 4 chars

-program size: 1024 chars (shared memory with tokens and name!)

### MEMORY MAP

[SOURCE\_CODE][OS CMD BUFFER][TOKENS][HEAP].....[NAMES]

### Program structure

Program Structure	Description/Comments	Example
<code>gosub sub1</code> <code>end</code> <code>subr: [sub1 &lt;code&gt; return</code>	Command 'end' has to be the last line of the main code. Subroutines have to follow the 'end'.	<code>gosub callme</code> <code>end</code> <code>callme: print "Hi!" return</code>

### System commands

Command	Description/Comments	Example
<code>ESC</code>	<del>Stop the running program or skip loading auto.bas when booting</del>	
<code>c</code>	Show code	
<code>r</code>	run code	
<code>n</code>	New – clear VM memory and code	
<code>t</code>	Show Tech Info (tokens, names, mem info)	
<code>g</code>	Compile to tokens (might be disabled to save memory)	
<code>u</code>	Upload tokens from UART The protocol (T sets token, A sets address, R runs: w/o addr set(print 7): 140T 130T 7T R; w/ addr set to 0(print -7): '0A 140T 181T 130T 7T R'	
<code>l</code>	List tests	
<code>0-...</code>	Load N-th tests	
<del><code>l-s</code></del>	<del>Load/Save program from/to EEPROM (auto.bas)</del>	
<del><code>ee, ed</code></del>	<del>EEPROM erase/dump</del>	
<code>@N, @N &lt;code&gt;</code>	@N - delete N-th line; @N <code> - insert <code> before N-th line	<code>@3 – removes 3rd line</code> <code>@4 cls – insert 'cls' before line 4</code>

### The BASIC language

#### **VARIABLES & EXPRESSIONS**

-variable name: letter&digits starting w/ a letter

-expr(arithmetic expression): combination of INT, ops/brackets( +, -, \*, /, %, (, ) ) and INT vars;

-hex integer format supported, e.g. 0xAA

Command	Description/Comments	Example
<code>var=expr</code>	INT var, name=expr, 1 <sup>st</sup> -reference creates var(value=0);	<code>sy=2 + a*20</code>

#### **PROGRAM FLOW CONTROL**

NOTE: The FOR-loop STEP/TO values are evaluated only once, before entering the FOR-loop.

NOTE: The relop operations include: >, <, >=, <=, <>, = (Since nB51 v0.3b1: use '==')

Command	Description/Comments	Example
<code>FOR v=expr TO expr [STEP expr] [code] NEXT var</code>	If STEP[default=1]; INT supported; loop nesting supported	<code>FOR k = 5 TO 1 STEP -1 NEXT k END</code>
<code>IF expr relop expr THEN [code] [ELSE] [code] ENDIF</code>	INT supported; nesting supported;	<code>if a&gt;1 then print "ok" else print "bad" endif</code>
<code>label:</code> <code>GOTO label/var</code>	Label starts with a letter, terminated by colon; up to 8 letter & digits( plus ' _ ');	<code>k=1 again: print k k=k+1</code> <code>if k&lt;5 then goto again: endif</code>
<code>GOSUB label/var</code>	NOTE :Label must be located after END	<code>gosub task0 end</code>
<code>label: RETURN</code>	label: [code] RETURN	<code>task0: print "done" return</code>
<code>END</code>	Last command. GOSUB labels follows END NOTE: END is obligatory if you use GOSUB	<code>End</code>

#### **INPUT, OUTPUT, DATA, MEMORY**

Command	Description/Comments	Example
<code>PRINT expr[, "txt"],</code>	Prints expr, sexpr separated by ','; ';' - skips NEW	<code>PRINT "6/3=", 6/3 (with NEW LINE)</code>

