Internship Report

Akhil Guliani

17 Jan, 2011

Acknowledgement

I thank the Global Internship Program in Engineering Design and Innovation of the Airtel IIT Delhi Centre of Excellence in Telecommunication (GIPEDI.AICET), Bharti School of Telecommunication Technology and Management, Indian Institute of Technology Delhi for giving me the opportunity to participate in the Program.

I thank Subrat Kar for being my Faculty Mentor and for giving me and all the interns his most valuable insights on life and his experince in the field of teaching. I also thank Akshat Bisht and Vijay Rao who, as Student Mentors, provided valuable and timely help.

Akhil Guliani
Date: 15 Jan, 2011

Certificate of Internship

This is to certify that Akhil Gulinai of Netaji Subhas Institue of Technology has completed this work and submitted this report in partial fulfillment of the requirements of the Global Internship Program in Engineering Design and Innovation of the Airtel IIT Delhi Centre of Excellence in Telecommunication (GIPEDI.AICET), Bharti School of Telecommunication Technology and Management, Indian Institute of Technology Delhi.

Prof. Subrat Kar
Program Coordinator, GIPEDI.AICET
Global Internship Program in Engineering Design and Innovation
Airtel IIT Delhi Centre of Excellence in Telecommunication
Bharti School of Telecommunication Technology and Management
Indian Institute of Technology Delhi
Hauz Khas, New Delhi 110016, INDIA
bharti-coord@admin.iitd.ac.in
+91.11.2659 6200

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The following report is the record for my work done under the Global Internship Program in Engineering Design and Innovation of the Airtel IIT Delhi Centre of Excellence in Telecommunication (GIPEDI.AICET), Bharti School of Telecommunication Technology and Management, Indian Institute of Technology Delhi.

I enrolled for the batch-4 ;December 15, 2010 to January 15, 2011; of the internship program. During the program I got acquainted with various free and open source softwares for hardware and software design such as gEDA, GCC, LaTeX etc. I also got to work on a free and opensource Operating system "Ubuntu 10.04 LTS"[3].

1.1 Problem statement and proposed methodology

The problem statement given to me was to understand and design a device programmer for the TI SoC CC2530.

The proposed methodology was to first read up on Embedded systems , microcontrollers and SoC's in regard with programming these devices.

Then follow it up by assembling the USBasp (designed by Vijay Rao) and using it program an Atmega 8 microcontroller.

This would be followed by assembling the CC-Flasher hardware on a general purpose PCB.

Then the assembled hardware was to be programmed with the modified code and to be tested with the CC2530.

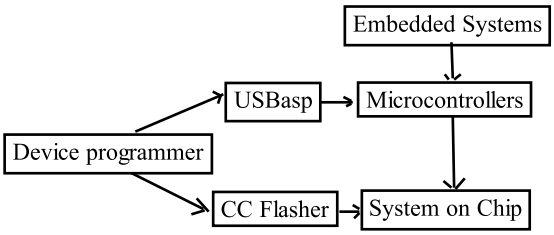

Figure 1.1: Schematic (block) diagram of entire project

1.2 Organisation of report

The report is organised in the following manner

Chapter 2 is about the assignments given during the first week of the internship

Chapter 3 is about the CC2530

Chapter 4 is about the CC-Flasher

Chapter 5 deals with the methodology and results of the project

Chapter 2

Assignments

2.1 LaTeX and gEDA

2.1.1 What is TeX ?

TeX is a markup language created by Donald Knuth to typeset documents attractively and consistently. The fine control TeX offers makes it very powerful, but also difficult and time-consuming to use.

2.1.2 What is LaTeX ?

LaTeX [19]is a macro package based on TeX created by Leslie Lamport. Its purpose is to simplify TeX typesetting. Many lauthors have contributed extensions, called packages or styles, to LaTeX. Some of these are bundled with most TeX/LaTeX software distributions; more can be found in the Comprehensive TeX Archive Network (CTAN).

Since LaTeX comprises of a group of TeX commands, LaTeX document processing is essentially programming. You create a text file in LaTeX markup. The LaTeX macro reads this to produce the final document.

Clearly this has some disadvantages in comparison with a WYSIWYG (What You See Is What You Get) program such as Openoffice.org Writer or Microsoft Office Word:

- the final result can't be seen straight away.
- The need to know the necessary commands for LaTeX markup.
- It can sometimes be difficult to obtain a certain typeset.

On the other hand, there are certain advantages to the markup language approach:

- The layout, fonts, tables and so on are consistent throughout.
- Mathematical formulae can be easily typeset.
- Indices, footnotes and references are generated easily.
- The documents will be correctly structured.

The LaTeX-like approach can be called WYSIWYM, i.e. What You See Is What You Mean: you can't see how the final version will look like while typing. Instead you see the logical structure of the document. LaTeX takes care of the formatting for you. The LaTeX document is a plain text file containing the content of the document, with additional markup. When the source file is processed by the macro package, it can produce documents in several formats. LaTeX supports natively DVI and PDF, but using other software you can easily create PostScript, PNG, JPG, RTF, etc.

2.1.3 What is gEDA ?

The term gEDA[18] refers to two things:

1. A set of software applications (CAD tools) used for electronic design released under the GPL. As such, gEDA is an ECAD (electronic CAD) or EDA (electronic design automation) application suite. gEDA is mostly oriented towards printed circuit board design (as opposed to integrated circuit design). The gEDA applications are often referred to collectively as "the gEDA Suite".
2. The collaborative of free software/open-source developers who work to develop and maintain the gEDA toolkit. The developers communicate via gEDA mailing lists, and have participated in the annual "Google Summer of Code" event as a single project. This collaborative is often referred to as "the gEDA Project".

