

Documentation for Single Board Heater System

Rakhi R
Rupak Rokade
Inderpreet Arora
Kannan M. Moudgalya
Kaushik Venkata Belusonti



IIT Bombay
June 10, 2014

Contents

List of Scilab Code	2
1 Introduction	3
2 Block diagram explanation of Single Board Heater System	4
2.1 Microcontroller	4
2.1.1 PWM for heat and speed control	5
2.1.2 Analog to Digital conversion	7
2.2 Instrumentation amplifier	7
2.3 Communication	8
2.3.1 Serial port communication	9
2.3.2 Using USB for Communication	9
2.4 Display and Resetting the setup	9
3 Performing a Local Experiment on Single Board Heater System	13
3.1 Using SBHS on a Windows OS	13
3.1.1 Installing Drivers and Configuring COM Port	14
3.1.2 Steps to Perform a Local Experiment	16
3.2 Accessing Single Board Heater System on a Linux System	20

List of Scilab Code

Chapter 1

Introduction

THIS IS INTRODUCTION.

Chapter 2

Block diagram explanation of Single Board Heater System

Figure 2.1 shows the block diagram of 'Single Board Heater System' (SBHS). Microcontroller ATmega16 is used at the heart of the setup. The microcontroller can be programmed with the help of an In-system programmer port (ISP) available on the board. The setup can be connected to a computer via two serial communication ports namely RS232 and USB. A particular port can be selected by setting the jumper to its appropriate place. The communication between PC and setup takes place via a serial to TTL interface. The μ C operates the Heater and Fan with the help of separate drivers. The driver comprises of a power MOSFET. A temperature sensor is used to sense the temperature and feed to the μ C through an Instrumentation Amplifier. Some required parameter values are also displayed along with some LED indications.

2.1 Microcontroller

Some salient features of ATmega16 are listed below:

1. 32 x 8 general purpose registers.
2. 16K Bytes of In-System Self-Programmable flash memory
3. 512 Bytes of EEPROM
4. 1K Bytes of internal Static RAM (SRAM)

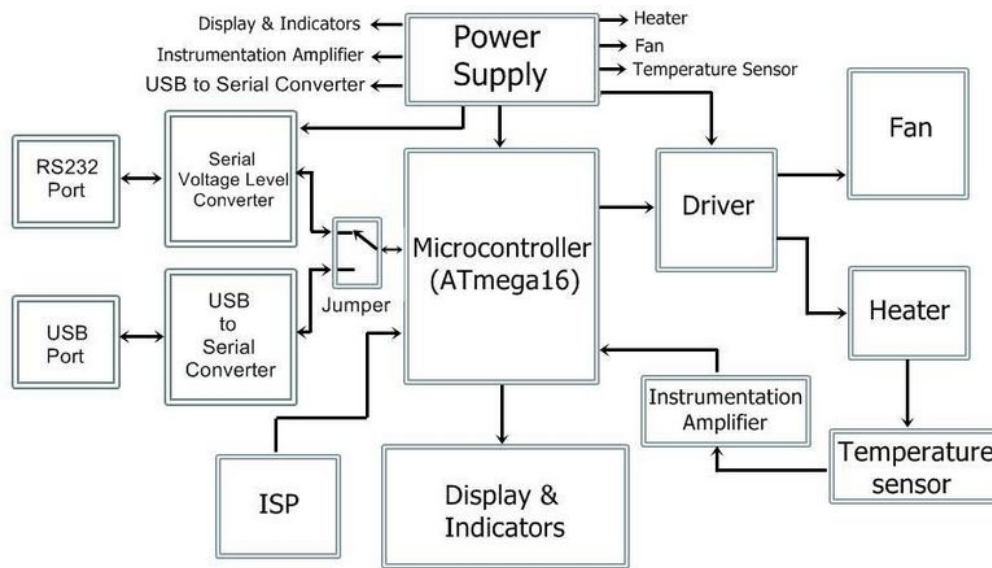


Figure 2.1: Block Diagram

5. Two 8-bit Timer/Counters
6. One 16-bit Timer/Counter
7. Four PWM channels
8. 8-channel,10-bit ADC
9. Programmable Serial USART
10. Up to 16 MIPS throughput at 16 MHz

Microcontroller plays a very important role. It controls every single hardware present on the board, directly or indirectly. It executes various tasks like, setting up communication between PC and the equipment, controlling the amount of current passing through the heater coil, controlling the fan speed, reading the temperature, displaying some relevant parameter values and various other necessary operations.

