Network Protocol Based Embedded Product Development & Automation of Network Switch Simulator

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(Embedded Systems)

By

Bhatt Zinkal D.

(13MECE03)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2015

Network protocol based Embedded product development & Automation of Network Switch Simulator

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of
Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

Bhatt Zinkal D.
(13MECE03)

Under the guidance of

External Project Guides:
Mrs. Khyati Jasani
Executive R & D Engineer,Masibus
Mr. Shashi Ranjan
Engineer, Principal, Broadcom India Research Pvt. Ltd.

Internal Project Guide:
Prof. Amit Degada
Professor (EC Dept.),
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2015

Declaration

This is to certify that

a. The thesis comprises my original work towards the degree of Master of Technology in Embedded Systems at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgment has been made in the text to all other material used.

- Bhatt Zinkal D.

Disclaimer

"The content of this paper does not represent the technology, opinions, beliefs, or positions of Company, its employees, vendors, customers, or associates."

Certificate

This is to certify that the Major Project entitled **"Network protocol based Embedded product development & Automation of Network Switch Simulator"** submitted by **Bhatt Zinkal Daxeshbhai (13MECE03)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems , Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge,haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Prof. Amit Degada
Guide

Dr. N. P. Gajjar
Program Coordinator

Dr. D.K Kothari
Section Head, EC

Dr. P.N.Tekwani
Head of EE Dept.

Dr. K Kotecha
Director, IT

this page is left blank intentionally to accommodate company certificate.

this page is left blank intentionally to accommodate company certificate.

Acknowledgements

I am grateful to my thesis supervisors **Prof. Amit Degada**, Assistant Professor, EC Department, Nirma University,**Mrs. Khyati Jasani**,Executive Engineer, at Masibus Automation And Instrumentation Pvt. Ltd. and **Mr. Shashi Ranjan**,Engineer,Principal at Broadcom India Research Pvt. Ltd. for their constant guidance and motivation.

I would also like to thank to **Dr. P. N. Tekwani** , Head of Electrical Engineering Department, **Dr. D.K Kothari**, Section Head, EC and **Dr. K Kotecha**, Director IT, Nirma University.

I am deeply indebted to **Dr. N .P .Gajjar**, Program Co-ordinator of M.Tech Embedded Systems for allowing me to undertake this thesis work and for their guidelines during the review process.

I also wish to thank **Mr. Vijay Patel**, Executive R & D Engineer, **Mr. Suhant Raval**, R & D Head Of masibus & **Mr. Balakrishanan Raju**, Sr. Staff Engineer and all other team members at Broadcom,Bangalore for their constant help and support. Without their experience and insights, it would have been very difficult to do quality work.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Bhatt Zinkal

13MECE03

Abstract

This Project work is divided in to two phase, in phase one project work is carried out at Masibus Instrumentation & Automation Pvt. Ltd., Gandhinagar. This company is in Design, Development & manufacturing of a large range of Process controller, one of these range is LC5296, it is a perfect process controller Accurate, reliable control of various process applications is provided by Masibus LC5296, this report presents the work carried out at Masibus for designing & development of an advanced temperature controller using latest controller & compact version which is the demanded by customers, so redesigning of this product is scheduled. As the technology is getting updated with time, the redesigning phase comes as a part of Re-development of existing product. The design present in this thesis uses controller of Renesas RF5100FC with some extra features & more compact version than earlier product was decided, by the end of the November Product was almost ready to deliver and supposed undergo some testing environment, so now it was a time to do some research on betterment of product which lead us to use some wireless network protocol in device instead of wired Modbus, also started working on it.

In second phase, work is carried out at Broadcom India Research Pvt. Ltd., Bangalore. At Broadcom my project work is started initially on configuration of one Proprietary network simulator tool of Broadcom for different chipset & for different environment, which is gradually converted in to automation of this entire configuration for Network simulator. The objective was to automate the process of network switch simulator initialization & invocation using script. Using this interactive script user will have the freedom to choose the different configuration options. SDK version will also be changed. Chip & SDK compatibility will also be checked. Based on user's input (with or without stacking option, number of chip, SDK & chip model, Debugging options) network switch simulator windows has to be invoked, for debugging purpose GNU debugger (GDB) was used.

List of Figures

Abbreviation Notation and Nomenclature

SDK ..... Software Development Kit
GNU ..... GNU's Not Unix
GDB ..... GNU's Not Unix De-Bugger
PCB ..... Printed Circuit Board
PDLC ..... Product Development Life Cycle
QoS ..... Quality of Service
SP ..... Set Point
PV ..... Process value
FCE ..... Final Control Element
CO ..... Controlled Output
PID ..... Proportional-Integral-Derivative
TCP ..... Transmission Control Protocol
TCP/IP ..... Transmission Control Protocol/Internet Protocol
SPI ..... Serial Peripheral Interconnect
USART ..... Universal Synchronous/Asynchronous Receiver/Transmitter
UART ..... Universal Asynchronous Receiver/Transmitter
RTC ..... Real Time Clock
PWM ..... Pulse Width Modulation
I2C ..... Inter-Integrated Circuit
WDT ..... Watch Dog Timer
ADC ..... Analog to Digital Converter
RTD ..... Resistance Temperature Detector
RET ..... Retransmission
COM ..... Communication
TH ..... Through Hole
SMD ..... Surface Mount Device
CS+ ..... Cube-Suite+

Chapter 1

Introduction

1.1 Introduction to Problem statement

In Masibus there is a large range of Process controller, one of these range is LC5296, it is a perfect process controller Accurate, reliable control of various process applications is provided by Masibus LC5296, But now it is time to make an advanced temperature controller using latest controller & by reducing its size we can have more compact version which is the demanded by customers, so redesigning of this product is scheduled. As the technology is getting updated with time, the redesigning phase comes as a part of Redevelopment of existing product. This time using new controller of Renesas RF5100FC with some extra features & more compact version than earlier product was decided, by the end of the November Product was almost ready to deliver and supposed undergo some testing environment, so now it was a time to do some research on betterment of product which lead us to use some wireless network protocol in device instead of wired Modbus, also started working on it. At the end of the semester 3 successfully presented a Product Demo during Final viva. Meanwhile I got an opportunity to work with Broadcom in Network & Switching Department, where I can get a chance to elaborate my knowledge of Networking, which is also related to my work what I have supposed to do at Masibus, so I garbed that opportunity. At Broadcom my project work is started initially on configuration

of one Proprietary network simulator tool of Broadcom for different chipset & for different environment, which is gradually converted in to automation of this entire configuration for Network simulator

1.2 Objective

To design & develop the Modbus based temperature process controller using Renesas RF5100FC micro-controller. This full embedded product development kind of work was carried out at Masibus Automation & instrumentation, Gandhinagar. To automate the process of network switch simulator initialization & invocation using script. Using this interactive script user will have the freedom to choose the different configuration options. SDK version will also be changed. Chip & SDK compatibility will also be checked. Based on user's input (with or without stacking option, number of chip, SDK & chip model, Debugging options) network switch simulator windows has to be invoked, for debugging purpose GNU debugger (GDB) is used. This Automation of Network Simulator Configuration Kind of work is carried out at Broadcom Corporation, Bangalore.