The word "gEDA" is a conjunction of "GPL" and "EDA". The names of some of the individual tools in the gEDA [18]Suite are prefixed with the letter "g" to emphasize that they are released under the GNU General Public License. Within the gEDA Suite, gEDA/gaf ("gaf" stands for "gschem and friends") is the smaller subset of tools grouped together under the gEDA name and maintained directly by the gEDA project's founders. GEDA/gaf includes:

- gschem - A schematic capture program

- gnetlist - A netlist generation program
- gsymcheck - A syntax checker for schematic symbols
- gattrib - A spreadsheet program for editing symbol attributes on a schematic.
- libgeda - Libraries for gschem, gnetlist, gsymcheck
- gsch2pcb - Forward annotation from schematic to layout using pcb

2.1.4 Problem Statement

To design the schematics and the PCB for the interface circuit for a proximity sensor to an A\D converter. Document this using LaTeX.

2.1.5 Results

This assignment made me comfortable with the use of markup language based document processors and free EDA packages for design and documentation of hardware.

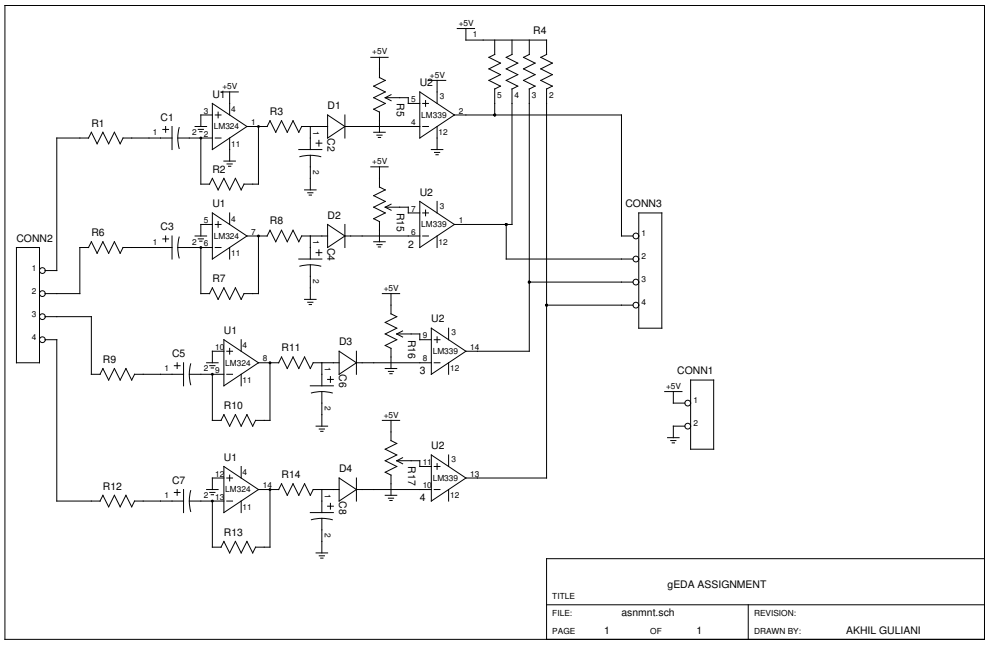

Figure 2.1: The Schematics for Assignment1

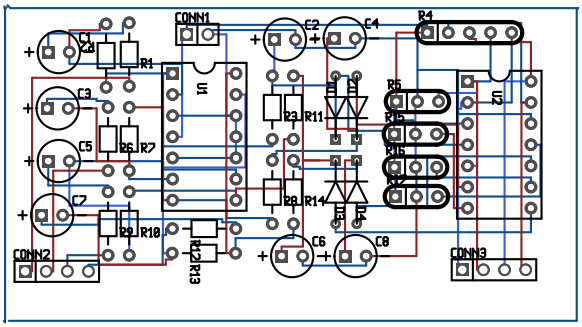

Figure 2.2: The PCB for Assignment 1

2.2 GNU C Compiler and Doxygen

2.2.1 What is GCC?

The Linux operating system provides an outstanding set of development tools for all programming languages. One such tool is a compiler for the C programming language called the GNU C Compiler, popularaly known as gcc[4, 12].

The GCC is a command line compiler and uses the following syntax in the terminal:

```
$ gcc -o <out_filename> <input_filename>.c
```

The above syntax generates binary output file which can be run using the following syntax in the terminal window:

```
$ ./<out_filename>
```

2.2.2 GNU make program

In a large program with many source and header files, the files typically depend on one another in complex ways. When a file is changed that other files depend on , all the files have to be recompiled. For large programs it becomes difficult and time consuming to determine which modules need to be recompiled. The make utilty helps in automating this process.

The make program looks at dependency lines in a file name called Makefile in the working directory. The dependecy lines indicate relationships among files, specifying a target file that depends on one or more prerquiste files. If any of the prerequiste files have been modified more recently than the target file then the make program updates the target based on the construction commands that follow the dependency line[12].

The followig is the structure of a make file:

```
target-file: prerequisite-list
        construction-commands
```

2.2.3 Doxygen

Doxygen is a program which helps in documenting the program code. It supports the following languages by default: C, C++, C#, Objective-C, IDL, Java, VHDL, PHP, Python, Fortran, and D. Doxygen reads the code and taking help from comments given by the user generates any or all of the following outputs as specified by the user[14, **?**].

- HTML output The generated HTML documentation can be viewed by pointing a HTML browser to the index.html file in the html directory.
- LaTeX output The generated documentation must first be compiled by a TeX compiler. To simplify the process of compiling the generated documentation, doxygen writes a Makefile into the latex directory.
- RTF output Doxygen combines the RTF output to a single file called refman.rtf. This file is optimized for importing into the Microsoft Word.
- XML output The XML output consists of a structured "dump" of the information gathered by doxygen. Each compound (class/namespace/-file/...) has its own XML file and there is also an index file called index.xml.
- Man page output The generated man pages can be viewed using the man program(for more information please refer[12]).

The comments given by the user are required to be wriiten in comment blocks as follows

```
/**
* text
*/
```

or like this

```
/*!
* text
*/
```

or like this

```
//////////////////////////////////
/// text
//////////////////////////////////
```

The text part may also include some special commands that are used to indicate what the comment block contains documentation for. Some of these special structural commands are[14]:

- \struct to document a C-struct.
- \union to document a union.
- \enum to document an enumeration type.
- \fn to document a function.
- \var to document a variable or typedef or enum value.
- \def to document a #define.
- \typedef to document a type definition.
- \file to document a file.
- \namespace to document a namespace.

