

## My first 8052 BASIC Project

PIC is a nice little chip, but it does have it's limitations. However, there is a relatively cheap way to have a BASIC programmable microcontroller system. Some time in early 80-ies Intel produced a mask-programmed 8052 MCU which contained a floating-point Basic interpreter with lots of interesting features, including built-in support for EPROM storage of BASIC programs. There is a plethora of information scattered on the web about this chip, but no one place provides a complete set of instructions to get you started. This project is my attempt to provide just that.

So, first of all, you need a 80C52 or compatible microcontroller. There are several producers, I used Atmel's AT89C52 which is FLASH based, so it can be reprogrammed (as opposed to 80C52 which is OTP part). Any 89C52 or 87C52 part will do fine for experimenting, but if you do not have a programmer for 8051 family of MCU-s, Atmel also has a AT89S8252 (8052 compatible chip with 2K of EEPROM built in) with a serial programming interface which makes building a programmer much easier.

Once you have the controller, you need a board to stick it in. I built mine from scavanged parts and it has serial port, full data and address bus buffering, 32K static RAM and 8K or 16K of EPROM on board. I'm feeding this beast from standard PC power supply, using 12V for EPROM programming directly. National Semiconductor's 12.5V VPP parts seem to program just fine at 12V. [Click here to see the schematics.](#)

**Bear in mind that this was my first 8052-BASIC project, so the hardware is quite a bit overengineered on some parts while it misses some stuff that should be implemented in other parts. A better, lower chip count and fully implemented 8052-BASIC core is available [here](#).**

Just a few short remarks:

- The external code and data address spaces are combined into one. Code space is 8K of internal ROM followed by shared memory space which starts with RAM from 0, and continues with ROM from 8000H. The address space from C000H is not used on board, and is available for I/O or ROM expansion.
- Resistor pack RN2 is not required, but it does help to keep the data bus in order.
- 74245 chips can cause serious glitches on power and signal lines, make sure you have bypass caps near them. I initially used Soviet parts and was unable to get them to behave at all.
- It seems tempting to use 27256 EPROM to provide full 32K of ROM storage as well, but as the PGM signal is shared with one of the chip selects, the programming timings will be impossible to sort out. I gave up after several days of head scratching and just divided the ROM space in two, decoding only first half on the board
- JP3 allows to switch off internal ROM when center pin is grounded.
- JP5 selects between 2764 (center pin connected to A13) and 27128 (center pin

grounded) EPROM

- I used 40 pin IDE connector for external bus, because it is small, connection cables are readily available, and I had a broken motherboard with two IDE connectors on it :)
- I built my version on one-sided perfboard, using point-to-point soldered wires (from phone cable) on component side. It's not the most convenient and best-looking, but it works.
- My version also has a RESET button in parallel with C7.

Now, once you have the board download the [interpreter source and binary code](#), program the MCU, stick it into socket on your board, hook up the PC with straight modem cable, and power on. Wait a second or two, then press space on the keyboard. If you have everything right, the interpreter will detect the baud rate you are using and output

```
*MCS-51(tm) BASIC V1.1*
```

```
READY
```

```
>
```

When you get this far, things are looking good. Enter **PRINT MTOP** at prompt, you should get 32767 if you used 32K RAM. To verify EPROM circuit, try entering **PROG1** at prompt, if you do not get a programming error then you just wrote the baud rate to use into EPROM, and BASIC will not wait for keypress next time you start it. Before going on to literature and references, here are two programs that I wrote, the first of them has proved very valuable.

You will usually work in RAM, but when you try to make a more permanent copy of your code, you enter **PROG** (or **FPROG**) at prompt to transfer the code into EPROM. BASIC will give you a sequence number of the program just stored and this number is the only way to refer to the stored program. To improve things, I always write this program into EPROM first:

```
10   REM Basic ROM directory
20   ADR=8010H : PN=1
30   IF XBY(ADR)<>55H THEN END
40   ADR=ADR+1 : BA=ADR
50   IF XBY(ADR)=1 THEN GOTO 80
60   ADR=ADR+XBY(ADR): GOTO 50
80   GOSUB 100: PRINT : PN=PN+1 : ADR=ADR+1: GOTO 30
100  PRINT "PRG",PN,"from ", : PH1. BA, : PRINT " to ", : PH1. ADR,
110  PRINT " (",ADR-BA,"bytes)",
120  NB=XBY(BA)-2: IF NB<4 THEN RETURN
130 IF XBY(BA+3)<>96H THEN RETURN
140  FOR XX=BA+4 TO BA+NB:PRINT CHR(XBY(XX)),: NEXT : RETURN
```

