BIG 8051 Microcontroller

User Manual

James Kelly and David Cain

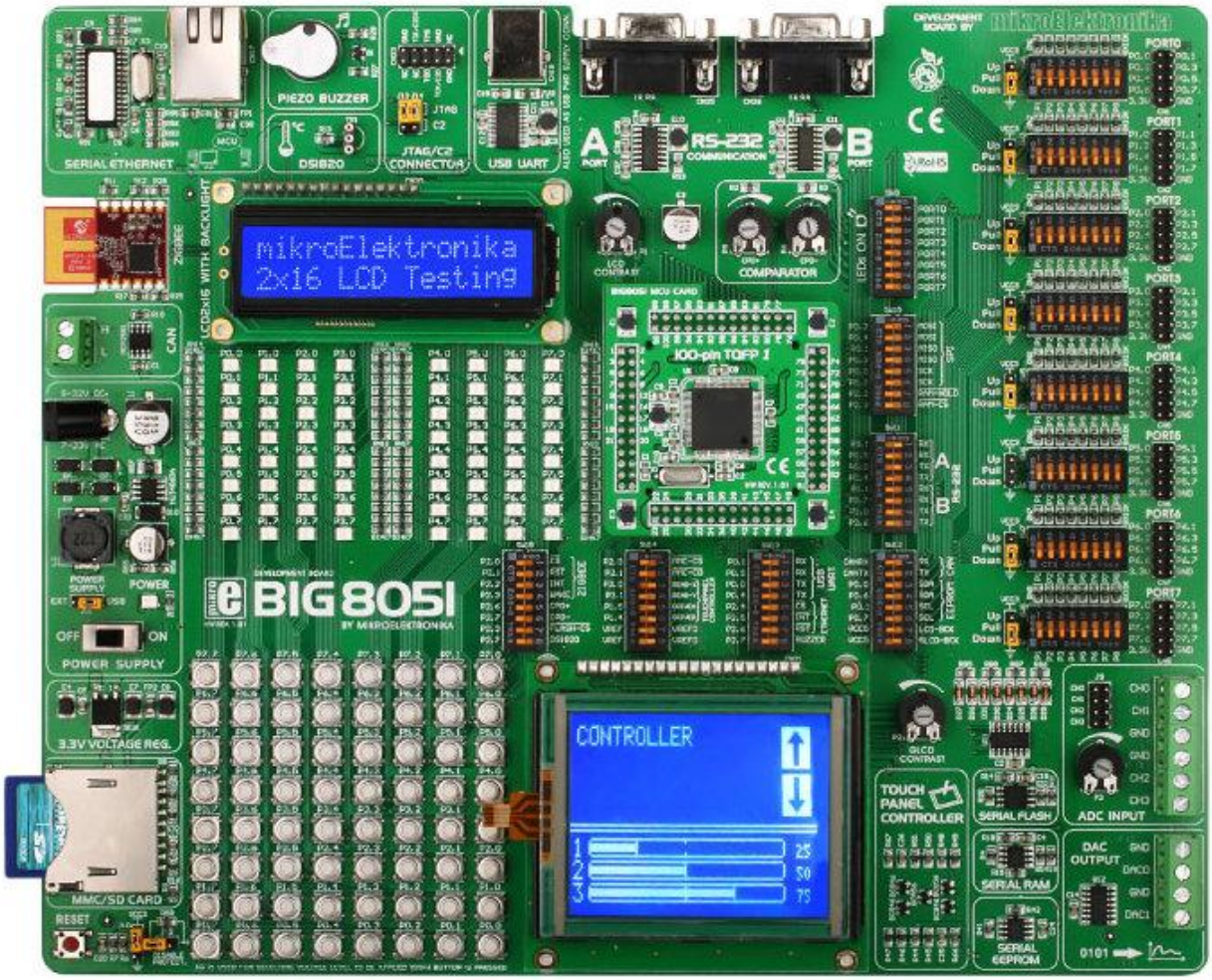
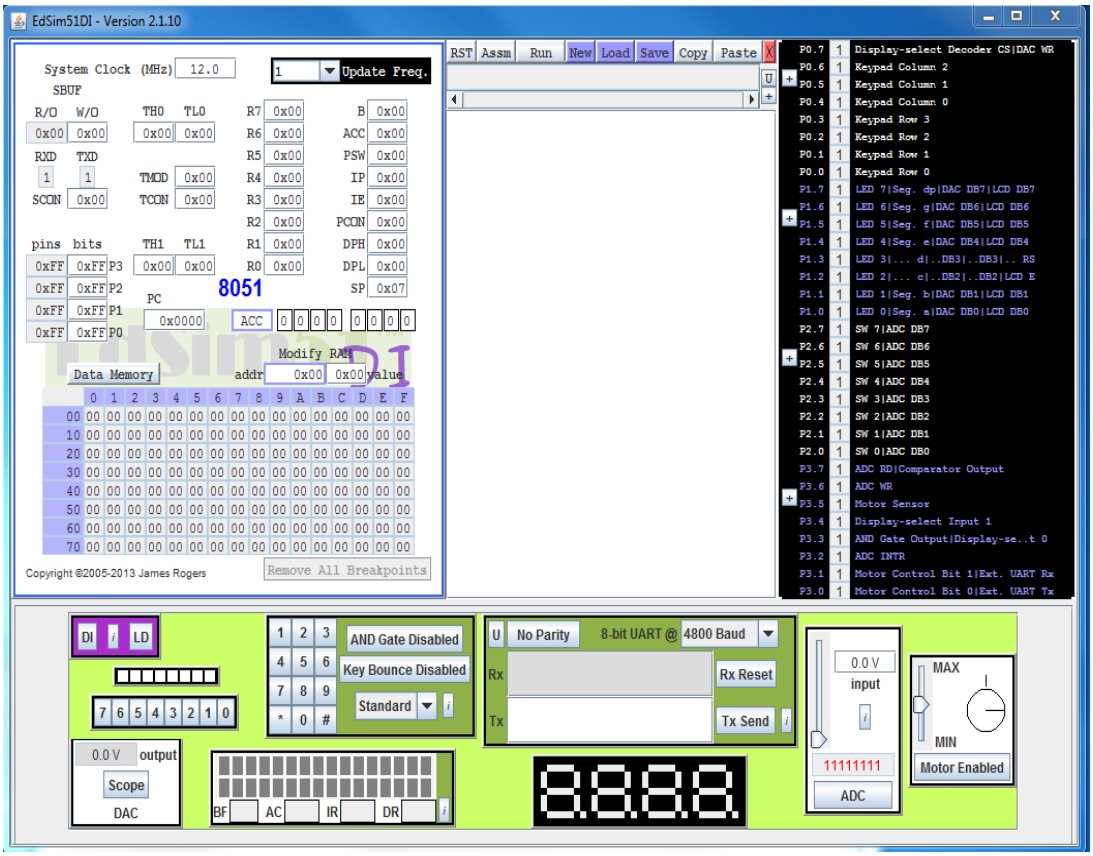