The above commands are followed by the name of their respective objects followed by a brief and long description.

2.2.4 Problem statement

1. Write a C program that accepts 'n' user-names one-by-one and prints them in a sorted order. A linked list has to be used to maintain the set of names to keep the value of 'n' variable. Each node in the list stores a user-name. When a name is entered it is inserted at its 'proper place' in the list. After all names are entered, the list is printed.
2. The linked list operations (addition, insertion, deletion) should be implemented in a separate file (userlist.c). A makefile should be used to build the the executable 'user-sorter'.
3. The program should be commented using doxygen style comments and doxygen should be used to generate reports in html and latex (which has to be compiled into a pdf). Mention your name, description of each function and its parameters at appropriate places in the program with doxygen-style tags so that this information automatically appears in the generated reports.

2.2.5 Results

An efficient way of code documentation using comments and markers was learned. Command line compiling and makefile usage were also used and successfully implemented.

Chapter 3

The TI SoC CC2530

3.1 Description

The CC2530 is a true system-on-chip (SoC) solution for IEEE 802.15.4, Zigbee and RF4CE applications. It enables robust network nodes to be built with very low total bill-of-material costs. The CC2530 combines the excellent performance of a leading RF transceiver with an industry-standard enhanced 8051 MCU, in-system programmable flash memory, 8-KB RAM, and many other powerful features. The CC2530 comes in four different flash versions: CC2530F32/64/128/256, with 32/64/128/256 KB of flash memory, respectively. The CC2530 has various operating modes, making it highly suited for systems where ultralow power consumption is required. Short transition times between operating modes further ensure low energy consumption. [6]

3.2 Features[6]

1. High-Performance and Low-Power 8051 Microcontroller Core With Code Prefetch
2. 32-, 64-, 128-, or 256-KB In-System-Programmable Flash
3. 8-KB RAM With Retention in All Power Modes
4. Hardware Debug Support
5. 2.4-GHz IEEE 802.15.4 Compliant RF Transceiver
6. Excellent Receiver Sensitivity and Robustness to Interference
7. Programmable Output Power Up to 4.5 dBm

8. In Compliance With Worldwide Radio-Frequency Regulations: ETSI EN 300 328 and EN 300 440 (Europe), FCC CFR47 Part 15 (US) and ARIB STD-T-66 (Japan)

3.3 Applications[6]

1. 2.4-GHz IEEE 802.15.4 Systems
2. RF4CE Remote Control Systems (64-KB Flash and Higher)
3. ZigBee Systems (256-KB Flash)
4. Home/Building Automation
5. Lighting Systems
6. Industrial Control and Monitoring
7. Low-Power Wireless Sensor Networks
8. Consumer Electronics
9. Health Care

3.4 Flash Programming for the TI SoC CC2530[6]

3.4.1 Introduction

The CC2530 contains upto 256k of flash memory for storage of program code. The flash memory is programmable by the user via on board software or via the debug interface. The flash controller handles writing and erasing the embedded flash memory. The embedded flash memory consists of up to 128 pages of 2048 bytes (CC2530/CC2531/CC2540) or 1024 bytes (CC2533) each. The flash controller has the following features:

- 32-bit word programmable
- Page erase
- Lock bits for write protection and code security
- Flash-page erase timing 20 ms
- Flash-chip erase timing 20 ms
- Flash-write timing (4 bytes) 20 ms [10]

3.4.2 The CC2530 Debug interface

3.4.2.1 Introduction

The two-wire debug interface allows programming of the on-chip flash, and it provides access to memory and register contents and debug features such as breakpoints, single-stepping, and register modification.

Debug mode is entered by forcing two falling-edge transitions on pin P2.2 (debug clock) while the RESET_N input is held low. When RESET_N is set high, the device is in debug mode. On entering debug mode, the CPU is in the halted state with the program counter reset to address 0x0000. Now the pin P2.1 acts as the debug-data bidirectional pin.

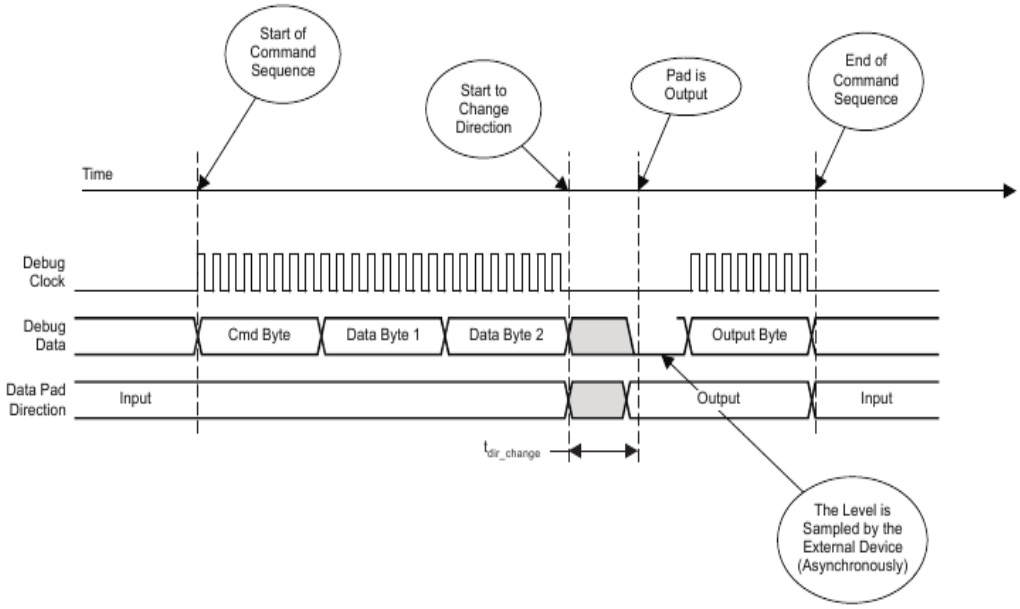

Figure 3.1: Debug timing and command structure

3.4.2.2 Writing flash using Debug mode

Programming of the on-chip flash is performed via the debug interface. The external host must initially send instructions using the DEBUG_INSTR debug command to perform the flash programming with the flash controller.