2.1.1 PWM for heat and speed control

The Single Board Heater System contains a Heater coil and a Fan. The heater assembly consists of an iron plate placed at a distance of about 3.5mm from a

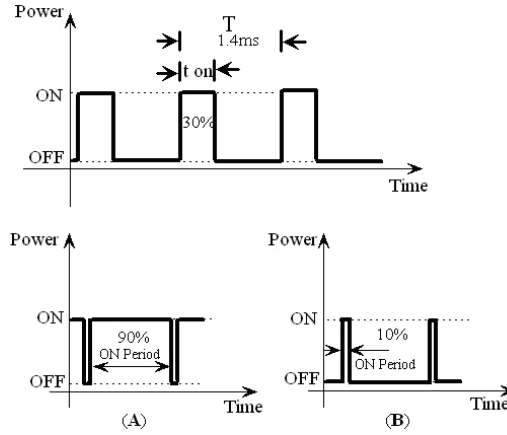


Figure 2.2: Pulse Width Modulation (A): On time is 90% of the total time period, (B): ON time is 10% of total time period

nichrome coil. When current passes through the coil it gets heated and in turn raises the temperature of the iron plate. Altering the heat generated by the coil and also the speed at which the fan is operated, are the objectives of our prime interest. The amount of power delivered to the Fan and Heater can be controlled in various ways. The technique used here is called as PWM (abbreviation of Pulse Width Modulation) technique. PWM is a process in which the duty cycle of the square wave is modulated.

$$\text{Duty cycle} = \frac{T_{ON}}{T} \quad (2.1)$$

Where T_{ON} is the ON time of the wave corresponding to the HIGH state of logic and T is the total time period of the wave. Power delivered to the load is proportional to T_{ON} time of the signal. This is used to control the current flowing through the heating element and also speed of the fan. An internal timer of the microcontroller is used to generate a square wave. The ON time of the square wave depends on a count value set in the internal timer. The pulse width of the waveform can be varied accordingly by varying this count value. Thus, PWM waveform is generated at the appropriate pin of the microcontroller. This generated PWM waveform is used to control the power delivered to the load (Fan and Heater).

A MOSFET is used to switch at the PWM frequency which indirectly controls the power delivered to the load. A separate MOSFET is used to control the power delivered to each of the two loads. The timer is operated at 244Hz.

2.1.2 Analog to Digital conversion

As explained earlier, the heat generated by the heater coil is passed to the iron plate through convection. The temperature of this plate is measured by using a temperature sensor AD590.

Some of the salient features of AD590 include:

1. Linear current output: $1\mu\text{A/K}$
2. Wide range: -55°C to $+150^\circ\text{C}$
3. Sensor isolation from the case
4. Low cost

The output of AD590 is then fed to the microcontroller through an Instrumentation Amplifier. The signal obtained at the output of the Instrumentation Amplifier is in analog form. It should be converted in to digital form before feeding as an input to the microcontroller. ATmega16 features an internal 8-channel , 10 bit successive approximation ADC (analog to digital converter) with $0-V_{cc}$ (0 to V_{cc}) input voltage range, which is used for converting the output of Instrumentation Amplifier. An interrupt is generated on completion of analog to digital conversion. Here, ADC is initialize to have $206\mu\text{s}$ of conversion time . Digital data thus obtained is sent to the computer via serial port as well as for further processing required for the on-board display.

2.2 Instrumentation amplifier

Instrumentation Amplifiers are often used in temperature measurement circuits in order to boost the output of the temperature sensors. A typical three Op-Amp Instrumentation amplifier is shown in the figure 2.3. The Instrumentation Amplifiers (IAs) are mostly preferred, where the sensor is located at a remote place and therefore is susceptible to signal attenuation, due to their very low DC offsets, high input impedance, very high Common mode rejection ratio (CMRR). The IAs have a very high input impedance and hence do not load the input signal source. IC LM348 is used to construct a 3 Op-Amp IA. IC LM348 contains a set of four Op-Amps. Gain of the amplifier is given by equation 2.2

$$\frac{V_o}{V_2 - V_1} = \left\{ 1 + \frac{2R_f}{R_g} \right\} \frac{R_2}{R_1} \quad (2.2)$$

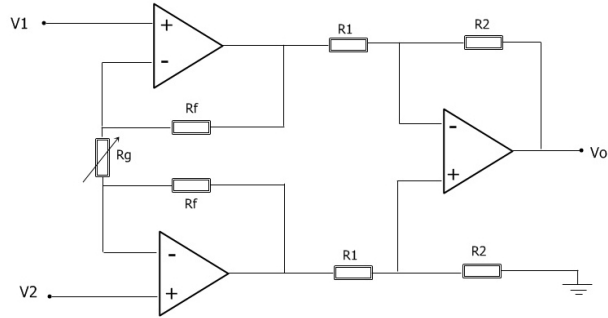


Figure 2.3: 3 Op-Amp Instrumentation Amplifier

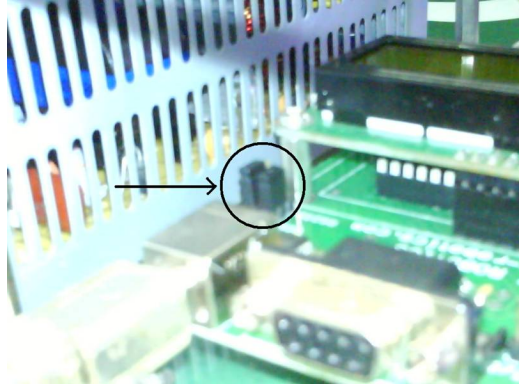


Figure 2.4: Jumper arrangement

The value of R_g is kept variable to change the overall gain of the amplifier. The signal generated by AD590 is in $\mu\text{A}/^\circ\text{K}$. It is converted to $\text{mV}/^\circ\text{K}$ by taking it across a $1\text{ K}\Omega$ resistor. The $^\circ\text{K}$ to $^\circ\text{C}$ conversion is done by subtracting 273 from the $^\circ\text{K}$ result. One input of the IA is fed with the $\text{mV}/^\circ\text{K}$ reading and the other with 273 mV. The resulting output is now in $\text{mV}/^\circ\text{C}$. The output of the IA is fed to the microcontroller for further processing.