1.3 Motivation

Masibus has a large range of Process controller, one of these range is LC5296, it is a perfect process controller Accurate, reliable control of various process applications is provided by Masibus LC5296, But as per clients demand we need to make an advanced temperature controller using latest controller & by reducing its size we can have more compact version which is the demanded, so redesigning of this product is scheduled. At Broadcom AE team has to maintain average resolving time for customers' query,for that certain things which cost more time is identified & new techniques or use of certain tools which can help to reduce the time to resolve the customers query. The use of network switch simulator is also one of suggested option but initialization for a particular chip with particular configuration is very tedious & time consuming task. Some basic configuration option has been manually

done for start up, but to avoid duplication of work & minimize time some task automation script was needed.

1.4 Thesis Organization

As whole project is carried out in two phases, the work done under the first phase is covered in chapter 2 to 5; the rest of the thesis work is carried out in second phase which is covered in chapter number 6 to 9. All the chapters are organized as follows.

Chapter 2 , *Theoretical Overview* , describes the basic of Process controller & concept of Modbus Protocol.

Chapter 3, *Embedded Product Development* , gives theoretical background to Network Protocol based Embedded Product development life cycle & detailing of each stages.

Chapter 4 , *Product Designing* , describes the both Hardware & Software section of Embedded product Designing.

Chapter 5 , *Future-Scope*, describes the future scope of work done at masibus with interconnection of work done at broadcom in second phase.

Chapter 6 , *Networking & Switching* ,describes the introduction to Automation of network simulator configuration work.

Chapter 7, *Network Switch* , describe the functionality & architecture of Network Switch.

Chapter 8, *Automation of network simulator configuration* , describes how waveforms are interpreted when failures occurs while performing cell level electrical checks.

Finally **Chapter 9** , *Conclusion* , concluding .

Chapter 2

Theoretical Overview

This chapter talks about some theoretical aspects of Process, what is process, need of process control and its types & modes. it also discussed about Process Controller and types of process controller.

2.1 Need of Process Control

Effective process control is required to maintain safe operations, quality products and business viability.

Safety The purpose of a Process Control system is safety: personnel, environmental and equipment safety. The safety of plant personnel and the community is the highest priority in any operation. An example of safety in a "common heat exchanger process" is the installation of a pressure relief valve in the supply. Other examples of safety incorporated into process control systems are rupture disks and blow out panels, a pressure switch that does not allow a pump to over pressurize a pipe or a temperature switch that does not allow the fluid flowing through a heat exchanger to overheat. [2]

Quality In addition to safety, process control systems are important to maintaining product quality. For an example in blending and batching operations, control systems maintain the

proper ratio of ingredients to deliver a consistent & good product. They tightly regulate heat to deliver consistent solids in cooking systems. Without this type of control, products would vary and undermine the quality. [2]

Profit When safety and quality concerns are met, process control objectives can be focused on profit. All processes experience variations and product quality demands that we operate within constraints. A batch system may require +- 0.5% tolerance on each ingredient addition to maintain quality. A cook system may require + 0.5 degrees on the exit temperature to maintain quality. Profits will be maximized the closer the process is operated to these constraints.[2]

2.2 Process

A process is broadly defined as an operation that uses resources to transform inputs into outputs. It is the resource that provides the energy into the process for the transformation to occur.