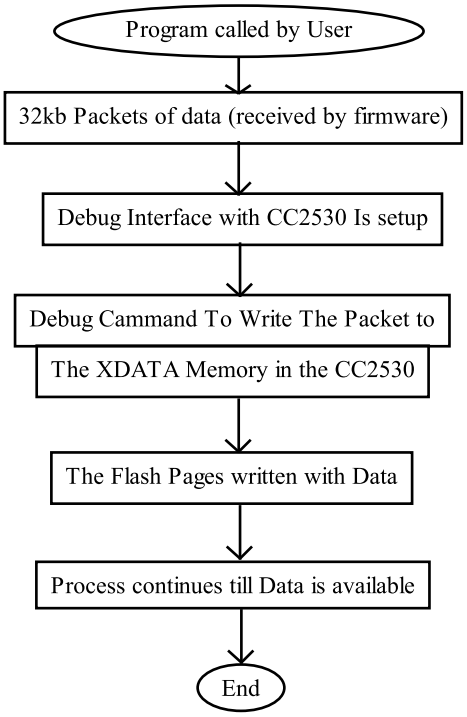

Figure 3.2: Block Diagram for Flash Writing

3.4.3 Changes from CC2430[8]

On the CC2530, it is no longer necessary to set a FWT (flash write timing) register as hardware automatically adjusts its timing depending on the clock source. On CC2530 one has to start the writing process and then write data to the register (the other way around was possible on CC2430).

The information page is not writable from the debug interface on the CC2530. The information page contains lock information for information page and configuration/calibration data from production test. This data includes an IEEE address that can be used by MAC software. Please see [6] for details. The flash lock bits that protect the flash from inadvertently being written in-system are on the CC2530 located on the last flash page. There is one lock bit per 2KB page for improved granularity of locking.

Chapter 4

The CC-Flasher

4.1 The CC-Flasher Software

The software for the CC-Flasher has been written in python, C and C++. The USB driver has been mainly written in python and c. the following figure gives a very basic flow chart outlining the working of CC-Flasher in its default condition.[11]

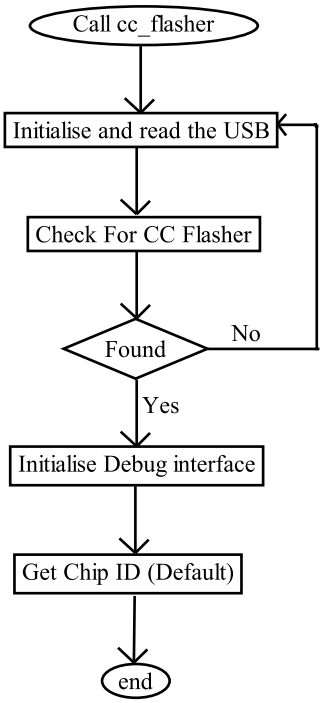

Figure 4.1: CC-Flasher flowchart;

4.1.1 The USB interface

4.1.1.1 USB Basics

USB uses two differential data signals, D+ and D-, which are normally complementary. However, the end of a packet is signalled by pulling both signals low. Data is not transmitted directly on the USB bus, it is NRZI encoded first. This means that a "0" bit is encoded as a bit change, and a "1" bit is encoded as no bit change.The transition occurs on the leading edge of the clock for the given bit.

However, NRZI can have long series of ones, so clock recovery can be difficult unless some form of run length limited (RLL) coding is used. The USB uses bit stuffing, which is efficient, but results in a variable data rate. USB inserts an additional 0 bit after 6 consecutive 1 bits. [15]

4.1.1.2 The USB Driver for the CC-Flasher

The CC-Flasher USB Driver software is interrupt driven. The start of a USB packet triggers an interrupt. The interrupt handler synchronizes with the sync byte, removes the NRZI encoding and bit stuffing, and stores the packed in one of the two RAM buffers. Two buffers are used so that the next packet can be received while the current one is being processed. Depending on the packet type, a reply packet may be sent back immediately in the interrupt handler.[13]

4.1.1.3 The USB Driver on the firmware Side

The firmware calls a usb_poll() function periodically to poll for incoming packets. Once an incoming packet is identified it is processed and the required action is initiated.[13]

4.1.2 The firmware

The CC-Flasher firmware is a modified version of the USBtiny Avr programmer. The CC-Flasher acts as an USB to Debug interface converter.

Standard control requests are directly handled by the USB driver. The firmware is responsible for interacting with the CC2530 and all the required interface code for this is in the dbg_init.h and dbg_init.c files.

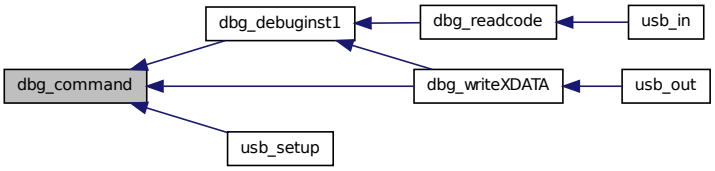

Figure 4.2: The flow of data between the Driver and the Target

4.2 The CC-Flasher Hardware

4.2.1 Hardware Details

Sno.	Element	Value	Quantity
1	Capacitor	22pF	4
2	Capacitor	10uF	1
3	Resistor	22 Ω	4
4	Resistor	10kΩ	1
5	Resistor	1kΩ	1
6	Resistor	1MΩ	1
7	Resistor	56Ω	2
8	Resistor	100Ω	1
9	Voltage Regulator	3.3V	1
10	ATtiny 2313	-	1
11	USB Connector	4 Pin	1
12	Header	10 Pin	1
13	Header	6 Pin	1
14	LED	Red and Yellow	1 each

Table 4.1: Bill of Materials

4.2.1.1 ATtiny 2313[1]

The ATtiny2313 is a low-power CMOS 8-bit microcontroller based on the AVR[16] enhanced RISC[20] architecture. By executing powerful instructions in a single clock cycle, the ATtiny2313 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power con- sumption versus processing speed.