2.3 Communication

The set up has the facility to use either USB or RS232 for communication with the computer. A jumper is been provided to switch between USB and RS232. The voltages available at the TXD terminal of microcontroller are in TTL (transistor-



Figure 2.5: RS232 cable

transistor logic). However, according to RS232 standard voltage level below -5V is treated as logic 1 and voltage level above +5V is treated as logic 0. This convention is used to ensure error free transmission over long distances. For solving this compatibility issue between RS232 and TTL, an external hardware interface IC MAX202 is used. IC MAX202 is a +5V RS232 transreceiver.

2.3.1 Serial port communication

Serial port is a full duplex device i.e. it can transmit and receive data at the same time. ATmega16 supports a programmable Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART). Its baud rate is fixed at 9600 bps with character size set to 8 bits and parity bits disabled.

2.3.2 Using USB for Communication

After setting the jumper to USB mode connect the set up to the computer using a USB cable at appropriate ports as shown in the figure 2.8. To make the setup USB compatible, USB to serial conversion is carried out using IC FT232R. Note that proper USB driver should be installed on the computer.

2.4 Display and Resetting the setup

The temperature of the plate, percentage values of Heat and Fan and the machine identification number (MID) are displayed on LCD connected to the microcon-

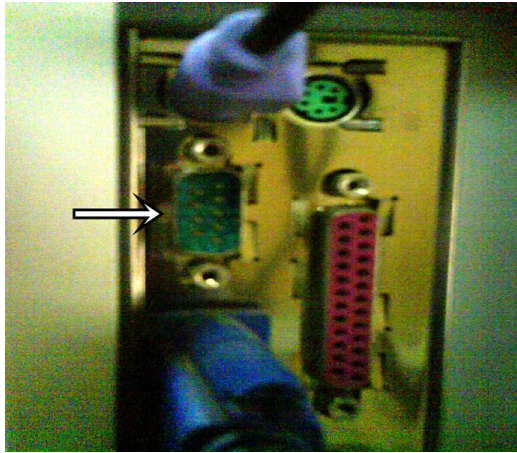


Figure 2.6: Serial port



Figure 2.7: USB communication

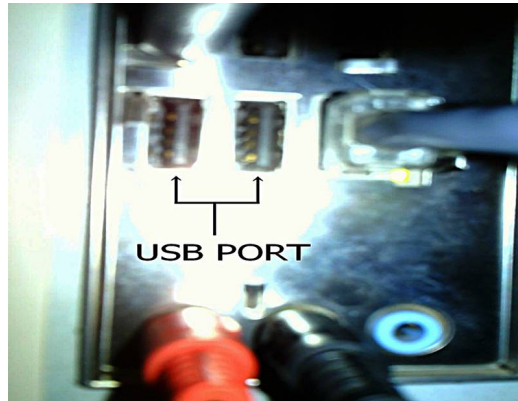


Figure 2.8: USB PORT



Figure 2.9: Display

troller. As shown in figure 2.9, numerals below TEMP indicate the actual temperature of the heater plate in $^{\circ}\text{C}$. Numerals below HEA and FAN indicate the respective percentage values at which heater and fan are being operated. Numerals below MID corresponds to the device identification number. The set up could be reset at any time using the reset button shown in figure 2.10. Resetting the setup takes it to the standby mode where the heater current is forced to be zero and fan speed is set to the maximum value. Although these reset values are not displayed on the LCD display these are preloaded to the appropriate units.

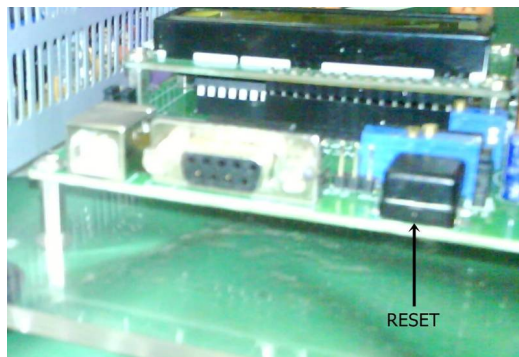


Figure 2.10: Reset

Chapter 3

Performing a Local Experiment on Single Board Heater System

This chapter explains the procedure to use Single Board Heater System with Scilab. An open loop experiment, step test is used for demonstrating this procedure. The process however remains the same for performing any other experiment explained in this document, unless specified otherwise.

Hardware and Software Requirements

For working with the Single Board Heater system, following components are required:

1. SBHS with USB cable and power cable.
2. PC/Laptop with Scilab software installed. Scilab can be downloaded from:
<http://www.scilab.org>
3. FTDI Virtual Com Port driver corresponding to the OS on your PC. Linux users do not need this. The driver can be downloaded from:
<http://www.ftdichip.com/Drivers/VCP.htm>

3.1 Using SBHS on a Windows OS

This section deals with the procedure to use SBHS on a Windows Operating System. The Operating System used for this document is Windows 7, 64-bit OS.