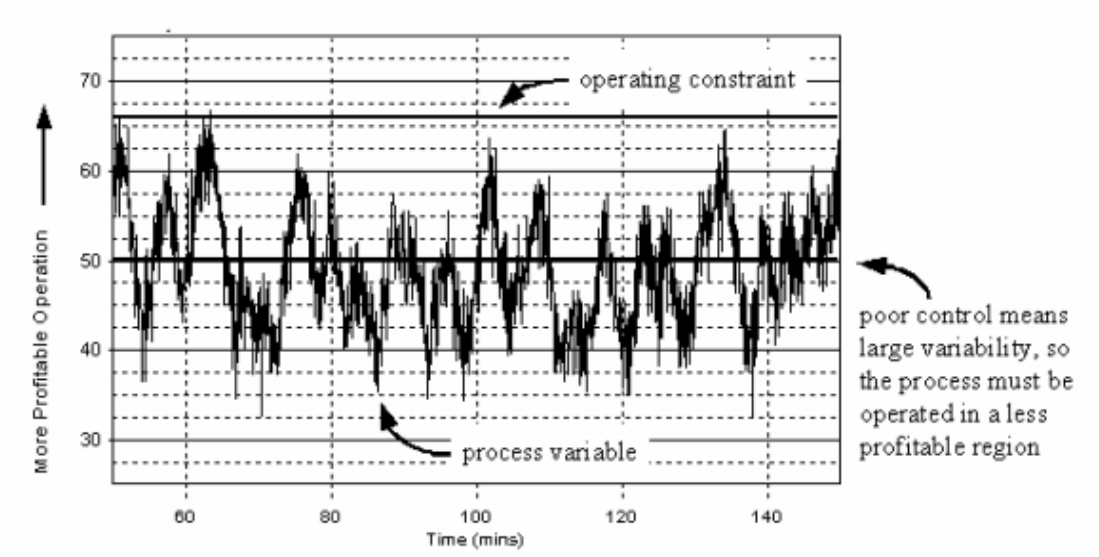

Figure 2.1: Poor Control

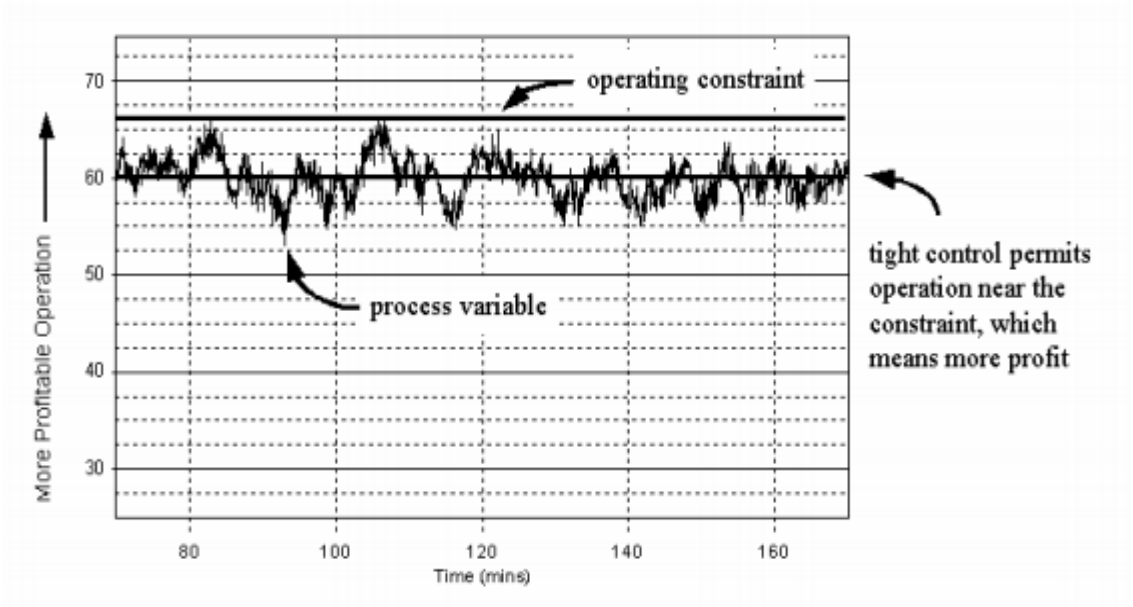

Figure 2.2: Tight Control

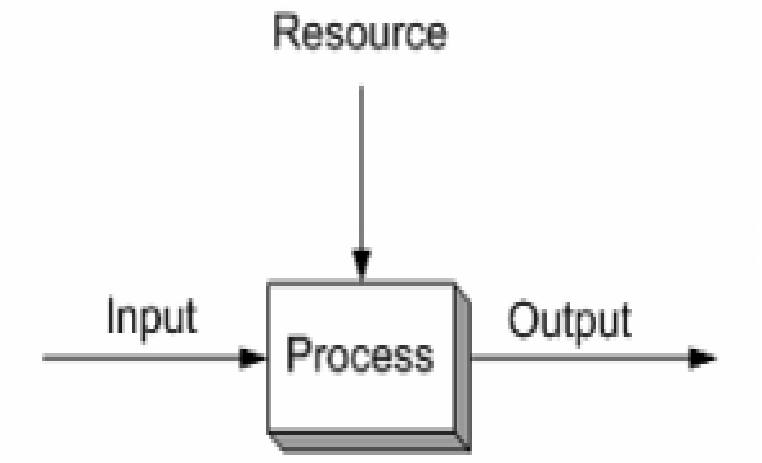

Figure 2.3: Block Representation of Process

2.2.1 Process Control

Process control is the act of controlling a final control element to change the manipulated variable to maintain the process variable at a desired Set Point.

2.3 Basics of Process Control

There are two type of Process Control 1. Open Loop Control 2. Closed Loop Control

- **What is Open Loop Control?** In open loop control the controller output is not a function of the process variable. In open loop control we are not concerned that a particular Set Point be maintained, the controller output is fixed at a value until it is changed by an operator. Many processes are stable in an open loop control mode and will maintain the process variable at a value in the absence of a disturbance. Disturbances are uncontrolled changes in the process inputs or resources. However, all processes experience disturbances and with open loop control this will always result in deviations in the process variable; and there are certain processes that are only stable at a given set of conditions and disturbances will cause these processes to become unstable. But for some processes open loop control is sufficient. Cooking on a stove top is an obvious example. The cooking element is fixed at high, medium or low without regard to the actual temperature of what we are cooking. [2]

- **What is Closed Loop Control?** In closed loop control the controller output is determined by difference between the process variable and the Set Point. Closed loop control is also called feedback or regulatory control. The output of a closed loop controller is a function of the error. Error is the deviation of the process variable from the Set Point and is defined as E = SP - PV. A block diagram of a process under closed loop control is shown in Fig 2.3 [2]

- **What are the Modes of Closed Loop Control?** Closed loop control can be Manual, On-Off, PID, Advanced PID (ratio, cascade, and feed-forward) or Model Based

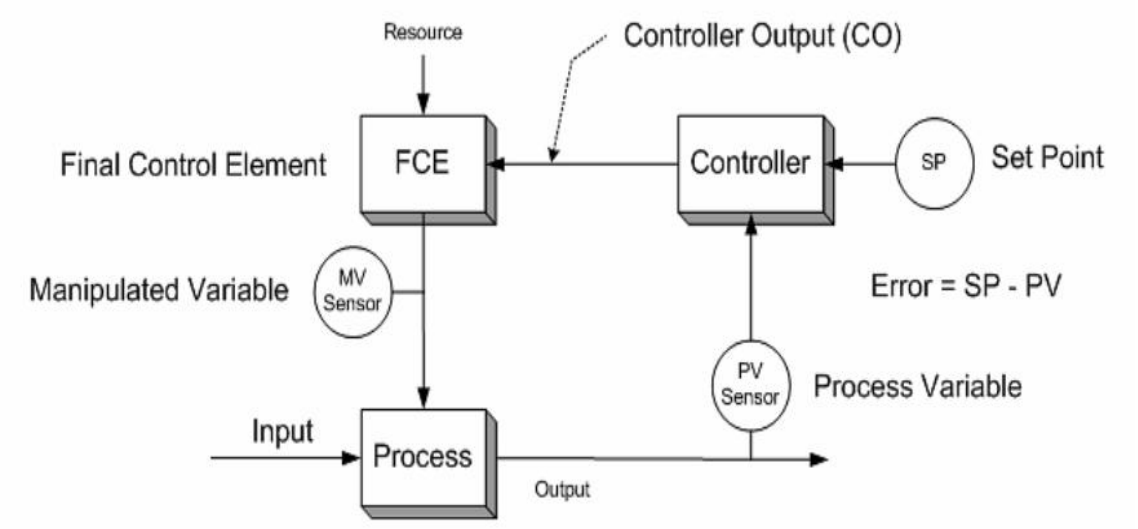

Figure 2.4: Closed loop control

depending on the algorithm that determines the controller output based on the error. [2]

- **Manual Control** In manual control an operator directly manipulates the controller output to the final control element to maintain a Set Point.