When run, this program lists all programs stored into EPROM by giving code start and end address, length of the code, and most importantly, if the first program line is a comment, it will print out the text after REM keyword. The example output might look something like that:

```
PRG 1 from 8011H to 8134H (291 bytes) Basic ROM directory
PRG 2 from 8136H to 81CDH (151 bytes) ROM dump
```

Now, when I power the board on, I just enter RROM 1 and I have a complete listing of all stored programs. Much better. The second useful piece of code is memory dump program:

```
10 REM ROM dump
20  ADR=8000H
30  PRINT : GOSUB 70: PRINT : PRINT "More?",
40  A=GET : IF A=0 THEN GOTO 40
50  IF A<>27 THEN GOTO 30
60  END
70  FOR XA=ADR TO ADR+255: X=XA.AND.0FH: IF X<>0 THEN GOTO 90
80  PRINT : PH1. XA,
90  PH0. XBY(XA),: NEXT : ADR=XA: RETURN
```

It starts at 8000H and dumps pages until you press ESC at the prompt. The data is taken from external data memory space, you need to change XBY() to CBY() to read internal ROM, or DBY() to read internal register space.

## IDE Interface

While 16K of EPROM is a lot compared to PIC memory space, it is nothing in today's information overload. For serious data logging, and just for fun, here is a [schematic drawing for IDE drive interface](#). If you are willing to waste 50% of the disk space, and stretch the IDE interface specs a bit, you can leave out all IC-s except IC1, and connect LED directly to /DASP line. You will be able to read and write only lower 8 bits of the data words, but otherwise it will work just fine, because all control registers are 8-bit. However, I wanted to access all bytes, so I built extra register to capture high byte of the data word. IC1 decodes IDE command block registers at 0E000H-0E007H, IDE control block registers at 0E008H-0E00FH, and the extra data register at 0E010H.

I am tinkering with a software now, so far I have

- [Program that reads IDE drive parameters and displays them](#)
- [Program that allows you to play with commands \(standby, sleep, reset, diagnostics, recalibrate\). It also reads and dumps sector contents](#)

As usual, once I got the hardware working and concept proven I lost interest in the project so this is what I have and there is but a very little hope that there will be something more coming under this project..

## Standards

- [ATA3R6](#) document documents the physical interface and HDD command set
- [ATAPI-4](#) document is needed if you want to play with CD-ROM drives

## 8052 reference sites

- <http://www.8052.com/> is a good starting point
- [UST Research Inc. Download area](#) contains a lot of goodies
- [BASIC-52,8051,8052, SOL, KIM, Tiny Basic code archive](#)
- [SDCC - Freeware, Optimizing C Compiler for 8051](#)

## Useful literature

- [BASIC-52 PROGRAMMING from Systronix Inc.](#) is a sort of replacement for original Intel 8052AH-BASIC user manual. No circuits.
- [The Microcontroller Idea Book by Jan Axelson](#), I have not yet received mine.
- The Best of Ciarcia's [Circuit Cellar](#) by Steve Ciarcia, ISBN 0-07-011025-5 contains lots of useful information and shcematics examples for 8052 BASIC and other widgets.
- Real hard-code hackers will, of course, just read a source code :)

本文内容来自互联网，著作权归原作者所有。由电子零件城（<http://www.epcity.com/>）整理并制作成 PDF 文件，仅供个人学习之用，不得用于任何商业目的，否则后果自负。如果您认为本 PDF 文件侵犯了您的任何权利，请来信 [epcity@epcity.com](mailto:epcity@epcity.com) 通知，本站立即删除。

搜集整理：电子零件城-笨笨兔（QQ：154502842） 2004-04-10