If you are using some other Operating System or the steps explained in section 3.1.1 are not sufficient to understand, refer to the official document available on the main ftdi website at www.ftdichip.com. On the left hand side panel, click on 'Drivers'. In the drop-down menu, choose 'VCP Drivers'. Then on the web page, click on 'Installation Guides' link. Choose the required OS document. We would now begin with the procedure.

3.1.1 Installing Drivers and Configuring COM Port

After powering ON the SBHS and plugging in the USB cable to the PC (check the jumper settings on the board are set to USB communication) for the very first time, the Welcome to Found New Hardware Wizard dialog box will pop up. Select the option Install from a list or specific location. Choose Search for best driver in these locations. Check the box Include this location in the search. Click on Browse. Specify the path where the driver is copied as explained earlier (item no.3) and install the driver by clicking Next. Once the wizard has successfully installed the driver, the SBHS is ready for use. Please note that this procedure should be repeated twice.

Now, the communication port number assigned to the computer port to which the Single Board Heater System is connected, via an RS232 or USB cable should be identified. For identifying this port number, right click on My Computer and click on Properties. Then, select the Hardware tab and click on Device Manager. The list of hardware devices will be displayed. Locate the Ports(COM & LPT) option and click on it. The various communication ports used by the computer will be displayed. If the SBHS is connected via RS232 cable, then look for Communications Port(COM1) else look for USB Serial Port. For RS232 connection, the port number mostly remains COM1. For USB connection it may change to some other number. Note the appropriate COM number. This process is illustrated in figure 3.1

Sometimes the COM port number associated with the USB port after connecting a USB cable may be greater than 9. Since the serial tool box can handle only single digit port number (upto 9), it is necessary to change this COM port number. Following is the procedure to change the COM port number. Double click on the name of the particular port. Click on Port Settings tab and then click on Advanced. In the COM port number drop-down menu, choose the port number to any other number less than 10. This procedure is illustrated in figure 3.2. After following the procedure the COM port number can be verified as described earlier.