The ATtiny2313 provides the following features: 2K bytes of In-System Programmable Flash, 128 bytes EEPROM, 128 bytes SRAM, 18 general purpose I/O lines, 32 general purpose work- ing registers, a single-wire Interface for On-chip Debugging, two flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, Universal Serial Interface with Start Condition Detector, a programmable Watchdog Timer with internal Oscillator, and three software selectable power saving modes.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, or by a conventional non-volatile memory programmer. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATtiny2313 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications. The ATtiny2313 AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

4.2.1.2 Serial Peripheral Interface or SPI Bus[21]

The Serial Peripheral Interface Bus or SPI bus is a synchronous serial data link standard named by Motorola that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines. Sometimes SPI is called a "four-wire" serial bus, contrasting with three-, two-, and one-wire serial buses.

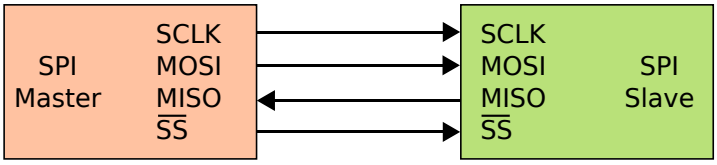

Figure 4.3: SPI bus: single master and single slave[21]

The SPI bus specifies four logic signals:

1. SCLK: Serial Clock (output from master)
2. MOSI; SIMO: Master Output, Slave Input (output from master);
3. MISO; SOMI: Master Input, Slave Output (output from slave);

4. SS: Slave Select (active low, output from master).

During an SPI communication, the master first configures the clock, using a frequency less than or equal to the maximum frequency the slave device supports. The master then pulls the slave select low for the desired chip.

During each SPI clock cycle, a full duplex data transmission occurs[21]:

- The master sends a bit on the MOSI line; the slave reads it from that same line
- The slave sends a bit on the MISO line; the master reads it from that same line

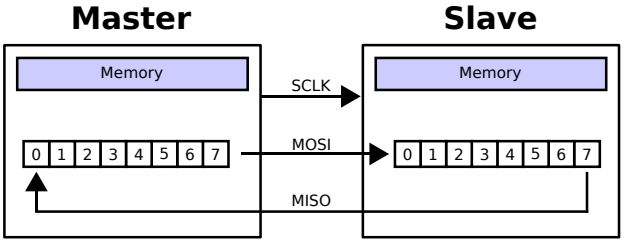

Figure 4.4: Two shift registers to form an inter-chip circular buffer[21]

Transmissions normally involve two shift registers of some given word size, usually eight bits, one in the master and one in the slave; they are connected in a ring. Data is usually shifted out with the most significant bit first, while shifting a new least significant bit into the same register. After that register has been shifted out, the master and slave have exchanged register values. Then each device takes that value and does something with it, such as writing it to memory. If there is more data to exchange, the shift registers are loaded with new data and the process repeats.

Transmissions may involve any number of clock cycles. When there are no more data to be transmitted, the master stops toggling its clock. Normally, it then deselects the slave.

Advantages of SPI Bus[21]

1. Full duplex communication
2. Complete protocol flexibility for the bits transferred
3. Not limited to 8-bit words
4. Arbitrary choice of message size, content, and purpose

5. Extremely simple hardware interfacing
6. Slaves use the master's clock, and don't need precision oscillators
7. Slaves don't need a unique address
8. Transceivers are not needed
9. Uses only four pins on IC packages, and wires in board layouts or connectors, much less than parallel interfaces

Disadvantages of SPI Bus[21]

1. Requires more pins on IC packages than other serial buses such as I^6C
2. No hardware flow control (but master can delay the next clock edge to slow the transfer rate)
3. No hardware slave acknowledgment (the master could be "talking" to nothing and not know it)
4. Supports only one master device
5. No error-checking protocol is defined; hence, generally prone to noise spikes causing faulty communication

4.2.1.3 SPI Interface Connector

Sno.	Pin Type	Pin No.
1	MOSI	7
2	MISO	9
3	SCK	5
4	RESET (SS)	3
5	VCC	1
6	GND	2
7	No Connect (GND)	4,6,8,10