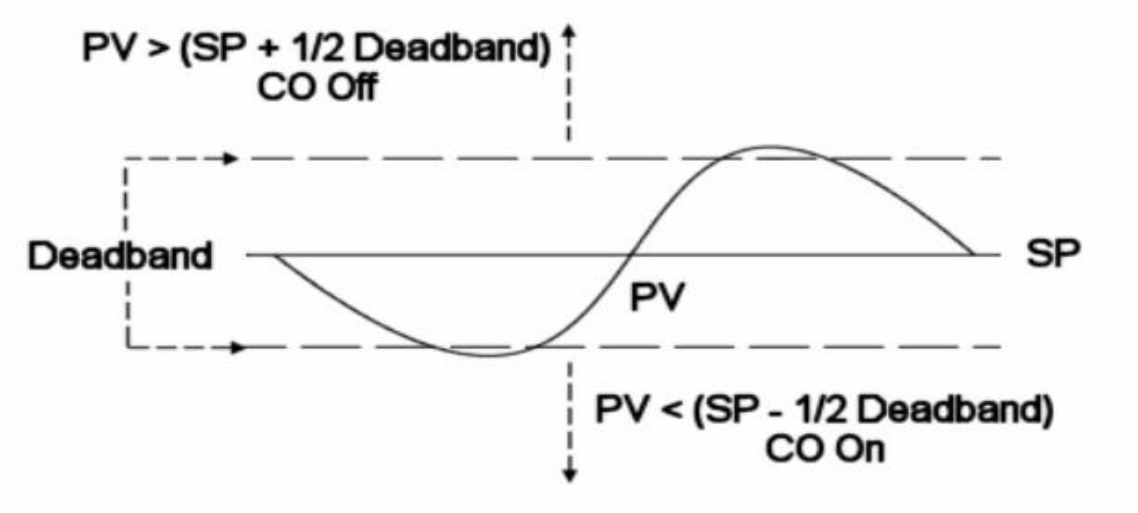

Figure 2.5: Dead band

- **On-Off Control** On-Off control provides a controller output of either on or off in response to error. As an on-off controller only proves a controller output hat is ei-

ther on or off, on-off control requires final control elements that have two command positions: on-off, open-closed. As the controller output can only be either on or off, the steam control valve will be either open or closed depending on the thermostat's control algorithm. For this example we know the thermostat's controller output must be on when the process variable is below the Set Point; and we know the thermostat's controller output must be off when the process variable is above the Set Point. But what about when the process variable is equal to the Set Point? The controller output cannot be both on and off. On-off controllers separate the point at which the controller changes its output by a value called the dead band. Upon changing the direction of the controller output, dead band is the value that must be traversed before the controller output will change its direction again. [2]

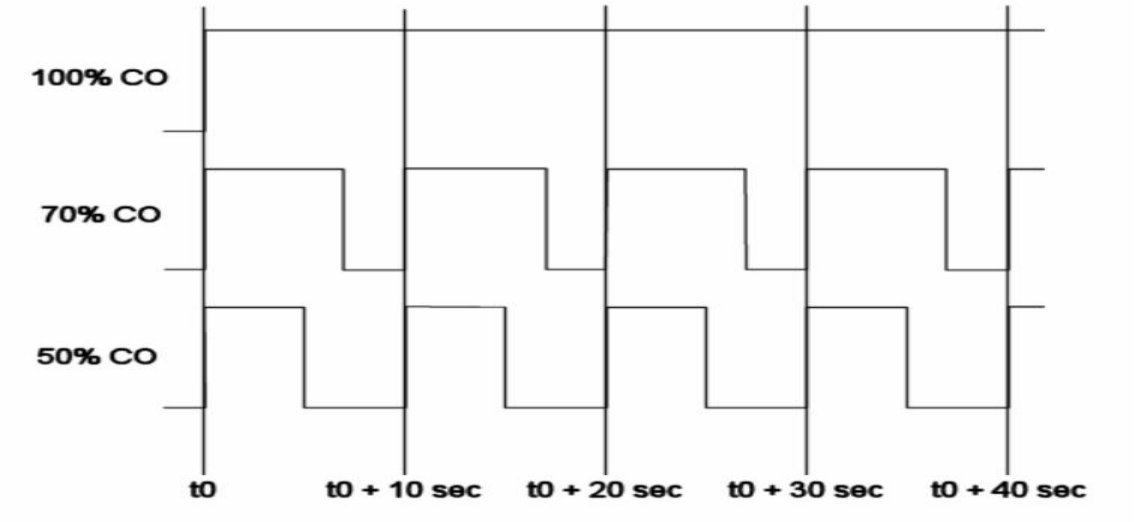

Figure 2.6: Time proportion

- **PID Control** PID control provides a controller output that modulates from 0 to 100 % in response to error.As an on-off controller only proves a controller output that is either on or off, on-off control requires devices that have two command positions: on-off, open-closed. As a PID controller provides a modulating controller output, PID control requires final control elements that have can accept a range of command values, such as valve position or pump speed. To modulate is to vary the amplitude

of a signal or a position between two fixed points. The advantage of PID control over on-off Control is the ability to operate the process with smaller error (no deadband) with less wear and tear on the final control elements. [2]

- **Time Proportion Control** Time proportion control is a variant of PID control that modulates the on-off time of a final control element that only has two command positions. To achieve the effect of PID control the switching frequency of the device is modulated in response to error. This is achieved by introducing the concept of cycle time. Cycle Time is the time base of the signal the final control element will receive from the controller. The PID controller determines the final signal to the controller by multiplying the cycle time by the output of the PID algorithm. [2]

While time proportion control can give you the benefits of PID control with less expensive final control elements it does so at the expense of wear and tear on those final control elements. Where used, output limiting should be configured on the controller to inhibit high frequency switching of the final control element at low controller outputs. [2]

2.4 Basics of Process Control

MODBUS Protocol is a messaging structure developed in 1979, which is used to establish master-slave or client-server communication between two or more intelligent devices. It is a “de facto standard”, truly open protocol and the most widely used network protocol in the industrial manufacturing environment. The MODBUS protocol provides an industry standard method that MODBUS supported devices use for parsing messages.

2.4.1 Communication between MODBUS devices

MODBUS devices communicate using a master-slave technique. In MODBUS the master can initiate transactions which are known as queries. The slaves either respond by supplying the requested data to the master, or it will take the action requested in the query. A slave is any peripheral device which processes information and sends its output to the

master using MODBUS. Masters can either address individual slaves, or it can also initiate a broadcast message to all slaves. Slaves return a response to all queries addressed to them individually, but it will not respond to broadcast queries which are received by all.

2.4.2 REGISTER MAP

MODBUS devices generally include a Register Map. MODBUS functions operate on register map registers which is used to monitor, configure, and control module I/O. one should refer to the register map for their device to gain a better understanding of its operation.