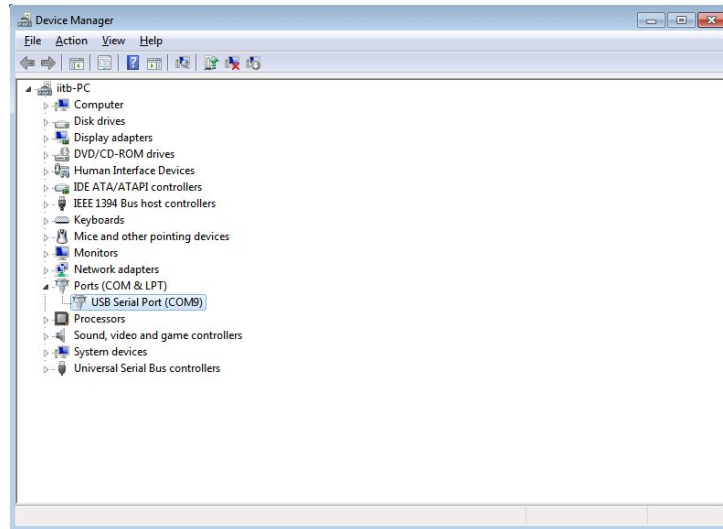


Figure 3.1: Checking Communication Port number

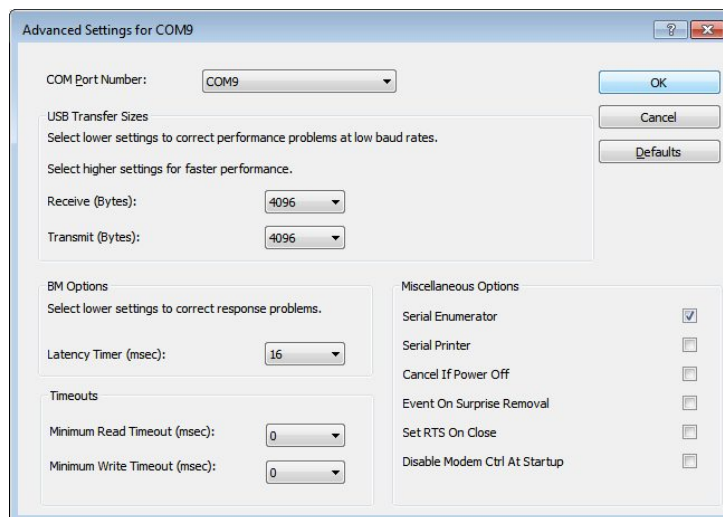


Figure 3.2: Changing Com port number

3.1.2 Steps to Perform a Local Experiment

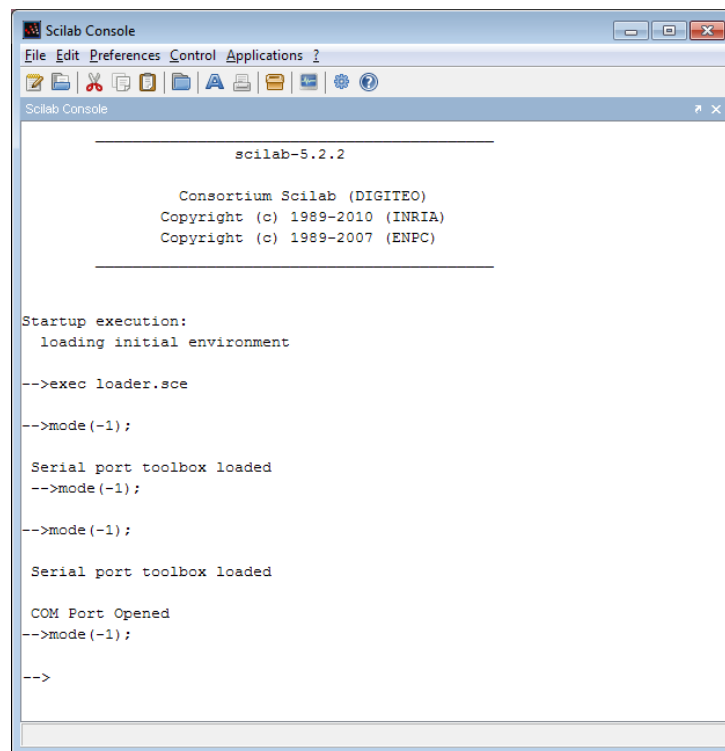
Go to `sbhs.os-hardware.in/downloads`. Let us take a look at the downloads page. There are two versions of the scilab code. One which can be used with SBHS locally i.e. when you are physically accessing SBHS using your computer and another to be used for accessing SBHS virtually. This section expects you to download the local version. On extracting the file that you will download, you will get a folder `Scilab`. This folder will contain many folders named after the experiment. You will also find a directory named `common-files`. We are going to use the folder named `Step_test`.

1. Launch Scilab from start menu or double click the Scilab icon on the desktop (if any). Before executing any scripts which are dependent on other files or scripts, one needs to change the working directory of Scilab. This will set the directory path in Scilab from where the other necessary files should be loaded. To change the directory, click on file menu and then choose "Change directory". This can also be performed by typing `cd<space>folder path`. Change the directory to the folder `Step_test`. There is another quicker way to make sure you are in the required working directory. Open the experiment folder. Double click on the scilab file you want to execute. Doing so will automatically launch scilab and also automatically change the working directory. To know your working directory at any time, execute the command `pwd` in the scilab console.
2. Next, we have to load the content of `common-files` directory. Notice that this directory is just outside the `Step_test` directory. The `common-files` directory has several functions written in `.sci` files. These functions are required for executing any experiment. To load these functions type `getd<space>folder path`. The `folder path` argument will be the complete path to `common-files` directory. Since this directory is just outside our `Step_test` directory, the command can be modified to `getd<space>..\common_files` So now we have all functions loaded.
3. Next we have to load the serial communication toolbox. For doing so we have to execute the `loader.sce` file present in the `common-files` directory. To do so execute the command `exec<space>..\common_files\loader.sce` or `exec<space>folder path\loader.sce`.

4. Next, click on `editor` from the menu bar to open the Scilab editor or simply type `editor` on the Scilab console and open the file `ser_init.sce`. Change the value of the variable `port2` to the COM number identified for the connected SBHS. For example, one may enter `'COM5'` as the value for `port2`. Notice that there is no space between COM and 5 and COM5 is in single quotes. Keep all other parameters untouched. Execute this `.sce` file by clicking on the execute button available on the menu bar of scilab editor window. The message `COM Port Opened` is displayed on successful implementation. If there are any errors, reconnecting the USB cable and/or restarting Scilab may help.
5. Next we have to load the function for the step test experiment. This function is written in `step_test.sci` file. Since we do not have to make any changes in this file we can directly execute it from scilab console without opening it. Run the command `exec<space>step_test.sci` in scilab console. The results are illustrated in figure 3.3.
6. Next, type `Xcos` on the Scilab console or click on `Applications` and select `Xcos` to open Xcos environment. Load the `step_test.xcos` file from the `File` menu. The Xcos interface is shown in figure 3.5. The block parameters can be set by double clicking on the block as shown in figure 3.6. To run the code click on `Simulation` menu and click on `Start`. After executing the code in Xcos successfully the plots as shown in figure 3.7 will be generated. Note that the values of fan and heater given as input to the Xcos file are reflected on the board display.
7. To stop the experiment click on the `Stop` option on the menu bar of the Xcos environment.

All of the activities mentioned above, from `getd<space>..\common_files` untill starting the xcos simulation, are coded in a file named `start.sce`. Executing this file will do all necessary things automatically with just click of a button. This file however assumes three things. These are

1. The location of `common-files` directory is not changed
2. The current working directory is correct
3. The port number mentioned in `ser_init.sce` is correct



The image shows a screenshot of the Scilab Console window. The window has a title bar 'Scilab Console' and a menu bar with 'File', 'Edit', 'Preferences', 'Control', and 'Applications'. Below the menu bar is a toolbar with various icons. The main area of the console displays the following text:

```
scilab-5.2.2

Consortium Scilab (DIGITEO)
Copyright (c) 1989-2010 (INRIA)
Copyright (c) 1989-2007 (ENPC)

Startup execution:
loading initial environment

-->exec loader.sce

-->mode(-1);

Serial port toolbox loaded
-->mode(-1);

-->mode(-1);

Serial port toolbox loaded

COM Port Opened
-->mode(-1);

-->
```