Table 4.2: SPI Pin Configuration

4.2.1.4 USB Connector

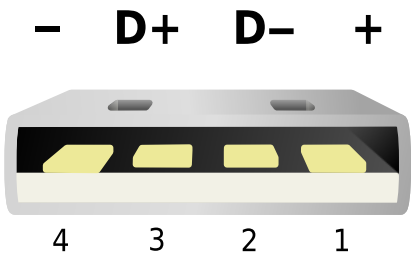

Figure 4.5: USB-A Connector

Pin No.	Type	Wire Colour
1	VCC	Red
2	D-	White
3	D+	Green
4	GND	Black

Table 4.3: USB Connector

4.2.1.5 Debug Interface Connector

Pin No	Function
1	DC
2	DD
3	RST
4	GND
5	VCC

Table 4.4: 5 Pin Debug Interface Connector

Pin No	Function
1	VCC
2	GND
3	DC
4	DD
5	RST
6	NC

Table 4.5: 6 Pin Debug Interface Connector

4.2.2 Schematics

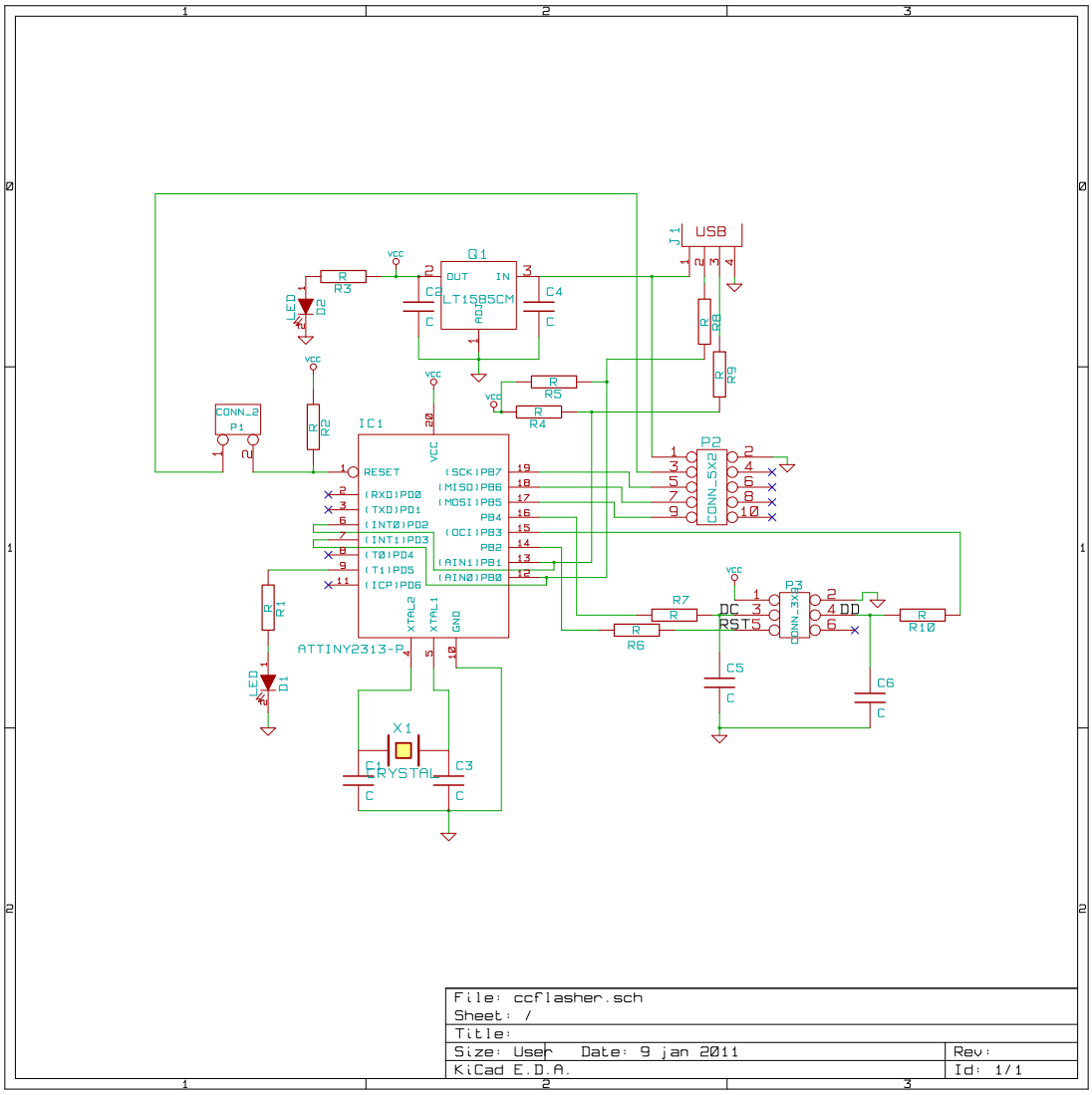

Figure 4.6: The Schematics

4.2.3 The PCB

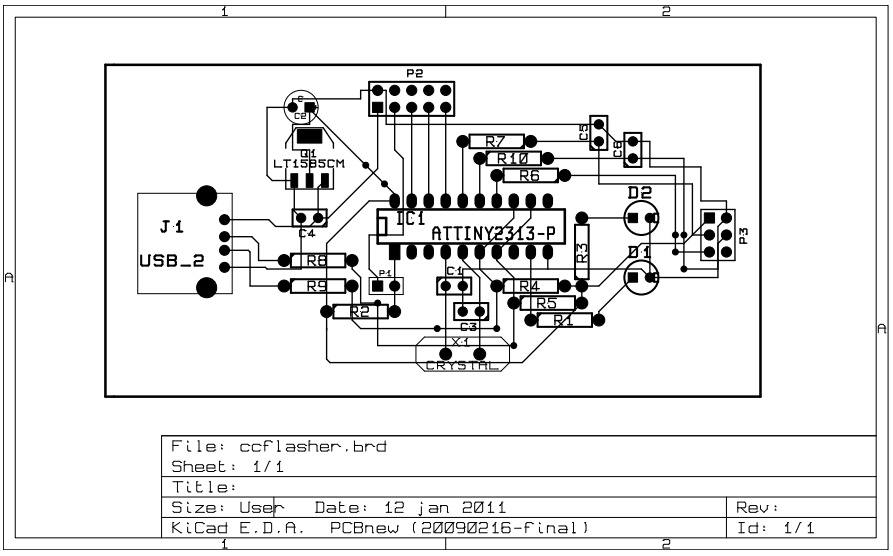

Figure 4.7: The Board

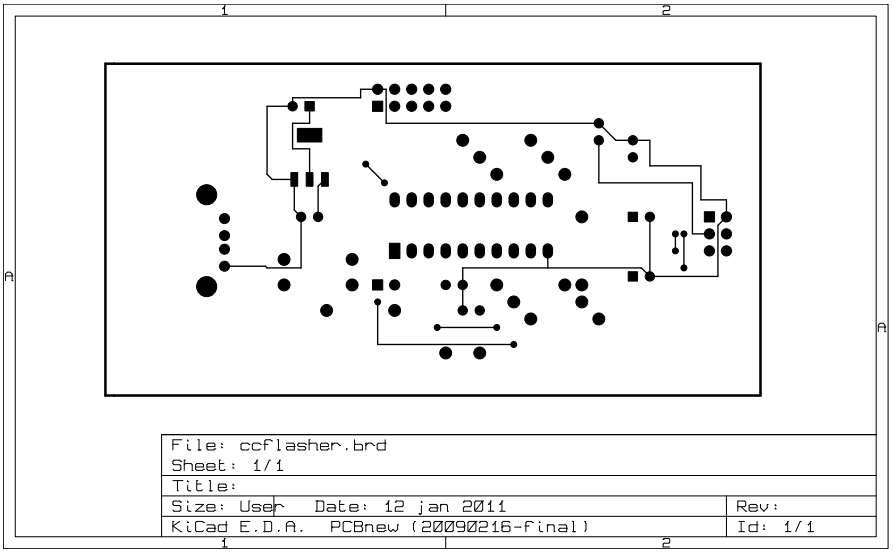

Figure 4.8: Front-Layer

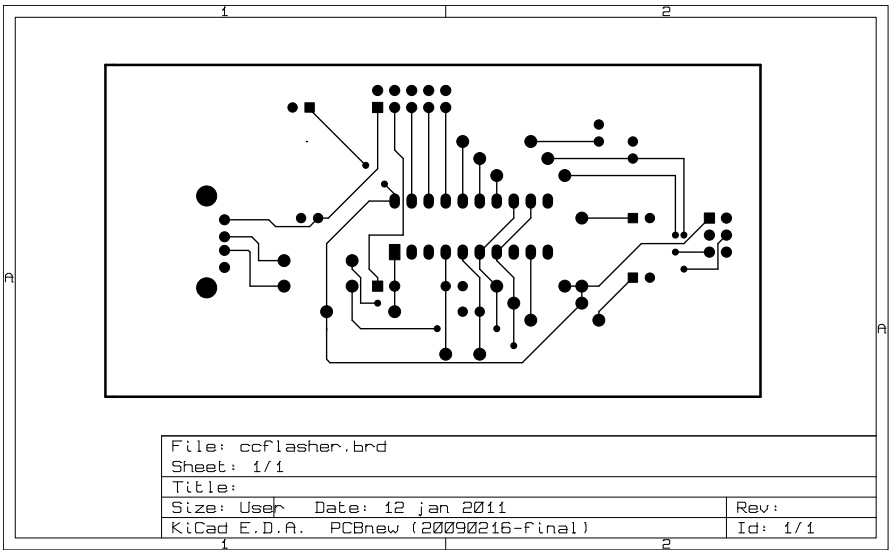

Figure 4.9: Back-Layer

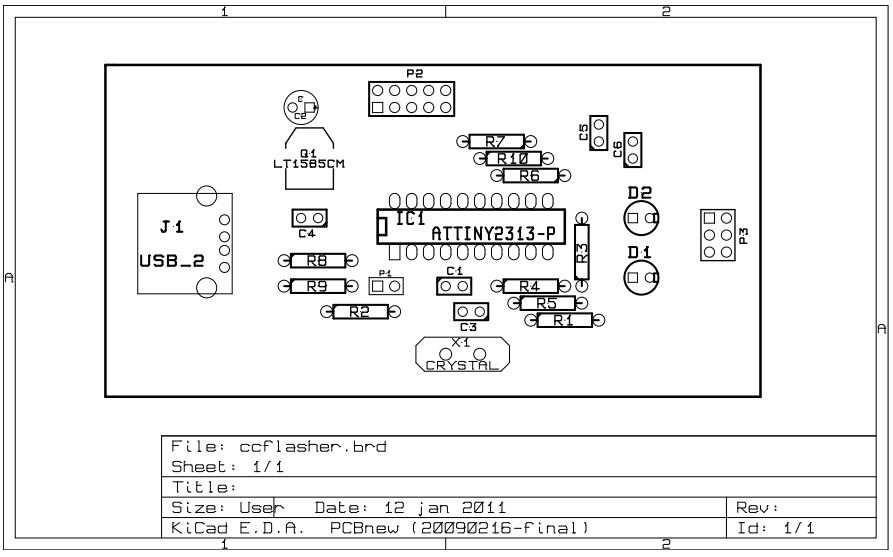

Figure 4.10: Silk Screen

4.2.4 The General PCB

This is a picture of the General PCB assembled and programmed in the Lab during the course of the Internship.