2.4.3 Serial Transmission Modes of MODBUS networks

The transmission mode defines the bit contents of the message bytes transmitted along the network, and also it defines how the message information is to be wrapped into the message stream and decoded. Standard MODBUS networks support any one of two types of transmission modes:
1. ASCII Mode & 2. RTU Mode. The mode of transmission is usually selected along with other serial port communication parameters as part of the device configuration. ASCII Transmission Mode: In the ASCII Transmission Mode (American Standard Code for Information Interchange), 2 ASCII characters are sent for each character byte in a message. This mode allows time intervals between two characters of up to a second during transmission without generating errors. [4] RTU (Remote Terminal Unit) Transmission Mode In RTU (Remote Terminal Unit) Mode, each 8-bit message byte contains two 4-bit hexadecimal characters, and in a continuous stream the message is transmitted. The effective character density increases throughput over ASCII mode at the same baud rate. Mostly RTU mode is used for remote location monitoring & controlling.

2.4.4 MODBUS MESSAGE FRAMING

A message frame is mark at the beginning and ending point of a message allowing the receiving device to determine which device is being addressed and to know when the mes-

sage length. It also allows detection of partial messages and errors flagged as a result. The transmitting device placed MODBUS message in a message frame. Whole message is also placed in a data frame that appends a start bit, stop bit, and parity bit. The word size for ASCII mode is 7 bits, while for RTU mode; the word size is 8 bits. Thus, every 8 bits of an RTU message is effectively considered 11 bits when accounting for the start, stop, and parity bits of the data frame. Do not confuse the message frame with the data frame of a single byte RTU Mode or 7- bit ASCII Mode. The structure of the data frame is depends on the transmission mode weather its ASCII or RTU. Some other network types and on MODBUS Plus, the network protocol handles the framing of messages and uses network specific start and end delimiters. [4]

2.4.5 ASCII Mode Message Frames

ASCII Mode messages start with a colon character ":" (ASCII 3AH) and end with a carriage return-line feed pair of characters. Hexadecimal 0-9 & A-F are the only allowable characters for all other fields. It only takes 7 significant bits to represent an ASCII character. Likewise, the MODBUS ASCII Mode data byte or character is 7 bits long only.
For ASCII Mode of transmission, each character needs 7 data bits. Thus, each character is 10 bits when considering for the start bit, parity bit, and stop bit of the data frame. In ASCII Mode, all network devices continuously monitor the network for the start of message character. When it is received, every network device decodes the next field to determine the addressed device. [4]

2.4.6 RTU Mode Message Frames

RTU mode messages start with a silent interval of at least 3.5 character times implemented as a multiple of character times at the baud rate being used on the network. The first field transmitted is the device address. The allowable characters transmitted for all fields are hexadecimal values 0-9, A-F. A networked device continuously monitors the network, including the silent intervals, and when the first field is received (the address) after a silent

interval of at least 3.5 character times, the device decodes it to determine if it is the addressed device. Following the last character transmitted, a similar silent interval of 3.5 character times marks the end of the message and a new message can begin after this interval. The entire message must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame (not a continuous stream), the receiving device flushes the incomplete message and assumes the next byte will be the address field of a new message.

In similar fashion, if a new message begins earlier than 3.5 character times following a previous message, the receiving device assumes it is a continuation of the previous message. This will generate an error, as the value in the final CRC field will not be valid for the combined messages. [4]

1. MODBUS ADDRESSES The master device addresses a specific slave device by placing the 8-bit slave address in the address field of the message (RTU Mode). The address field of the message frame contains two characters (in ASCII mode), or 8 binary bits (in RTU Mode). Valid addresses are from 1-247. When the slave responds, it places its own address in this field of its response to let the master know which slave is responding.

2. MODBUS FUNCTIONS The function code field of the message frame will contain two characters (in ASCII mode), or 8 binary bits (in RTU Mode) that tell the slave what kind of action to take. Valid function codes are from 1-255, but not all codes will apply to a module and some codes are reserved for future use. [4]

2.5 MODBUS DATA FIELD

The data field provides the slave with any additional information required by the slave to complete the action specified by the function code. The data is formed from a multiple of character bytes (a pair of ASCII characters in ASCII Mode), or a multiple of two hex digits in RTU mode, in range 00H-FFH. The data field typically includes register addresses; count values, and written data. If no error occurs, the data field of a response from a slave will return the requested data. If an error occurs, the data field returns an exception code that

the master's application software can use to determine the next action to take. [4]

2.5.1 MODBUS ERROR CHECKING

MODBUS networks employ two methods of error checking: parity checking

1. Parity checking of the data character frame (even, odd, or no parity)

2. Frame checking within the message frame (Cyclical Redundancy Check in RTU Mode, or Longitudinal Redundancy Check in ASCII Mode). [4]

Parity Checking

A MODBUS device can be configured for even or odd parity, or for no parity checking. This determines how the parity bit of the character's data frame is set. If even or odd parity checking is selected, the number of 1 bits in the data portion of each character frame is counted. Each character in RTU mode contains 8 bits. The parity bit will then be set to a Zero or a One, to result in an even or odd total number of 1 bits. [4]

Frame checking

LRC Longitudinal Redundancy Check (ASCII Mode Only) In the ASCII transmission mode, the character frame includes an LRC field as the last field preceding the CRLF characters. This field contains two ASCII characters that represent the result of a longitudinal redundancy calculation for all the fields except the starting colon character and ending CR LF pair of characters. CRC Error Checking (RTU Mode Only) RTU Mode message frames include an error checking method that is based on a Cyclical Redundancy Check (CRC). The error-checking field of a message frame contains a 16-bit value (two 8-bit bytes) that contains the result of a Cyclical Redundancy Check (CRC) calculation performed on the message contents. [4]

2.5.2 MODBUS EXCEPTIONS

If an undefined function code is sent to a module, then the exception code 01 which means Illegal then Function will be returned in the data field of the response message. If a holding register is written with an invalid value, then exception code 03 will be returned in the

response message. [4]

2.6 MODBUS/TCP

MODBUS/TCP is a communication protocol designed to allow industrial equipment such as to communicate over an internet network. Modbus/TCP is one of the most popular protocols embedded inside the TCP/IP frames of Ethernet. Modbus/TCP basically embeds a Modbus frame into a TCP frame in a simple manner. It is a connection-oriented transaction, which means each & every query expects a response. This query/response technique fits well with the master/slave nature of Modbus, adding to the deterministic advantage that Switched Ethernet offers industrial users. The use of OPEN Modbus within the TCP frame provides a totally scalable solution from 10 nodes to 10,000 nodes without the risk of compromise that other multicast techniques would give. MODBUS® TCP/IP has become an industry "de facto" standard because of its openness, simplicity, low cost development and minimum hardware requirement to support it. There are more than 200 MODBUS® TCP/IP devices available in the market. It is used to exchange information between devices, monitor and program them. It is also used to manage distributed I/Os, being the preferred protocol by the manufacturers of this type of devices. MODBUS TCP/IP uses TCP/IP and Ethernet to carry the MODBUS messaging structure. MODBUS/TCP requires a license but all specifications are public and open so there is no royalty paid for this license. Making use of TCP/IP also offers the use of embedded Web pages which make life even more user-friendly! Simply by surfing to plant intranet for the information using web browser one can get the needed information. [4]

2.6.1 Performance from a MODBUS TCP/IP system

The performance basically depends on the network and the hardware. If you are running MODBUS® TCP/IP over the Internet, one won't get better than typical Internet response times. However, for communicating for debug and maintenance purposes, this may be per-

fectly adequate and save from having to catch a plane or go to site on a Sunday morning! For a high-performance Intranet with high-speed Ethernet switches to guarantee performance, the situation is completely different.