Figure 3.3: Expected responses seen on the console

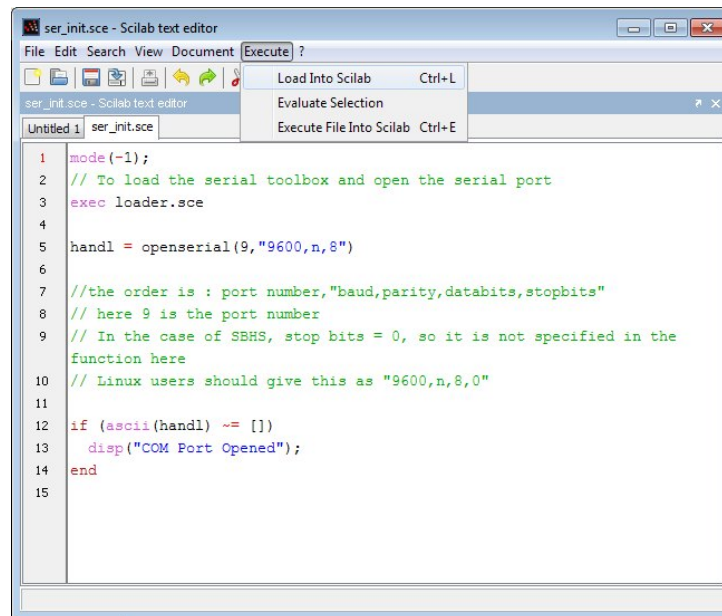


Figure 3.4: Executing script files

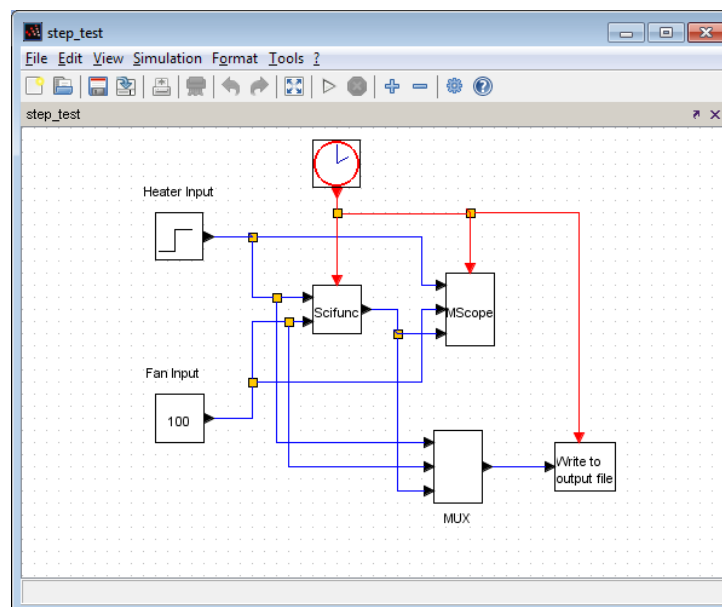


Figure 3.5: Xcos Interface

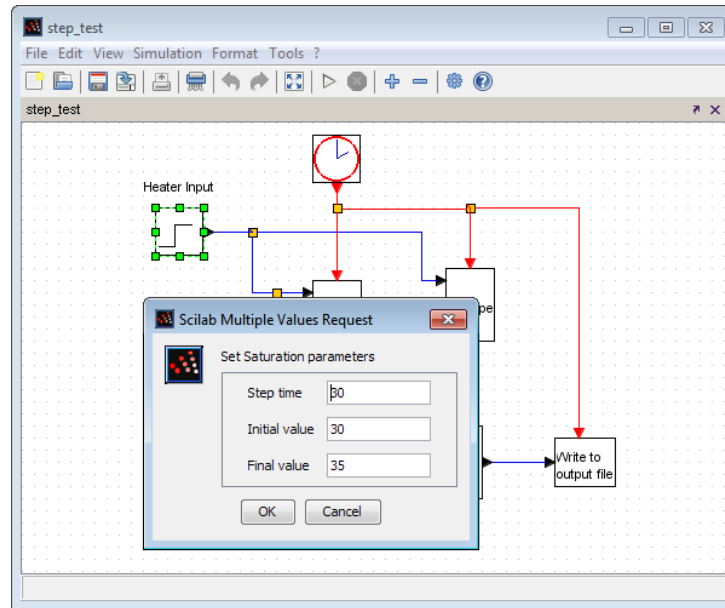


Figure 3.6: Setting Block Parameters

3.2 Accessing Single Board Heater System on a Linux System

This section deals with the procedure to use SBHS on a Linux Operating System. The Operating System used for this document is Ubuntu 10.04. For Linux users, the instructions given in section 3.1 hold true with a few changes as below:

FTDI COM port drivers are not required for connecting the SBHS to the PC. After plugging in the USB cable to the PC, check the serial port number by typing `ls /dev/ttyUSB*` on the terminal, refer Fig.3.8. Usually, the highest numbered one will be the required device port number. eg:- `/dev/ttyUSB0`. If you want to connect more than one USB devices, then type `tail -f /var/log/messages | grep ttyUSB` on the linux terminal just before plugging in the individual USB cable, refer Fig.3.9. The USB number will be shown on the screen. Press `Ctrl+c` to abort the command.