[;]?	LINE?	PRINT 1; (w/o NEW LINE because of ";")
INPUT var,...	Assign int values to (array) var	INPUT "num?", name
DATA const, const READ var, var RESTORE	DATA - INT constants supported READ - assign DATA specified input to vars RESTORE - Reset data pointer	DATA 1, 2 READ v, y
POKE addr,value PEEK(addr)  Since nB51 v0.3b1: SPOKE/SPEEK for SFR XSPOKE/XPEEK for XDATA and XSFR  The following variables provide HEAP addresses: HS – start memory of the HEAP HE – end memory of the HEAP	Memory write/read; hex supported; mem-map(incl. phantom addr): xdata: real addr, sfr: 0xC00+addr; magic: 0xF00+addr	Magix: 0xF00 – contains the memory address of the HEAP  Since nB51 v0.3b1: NO magic addresses, just use physical addresses as per 8051 datasheet

#### TIME (two independent timers: for PAUSE and for GETTICK() settable byGETTICK(N))

Command	Description/Comments	Example
GETTICK(msec)	Sets msec for the GETTICK timer	Gettick(0) (resets the timer for gettick())  SETTICK 0 (resets the timer)
Since nB51 v0.3b1: SETTICK msec		
GETTICK()	Ticks; MAX 32767 msec	a=gettick()
PAUSE msec	Delay (blocking) in msec	PAUSE 2*500

#### LOGICAL

Command	Description/Comments	Example
AND(expr,expr)		PRINT AND(0x3,0xF)
OR(expr,expr)		PRINT OR(0x3,0xF)
NOT(expr)		PRINT HEX\$(NOT(0x0F))

#### GPIO

Command	Description/Comments	Example
PMODE pin,mode	const: IN,OUT,PUP,PDOWN, ADC, PWM	pmode 100, tsensor temp= aread(100)
AREAD(pin)	Read analog pin; pins=?;	pmode 26, adc voltage=aread(26)
AWRITE pin,cycle	PWM duty=cycle/65535,cycle<65535	pmode 22, pwm awrite 22, 16000
DREAD(pin)	Read digital pin	y=15 pmode y,in pmode y,pullup for k=1 to 2 step 0 pause 50 print dread(y) next k
DWRITE pin,value	Write digital pin	pmode 25,out dwrite 25,high pause 3000 dwrite 25,low

#### OLED (OLED 0.91" supported)

NOTE: use 'lref' to refresh the screen as ALL the lxxx command update only the memory buffer

Command	Description/Comments	Example
LPLOT X,Y	Draw point at X, Y	for x=0 to 127 lplot x,fix(15+15*sin(6.28*x/128)) next x lref
LDRAW X0,Y0,X1,Y1	Draw line from X0, Y0 to X1, Y1	ldraw 10, 10, 20,20 lref
LCIRCLE x,y,r	Draw circle at x, y, r	lcircle 15, 15, 10
LPRINT expr,sexpr [AT x,y]	Print expr, sexpr separated by ','	lprint "2+2=", 2+2 AT 10,10 lref

	{AT x, y} (default 0,0)	
<i>LCLS</i>	Clear Screen	<i>lcls lref</i>
<i>LREF</i>	Refresh LCD (copy mem to LCD)	<i>lref</i>

#### **~~TONE SUPPORT~~** *(tested on piezo)*

<b>Command</b>	<b>Description/Comments</b>	<b>Example</b>
<i>TONE pin, freq</i>	Start PWM square signal at freq, duty 50% on a pin; freq>10Hz; non-blocking;	<i>TONE 18, 440</i>
<i>NOTONE pin</i>	Stop generating the signal on a pin	<i>NOTONE 18</i>

#### **~~SERVO SUPPORT~~** *(tested on Tower Pro Sg90 where the pulse is between 500-2400 usec) - EXPERIMENTAL*

<b>Command</b>	<b>Description/Comments</b>	<b>Example</b>
<i>SERVO pin, angle [pulsemin, pulsemax]</i>	Enable servo on a pin, set the angle (0...180 deg); optional: min/max pulse in usec (default: 500/2000usec, allowed min/max values: 500-2400usec)	<i>SERVO 19, 90</i> <i>SERVO 19, 9, 500, 2400</i>
<i>NOSERVO pin</i>	Disable servo on a pin	<i>NOSERVO 19</i>

#### **~~I2C/UART SUPPORT~~**

~~~xREAD/xWRITE are blocking (waits for the data forever); non-blocking by checking if the fifo(xREADABLE/xWRITEABLE)~~