2.6.2 How can existing MODBUS devices communicate over MODBUS TCP/IP?

MODBUS® TCP/IP is simply MODBUS® protocol with a TCP wrapper. It is therefore very simple for existing MODBUS® devices to communicate over MODBUS® TCP/IP. To do this a gateway device is required to convert MODBUS protocol to MODBUS TCP/IP. [4]

2.6.3 Advantages of MODBUS/TCP

The key advantages of this protocol can be summarized as follows

• It is scalable in complexity. A device, which has only a simple purpose, need only implement one or two message types to be compliant.

• It is highly scalable in scope. A collection of devices using MODBUS/TCP to communicate can range up to 10,000 or more on a single switched Ethernet network.

• It is simple to administer and enhance. There is no need to use complex configuration tools when adding a new station to a Modbus/TCP network.

• There is no vendor-proprietary equipment or software needed. Any computer system or microprocessor with Internet style (TCP/IP) networking can use MODBUS/TCP.

• It is very high performance, limited typically by the ability of the computer operating systems to communicate. Transaction rates of 1000 per second or more are easy to achieve on a single station, and networks can be easily constructed to achieve guaranteed response times in the millisecond range.

• It can be used to communicate with the large installed base of MODBUS devices, using conversion products, which require no configuration. [4]

2.6.4 Conclusion

MODBUS is an application layer messaging protocol, positioned at level 7 of the OSI model that provides client/server communication between devices connected on different types of buses or networks. The industry's serial de facto standard since 1979, MODBUS continues to enable millions of automation devices to communicate. Today, support for the simple and elegant structure of MODBUS continues to grow. The Internet community can access MODBUS at a reserved system port 502 on the TCP/IP stack.

MODBUS is used to monitor and program devices; to communicate intelligent devices with sensors and instruments; to monitor field devices using PCs and HMIs; MODBUS is also an ideal protocol for RTU applications where wireless communication is required.

Chapter 3

Embedded Product Development

This chapter describes the flow of embedded product development life cycle; the stages are described below,

1. Requirement Analysis & specification
2. Feasibility analysis
3. Prototype Design & Development
4. Product Development
5. Product integration & Installation
6. Testing
7. Manufacturing
8. Product Operation & maintenance

3.1 Requirement Analysis & specification

Requirements analysis is critical to the success of Project, Conceptually, requirements analysis includes three types of activities: Eliciting requirements: (e.g. the project charter or definition), business process documentation, and client meetings. This is sometimes also called requirements gathering. Analyzing requirements: determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent

conflicts. Recording requirements: Requirements may be documented in various forms, usually including a summary list and may include natural-language documents; use cases, use stories or process specifications.

Specification given by client for project is described below, Desired Specification:

Size specification 48 x 48 mm

Universal input (TC, RTD, Volts, mA)

Accuracy 0.1% of F.S.

Retransmission with accuracy: 0.25% of F.S.

Fail-safe design protecting the process in case of system malfunctioning

RS485 Modbus Communication

Bright Red seven segment

Status Indication LEDs

Display brightness control

Transmitter Power Supply

Two PWM output

3.2 Feasibility analysis

According to specification the feasibility analysis has been done. In that two-three points come to the picture.

1. As Old design has limit number of features the controller used was with limited I/O lines, but as implementing to advanced specification extra i/o lines are needed in controller.

2. As size is another constraint of design we need to compact designing so instead of using on board extra ADC unit ,inbuilt ADC in MCU with good resolution is preferred

3. Cost is also the constraint so instead of using extra multiplexer in circuit I/O lines in MCU is preferred. After doing feasibility analysis of specification the major change in existing product line suggested was microprocessor unit change. According to feasibility analysis Renesas R5F100FC controller has been selected.

Figure 3.1: LC5296 Dual display on-off controller

3.3 Prototype Design & Development

In product development cycle before going to start actual designing it is always preferable to have prototype designing.in this case old temperature controller LC5296 has been chosen for Prototyping. LC5296 is on-off controller, in that MCU was ATMEGA32. LC5296 is divided in to three section, 1.Power supply section, 2.CPU & Display section and 3. Signal conditioning, communication & retransmission unit. There was two challenges, first to change microprocessor unit and according to that modification in CPU & Display section, another change is to re-size Product physically. At first level CPU & related circuit changes has been modified. Code has been written in C using Cubesuite+ software, Code development takes half of the period of designing in this it took about three and half months. As of now it is developed and successfully tested on prototype model. Now left with second challenge i.e. resizing task which has been scheduled for letter phase of PDLC.

Chapter 4

Product Designing

This chapter described hardware & software component of Temperature Process controller design with detail of each, it also include the block diagram of all circuit & also flowchart of Modbus protocol. **Hardware section** In this phase of designing according to Product specification all hardware parts selection is being made. Then after selecting it circuit designing is made, according to circuit designing PCB layout is done. Flow of hardware design is given below.

Stage 1: Enclosure selection

Stage 2: Component Selection

Stage 3: PCB Size & Shape finalization

Stage 4: Circuit Designing

Stage 5: layout

Stage 6: Mock up designing

Stage 7: PCB ordering & filling

Stage 8: Integration

4.1 Enclosure selection

According to clients specification they want 48x48mm front, but for depth there was no specific size given. Ventura make enclosure was selected, it has three option in 48x48mm front 48x48x72, 48x48x85 and 48x48x110.

4.2 Component Selection

Major Component Selection is done comparing the various parameters. Major components were OPAMP and MCU, Parameters considered for MCU selection were stability of output, noise margin, number of I/O pins, propagation delay and ADC resolution. Comparison of old and new MCU is given below.