Table of Contents:

How to set up the 8051 Simulator

1) Make sure you have Java installed, you can download it at http://java.com/en/download/index.jsp
2) Run the installer you downloaded
    - Make sure to *uncheck* installing the Ask toolbar
3) Download http://www.edsim51.com/8051simulator/edsim51di.zip
    - If the software has been updated since this writing, the newer version can be downloaded at http://www.edsim51.com/
4) Unzip the file
5) Inside the edsim51di folder, run edsim51di.jar



- In the middle window are your main controls where you will be able to write and execute your code
- The bottom window allows you to manipulate virtual inputs to the 8051, and see outputs on its various LCDs and LEDs
- On the left is a representation of what is stored in the memory and registers of your 8051
    - Note: To clear some of these requires closing and re-opening the simulator
- In the black box on the right you'll find a reference for how to access various "hardware" elements of the simulated 8051, such as switches and LEDs, and whether each currently sees a logic 1 or logic 0

Coding the simulator

1) Above the assembly code window, you will see the toolbar depicted below. Either select "New" to begin a new assembly program or "Load" to import an existing one

2) Once a program has been written or loaded, select either "Assm" or "Run" to execute it. If the simulator identifies any errors in the code, the program will not run, and a message will appear indicating where the problem originated
    a. "Assm" is used to manually step through the code and observe individual steps. Once it has been clicked, the button will change to "Step" and must be pressed repeatedly to advance the program one line of code at a time
    b. "Run" will run the program all the way to the end without stopping. At any point, the user may pause the program by clicking the button again
3) If you wish to stop the program from running entirely, select "RST". This will end the program and allow you to make modifications or simply begin the program again
4) When you are finished, you can save your code as a .asm file by clicking save, which will allow you to import your work at a later time

Sample Code

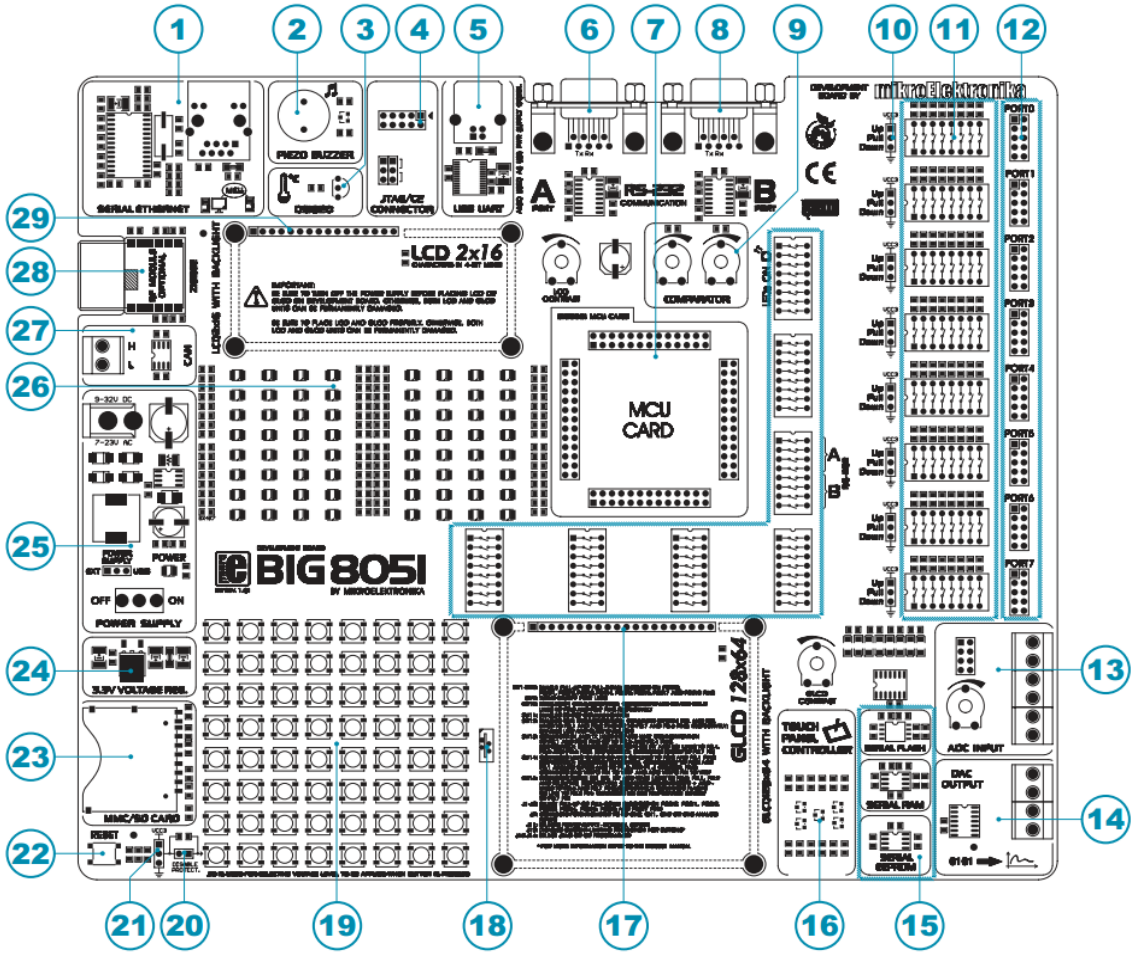
1) A simple program to display binary values from 0 to 255 on the LEDs (note, since the simulator considers logic 0 to be "lit" for the LEDs, the program must decrement instead of increment)

```
main:
       DEC P1 ;decrement instead of increment because logic 0 lights the LEDs
       JMP main
```