Note down this number and change the port number (the first argument of the `openserial()` function) in the `ser_init.sce` file to the noted port number. The second argument of the `openserial()` function should be `"9600,n,8,0"`, refer Fig.3.10. This defines baud rate, parity, data bits and stop bits in that order. It has

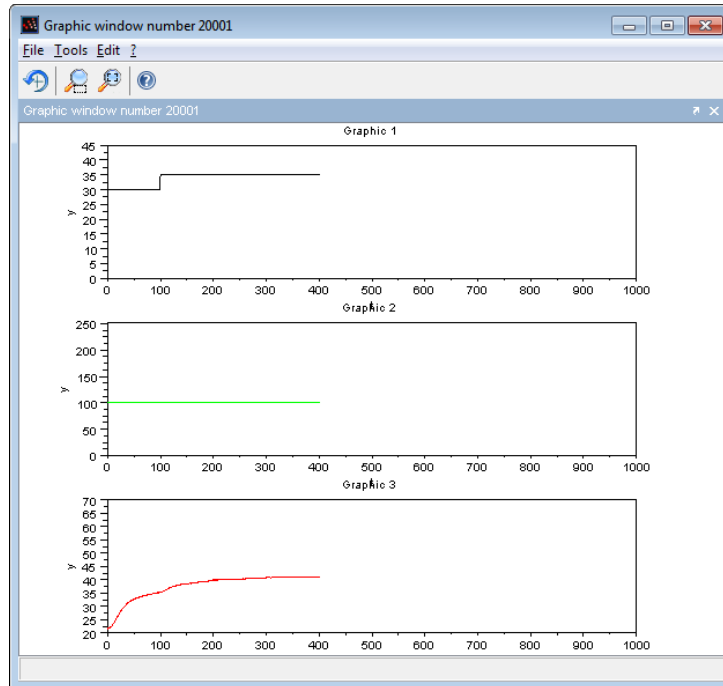
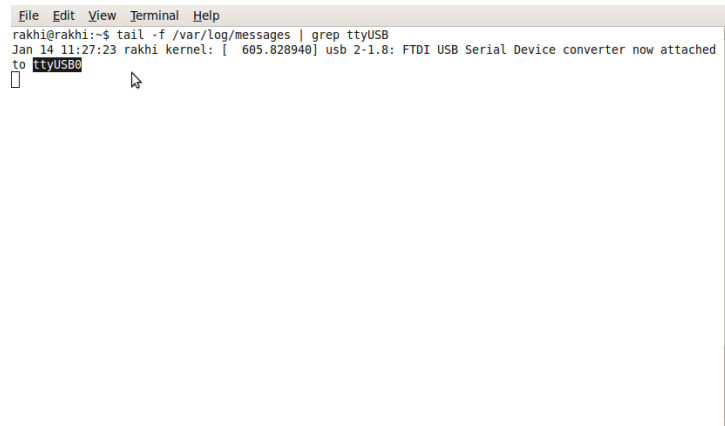


Figure 3.7: Plot obtained after executing step_test.xcos

```
File Edit View Terminal Help
rakhi@rakhi:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
rakhi@rakhi:~$
```

Figure 3.8: Checking the port number in linux (1)

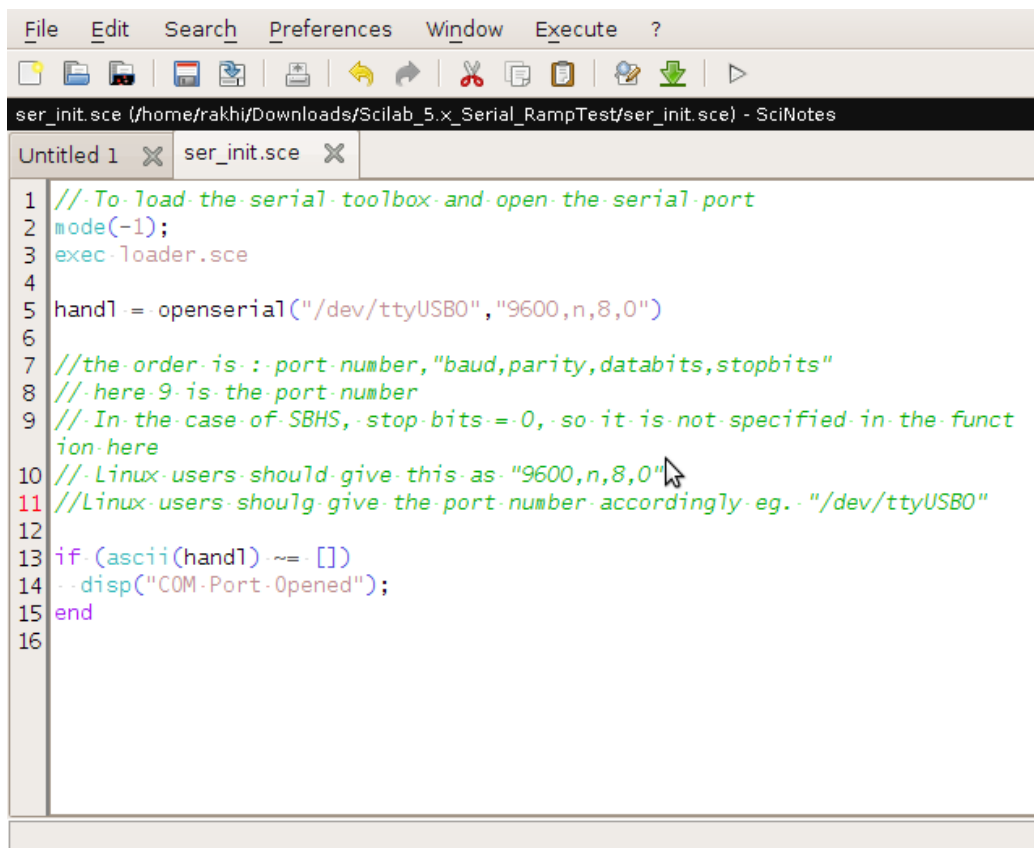


```
File Edit View Terminal Help
rakhi@rakhi:~$ tail -f /var/log/messages | grep ttyUSB
Jan 14 11:27:23 rakhi kernel: [ 605.828940] usb 2-1.8: FTDI USB Serial Device converter now attached
to ttyUSB0
```