Parameter	Atmega32-16AC	R5F100FCAFP#v0
Company Name	Atmel	Renesas
Core	AVR(8 bit)	RL78 (16-bit)
Program/RAM/Data Memory	32(flash)/2/1 KB	32(flash)/2/4 KB
Max. Frequency	16MHz	32MHz
I/O Lines	32	40
Interfaces	2-Wire, SPI, USART	LIN,UARTs,I2C,CSI
Internal ADC	8 bit	10 bit
Timer	3 Timers	4 (8 bit) + 8 (16 bit) + 1 (WDT) + 1 (12bit interval)
Additional Details	--------	PWM O/P(7 ch), RTC, Sub system clock ,Power on reset, Low Voltage Detection
cost	Around 1$	>= 1$

Figure 4.1: Comparison

4.3 PCB Size & Shape finalization

After selecting Enclosure Size of PCB, PCB Size is finalized, now as per the size of PCB some modification in existing circuit diagram is done. Shape of PCB is also playing major

role as PCB should be able to fit exactly in to enclosure, by keeping the facts in to mind that maximum area on PCB can be optimized.

4.4 Circuit Designing & block diagram

4.4.1 Display section

In Display section All Output signals are gathered here, out of Relay & PWM signals are also provided here in front panel.

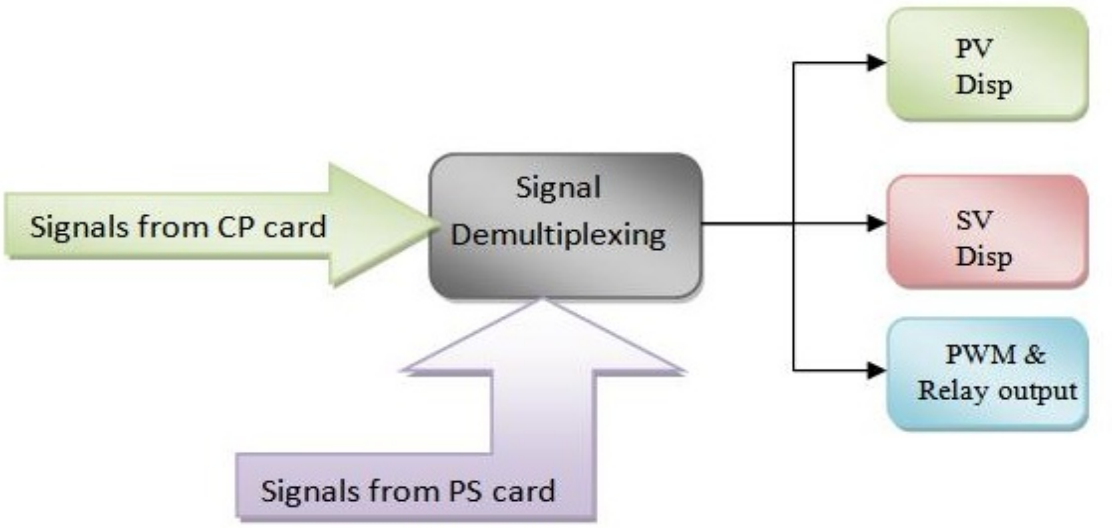

Figure 4.2: Display Section

4.4.2 CPU, Signal conditioning & retransmission section

This is the main section of whole device, because it contains the CPU which is the heart of the whole product design. Here CPU will process the output from all section & according to code it will modify or give the desired processed output in required form.

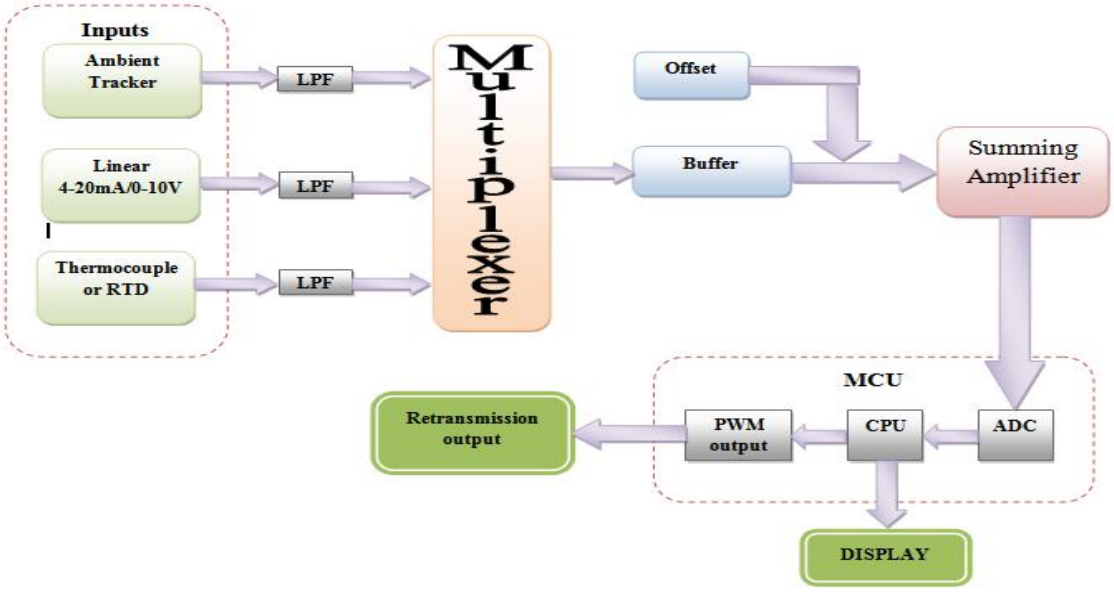

Figure 4.3: CPU, Signal Conditioning & retransmission section

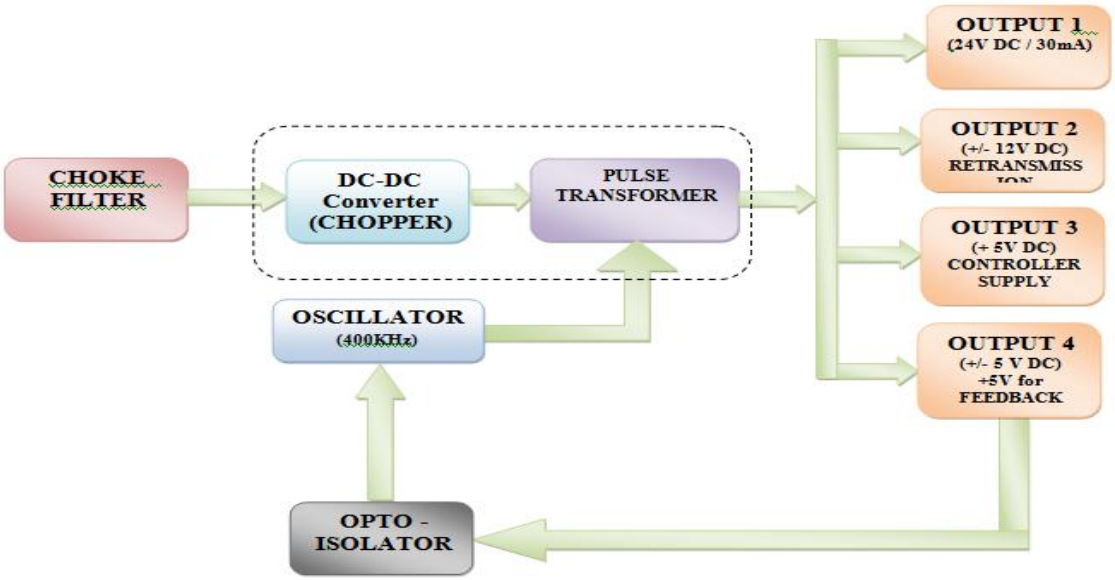

Figure 4.4: PS Section

4.4.3 Power Supply Section

Without Power supply section whole circuit designing is useless, it is very necessary part of circuit designing, which includes digital, analog & mixed power supply signals.