2) A Program to mirror the 8 push buttons onto the corresponding LEDs

```
main:
       MOV P1, P2 ;move values on buttons into the LEDs
       JMP main
```

Physical Components



1) Ethernet Connection
2) Piezo Buzzer
3) Temperature sensor connection
4) Programmer connection
5) USB connection
6) RS-232A Module
7) MCU card connector
8) RS-232B Module
9) Comparator
10) Jumpers used to select pull-up/pull-down resistors
11) DIP switches (turn pull-up/pull-down resistors on or off)
12) Input/Output Ports
13) Analog/Digital Converter Input
14) Digital/Analog Converter Output
15) Memory Modules
16) Touch panel controller
17) Connector for GLCD display
18) Connector for touch panel

19) Push Buttons
20) Jumper to shorten protective resistor
21) Jumper to select logic state of push buttons
22) Reset button
23) MMC/SD card connector
24) 3.3V Voltage regulator
25) Power supply module
26) LEDs
27) CAN module
28) ZigBee module
29) Connector for LCD display

Setting up the Board

- To power the board from a USB connection, ensure that the orange jumper above the power switch (component 25 in the schematic above is set to “USB”
- Set jumpers at point 4 to JTAG
- Connect the provided Silicon Laboratories Debug Adaptor (pictured below) to the board at point 4, right above the jumpers used in the previous step

- (Optional) – the following steps are used to activate a few components commonly used in the examples
    - Each switch on SW9 powers a column of LEDs
    - Switch 8 of panel SW13 powers the buzzer
    - Small screen backlight is controlled by switch 7 of panel SW12
    - Large screen backlight is controlled by switch 8 of panel SW12

For more information on each individual component, including which switches on the board need to be flipped to activate them, see the full manual provided by mikroelectronica at http://www.mikroe.com/downloads/get/1460/big8051_manual_v100.pdf

Software

The CD included with the board provides a link to a website where the requisite programs may be downloaded, along with a number of simple sample programs

- Install the Silicon Labs IDE
    - http://www.silabs.com/Support%20Documents/Software/8BitToolsInstaller-web.exe
- Download example files and extract them to a location of your choice
    - http://www.mikroe.com/eng/downloads/get/1457/big8051_mikroc_v100.zip
- Choose Options/Connection Options/JTAG
- Go to Debug/Connect or click the Connect button next to the download button
- Hit the Download button
- Browse for a .hex file in the examples folder and hit Download
- Hit the green Run button

At this time, the BIG8051 seems to only accept .hex files. Any attempt to download a different file type to the board is met with an unsupported file type error