Figure 3.9: Checking the port number in linux (2)

been found that if the last parameter i.e., stop bits is omitted instead of specifying it as zero, Scilab produces an error. Execute this file. Once the serial port initialisation is successfully completed, a message is displayed as shown in Fig.3.11. If there are any errors, reconnecting the USB cable and/or restarting Scilab may help.

Now execute the `step_test.sci` file. The results are illustrated in figure 3.3. Type `Xcos` on the Scilab console or click on **Applications** and choose `Xcos` to open `Xcos` environment. Load the `step_test.xcos` file from the **File** menu. The `Xcos` interface will open as shown in figure 3.5. The block parameters can be set by double clicking on the block as shown in figure 3.6. To run the code click on **Simulation** menu and select **Start**. After executing the code in `Xcos` successfully the plots as shown in figure 3.7 will be generated. Note that the values of fan and heater given as input to the `Xcos` file are reflected on the board display. To stop the experiment click on the **Stop** option on the menu bar of the `Xcos` environment.



```
1 //To load the serial toolbox and open the serial port
2 mode(-1);
3 exec loader.sce
4
5 hand1 = openserial("/dev/ttyUSB0","9600,n,8,0")
6
7 //the order is : port number, "baud,parity,databits,stopbits"
8 //here 9 is the port number
9 //In the case of SBHS, stop bits = 0, so it is not specified in the function here
10 //Linux users should give this as "9600,n,8,0"
11 //Linux users should give the port number accordingly eg. "/dev/ttyUSB0"
12
13 if (ascii(hand1) ~= [])
14     disp("COM Port Opened");
15 end
16
```

Figure 3.10: Configuring port number and other parameters

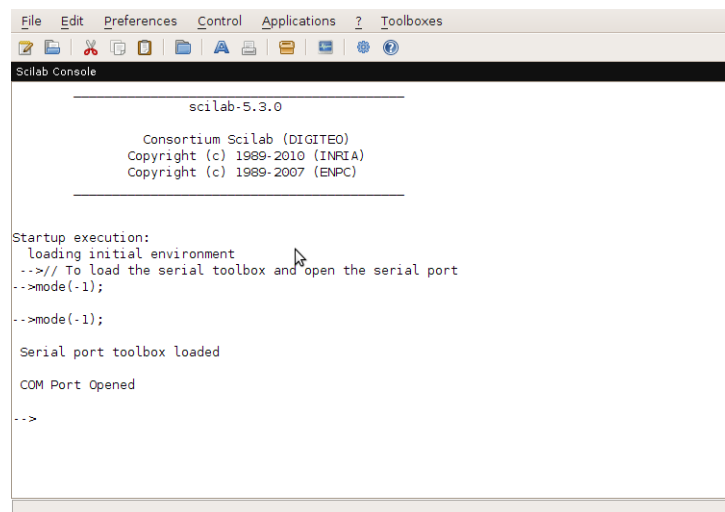


Figure 3.11: Scilab Console Message after Opening Serial Port

Bibliography

- [1] Fossee moodle. <http://www.fossee.in/moodle/>. Seen on 10 May 2011.
- [2] Spoken tutorials. http://spoken-tutorial.org/Study_Plans_Scilab. Seen on 10 May 2011.
- [3] K. M. Moudgalya. Introducing National Mission on Education through ICT. <http://www.spoken-tutorial.org/NMEICT-Intro>, 2010.
- [4] K. M. Moudgalya and Inderpreet Arora. A Virtual Laboratory for Distance Education. In *Proceedings of 2nd Int. conf. on Technology for Education, T4E*, IIT Bombay, India, 1–3 July 2010. IEEE.
- [5] Kannan M. Moudgalya. *Digital Control*. John Wiley and Sons, 2009.
- [6] Kannan M. Moudgalya. *Identification of transfer function of a single board heater system through step response experiments*. 2009.
- [7] Katsuhiko Ogata. *Modern Control Engineering*. Prentice-Hall of India, 2005.
- [8] Dale E. Seborg, Thomas F. Edgar, and Duncan A. Mellichamp. *Process Dynamics and Control*. John Wiley and Sons, 2nd edition, 2004.
- [9] Virtual labs project. Single board heater system. http://www.co-learn.in/web_sbhs. Seen on 11 May 2011.