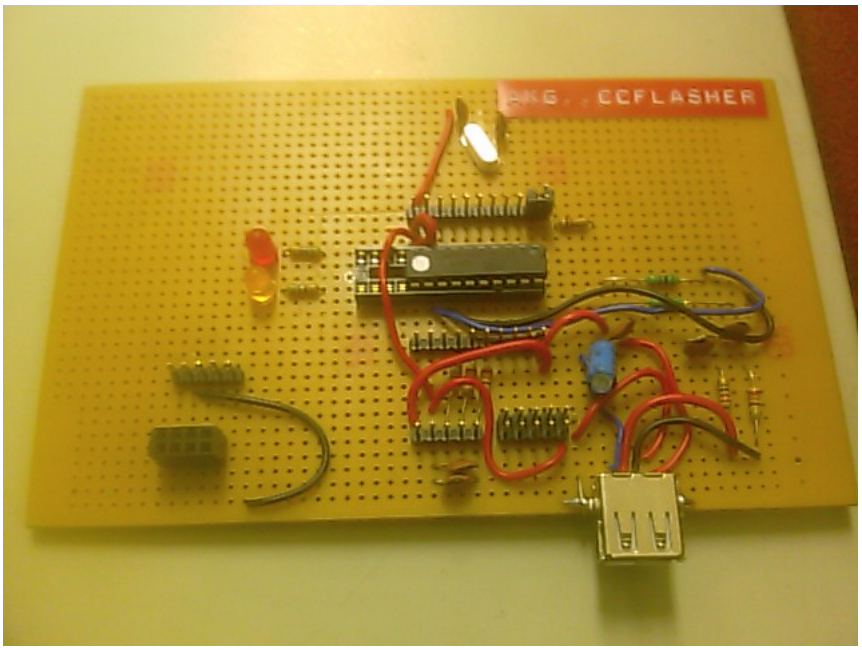

Figure 4.11: The GPCB

Chapter 5

Methodology and Results

5.1 Methodology

1. The PCB layout and schematics have been made in KiCad a free EDA IDE for Ubuntu.

2. The CC-Flasher software is used as it is with minor modifications to the files:

- usbdebuginterface.cpp(CC-Flasher Source)
- usbdebuginterface.h(CC-Flasher Source)
- dbg_init.c (CC-Flasher Firmware)
- dbg_init.h (CC-Flasher Firmware)

1. The changes in the software are in accordance to the following documents provided by the TI :1. CC2430 flash programming Guide [7] ; 2. CC2530 data-sheet [6]

2. The firmware has been burned to the hardware using USBasp programmer (PCB designed by Vijay Rao) and avrdude.

5.2 Results

1. 1. The CCFlasher hardware as described on[2] was replicated, as it didn't require any modifications to program CC2530.