Appendix A: Sample Project

Below is a fully functional program that may be run on the simulator. The program is designed to control the water level in a bucket used as part of a hydroponic garden. Switches 1, 2 and 3 on the board serve as stand ins for a trio of float sensors that will activate when the water level reaches a set height. The first four LEDs on the board correspond to the four devices that the system needs to control. LED 0 represents the input valve, LED 1 represents an aerator that needs to run for 5 seconds each time a switch is flipped for the first time. LEDs 3 and 4 represent the drain valve that will allow the water out (3), and a light that is required to be turned on while the bucket empties (4). Recall that, as with the earlier example programs, logic 0 is required to "light" the LEDs, so CLR is used to light them instead of SETB.

```
main:
       JNB F0, initialize ; F0 is a flag that is set when initialization has been run
once
       SETB P1.1 ; ensure that the aerator is off
       JB P0.3, drain ;ifP0.3 is set, the bucket has finished filling and should begin to
drain
       CLR P1.0 ; Begin filling the bucket
       JNB P2.3, Switch3 ; is the third switch tripped? (note that the last switch must
be the first checked so as to not get stuck)
       JNB P2.2, Switch2 ; Is the second switch tripped?
       JNB P2.1, Switch1 ;Is the first switch tripped?
       JMP main ; continuously loop

Switch1:
       JB P0.0, main ; If this function has been executed before return to main
       SETB P0.0 ; set as flag that this function has been executed
       JMP aerate ; jump if this is the first time the switch read as pressed

Switch2:
       JB P0.1, main  ; If this function has been executed before return to main
       SETB P0.1 ; set as flag that this function has been executed
       JMP aerate ; jump if this is the first time the switch read as pressed

Switch3:
       JB P0.2, main ; If this function has been executed before return to main
       SETB P0.2 ; set as flag that this function has been executed
       SETB P0.3 ; set to indicate that bucket is full
       JMP aerate ; jump if this is the first time the switch read as pressed

aerate:
       SETB P1.0 ; stop filling
       CLR P1.1 ; start aerator
       JMP delay ; jump to delay

delay:
       MOV R0,#20 ; set loop count to 20
       JMP countdown ; jump to countdown


countdown:
       DEC R0 ; Decrement R0
    DJNZ R0,countdown ; keep looping until R0 hits 0
```

```
	JMP main ; return to main

drain:
	CLR P1.2 ; open drain valve
	CLR P1.3 ; turn on light
	JNB P2.1, drain	; keep looping until the water drops below the first sensor
	SETB P1.2 ; close drain valve
	SETB P1.3 ; turn off light
	JMP initialize ; Reset the flags so the program can restart automatically

initialize: ; This functions sets flags to ensure certain functions are only run once
(aerate once per switch,  drain once)
	CLR P0.0 ; This will be set when the first sensor is tripped
	CLR P0.1 ; This will be set when the second sensor is tripped
	CLR P0.2 ; This will be set when the third sensor is tripped
	CLR P0.3 ; This will be set after the third switch is tripped to confirm that the
bucket is full
	SETB F0 ; Set to confirm that this function has been run once
	JMP main ; return to main
```

Appendix B: External links

For convenience, all links found in the manual above have been collected here

The requisite version of Java for the simulator can be found here:
http://java.com/en/download/index.jsp

The simulator itself is found here:
http://www.edsim51.com/8051simulator/edsim51di.zip

Or here if a newer version has been made available:
http://www.edsim51.com/

The original user manual for the BIG8051 board may be found here:
http://www.mikroe.com/downloads/get/1460/big8051_manual_v100.pdf

The Silicon Labs IDE can be found her
http://www.silabs.com/Support%20Documents/Software/8BitToolsInstaller-web.exe

Sample .hex files to run on the board can be downloaded here:
http://www.mikroe.com/eng/downloads/get/1457/big8051_mikroc_v100.zip

Additionally, the links below might provide additional assistance getting started with the simulator

Additional example files for the simulator may be found here:
http://www.edsim51.com/examples/edsim51examples.zip

A video tutorial of the simulator board can be found here:
http://www.youtube.com/watch?v=EGf68G4TUkg