2. The CCFlasher firmware was modified to be able to program the CC2530.

3. The CCFlasher was tested and was able to comminucate with the programmer software on the PC.

4. The CCFlasher was not able to detect the CC2530 and hence could not program it.

5.3 Conclusions

1. The concepts of embedded systems, hardware software co-design and device programing were understood.

2. The debug interface for the CC2530 and CC2430 was understood.

3. PCB design and hardware assembly fundamentals were understood.

4. C language programing and compiling for embedded systems were understood.

5. System-On-Chip(SOC) was understood.

6. The changes in the CC-Flasher software were identified.

5.4 Future work

1. The CCFlasher requires more modifications to be able to detect CC2530.

2. A test can be done using a CC2430(which is know to work with the CCFlasher), to check that the hardware made is functioning properly.

Chapter 6

Appendix

Resume

Name	:	Akhil Guliani
Date of Birth	:	15 / 5/ 1990
Gender	:	Male
Address (residential)	:	A-140 Yojana Vihar, Delhi – 110092
Telephone No	:	011 – 22149637 (residential)
		9990684442 (M)
E-mail id	:	akhil.guliani@gmail.com
Educational Qualifications	:	
1. School Education	:	

Class	Name of School	Name of Board	Year of Passing	Total Percentage	Subjects Taken
X (secondary)	Ryan International School, Trans Yamuna	CBSE , Delhi (Central Board for Secondary Education)	2006	92.6 %	Mathematics Science English Sanskrit Social studies
XII (Senior Secondary)			2008	90.6 %	Mathematics English Physics Chemistry Biology

2. University Education :

Currently enrolled in fifth semester of BE (ICE) at Netaji Subhas Institute of Technology, Delhi; under the Delhi University

Exams Given	Results
Semester 1	Passed in first division with 73%
Semester 2	Passed with distinction with 78%
Semester 3	Passed with distinction with 77%
Semester 4	Passed with distinction with 77%
Semester 5	Results awaited

Aggregate percentage so far : 76%

Internships undertaken :

1. Undertook Winter Internship at NSIT Delhi under Dr. Smriti Srivastava From 10/12/2009 to 10/1/2010
2. Under took Casual studentship at IIT , Delhi under Dr IP Singh from 1/6/2010 to 31/7/2010
3. Undertook GIPEDI.AICET at Bharti school, IIT Delhi under Dr. Subrat Kar From 15/12/2010 to 15/1/2011

Conferences and workshops Attended :

1. Robotics workshops held at NSIT by Robokriti in 2008
2. Ethical Hacking by Ankit Fadia Held at NSIT in 2009
3. YUVA meet 2010 on Global Warming held at British Council by TERI
4. Renewable Energy Festival, January 2011 Held At NSIT

Events Organized and Participated:

1. Innovision 2008 ; volunteer
2. Innovision 2009-2010 ; Joint Secretary ,Event management
3. Innovision 2009-2010 ; B-Plan 3rd place
4. Resonance 2009-2010 ; LAN Gaming, Organizer
5. IEEE GOLD Workshop on UNIX in 2010 at NSIT, Organizer
6. Renewable Energy Festival, January 2011 Held At NSIT, as Speaker

Bibliography

[1] Attiny2313 atmel page.

[2] Cc flasher hardware. web.

[3] Ubuntu.

[4] GNU. Gcc website.

[5] Travis Goodspeed. Goodfet.

[6] Texas Instrumens. *CC2530 user guide (SWRU191B)*. TI, September 2010.

[7] Texas Instruments. Cc2430 flash programming guide.

[8] Texas Instruments. Cc2430 to cc2530 migration guide.

[9] Texas Instruments. Cc2430 user guide.

[10] Texas Instruments. *CC2530 Datasheet SWRS081*, April 2009.

[11] Peter Kuhar. Cc flasher, 2007.

[12] Mark G. Sobell. *A Practical Guide to Linux Commands, Editors, And Shell Programming*. Pearson Education, Inc, 2009.

[13] Usbtiny.

[14] Dimitri van Heesch. Doxygen web site. [Online; accessed 26-January-2011].

[15] Wikipedia. Bit stuffing — wikipedia, the free encyclopedia, 2010. [Online; accessed 16-January-2011].

[16] Wikipedia. Atmel avr — wikipedia, the free encyclopedia, 2011. [Online; accessed 26-January-2011].

[17] Wikipedia. Embedded system — wikipedia, the free encyclopedia, 2011. [Online; accessed 15-January-2011].

[18] Wikipedia. Geda — wikipedia, the free encyclopedia, 2011. [Online; accessed 26-January-2011].

[19] Wikipedia. Latex — wikipedia, the free encyclopedia, 2011. [Online; accessed 26-January-2011].

[20] Wikipedia. Reduced instruction set computing — wikipedia, the free encyclopedia, 2011. [Online; accessed 26-January-2011].

[21] Wikipedia. Serial peripheral interface bus — wikipedia, the free encyclopedia, 2011. [Online; accessed 26-January-2011].

[22] Wikipedia. Ubuntu (operating system) — wikipedia, the free encyclopedia, 2011. [Online; accessed 26-January-2011].