4.4.4 Retransmission & communication section (Add on cards)

Retransmission & communication sections are provided as different add-on cards, which are optional; retransmission section provides the ability to provide an input to other device, in simple words it is an ability of device to work as source by providing 4-20mA & 0-10 V standard output as input to other devices.

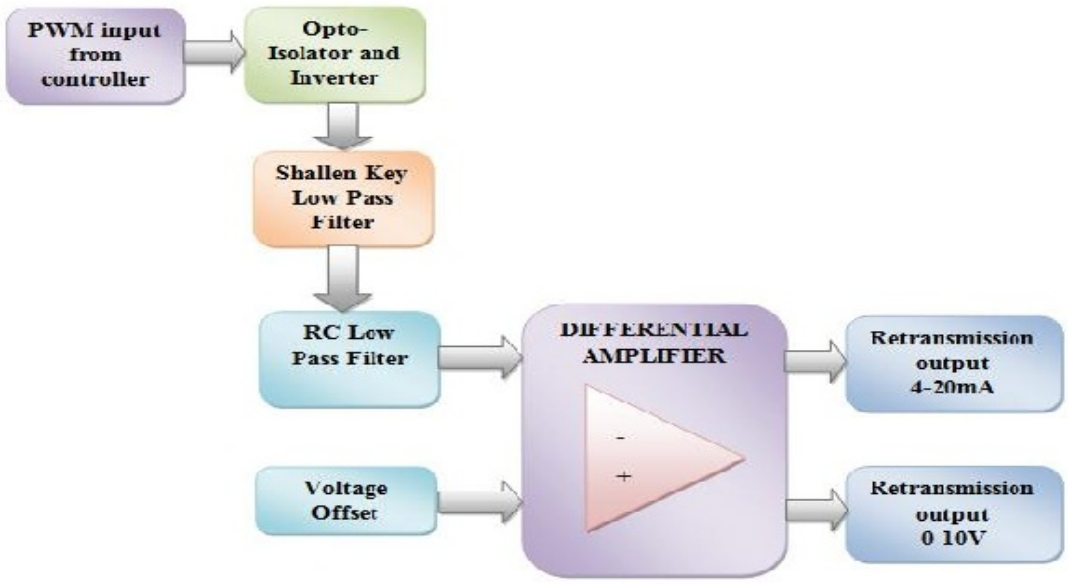

Figure 4.5: RET & Com Section

4.5 Layout

For Layout Designing at Masibus ORCAD 16.3 Tool has been used, and using the advanced features of ORCAD tool PCBs & Add on card has been designed. During layout designing some care should be taken care; some of those points are here:

1. When Digital circuitry & Analog circuitry are placed on single PCB, the minimum distance between analog & digital power signals should be maintain.
2. When Power supply PCB is designed the ground for digital & analog should be placed with bare minimum distance for best performance & noise minimization.
3. The signal should flow step wise no short cut for common connected point.
4. The OPAMP & MCU ICs should have bare minimum distance from transformer because the heat of transformer can decrease the performance.
5. Some TH components having wide footprint than SMD, if the performance is not decreased by replacement then try to replace it with SMD.

4.6 Mock up designing

After PCB designing mockup is prepared for the physical verification. Demo PCBs are made, footprint of component, spacing between two components, spacing from boundary, Spacing from transformer etc. are going to be checked at this stage of design.

4.7 PCB ordering & filling

After checking Demo PCBs, final order is take place & after receiving designed PCBs, component filling is done by manufacturing department. When final PCB is ready it sent back to R & D engineer for integration.

4.8 Integration

After receiving filled PCBs integration has been done, Program is being dumped using Renesas's E1 downloader.

4.8.1 Renesas E1 downloader

The E1 or E20 emulator is used by connecting it to the target MCU mounted on the user system. As by default setup we have to use 14 pin connector but as we need only 6 pin

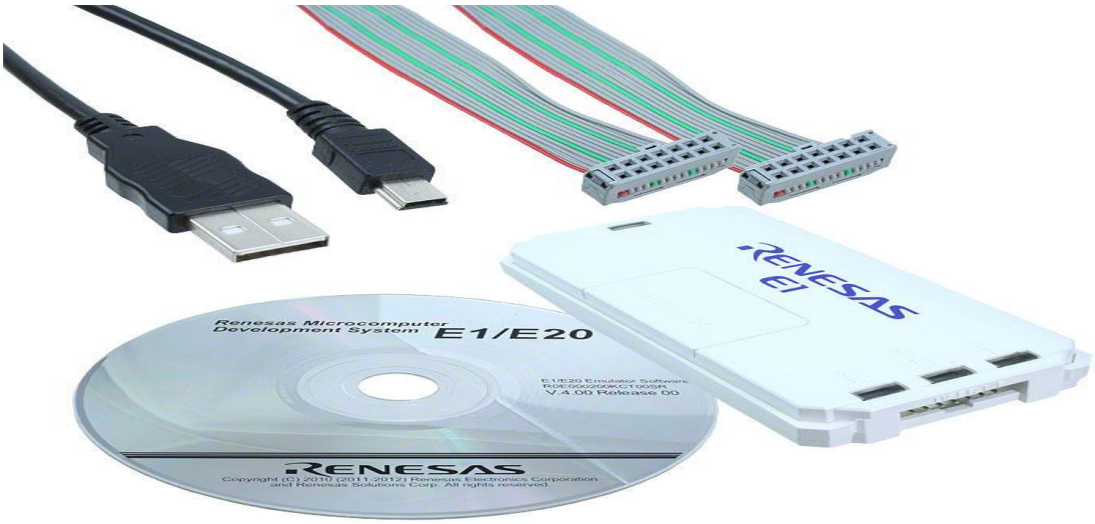

Figure 4.6: Renesas Downloader

for program another small 14 pin to 6 pin connector is made as intermediate connector. Software section

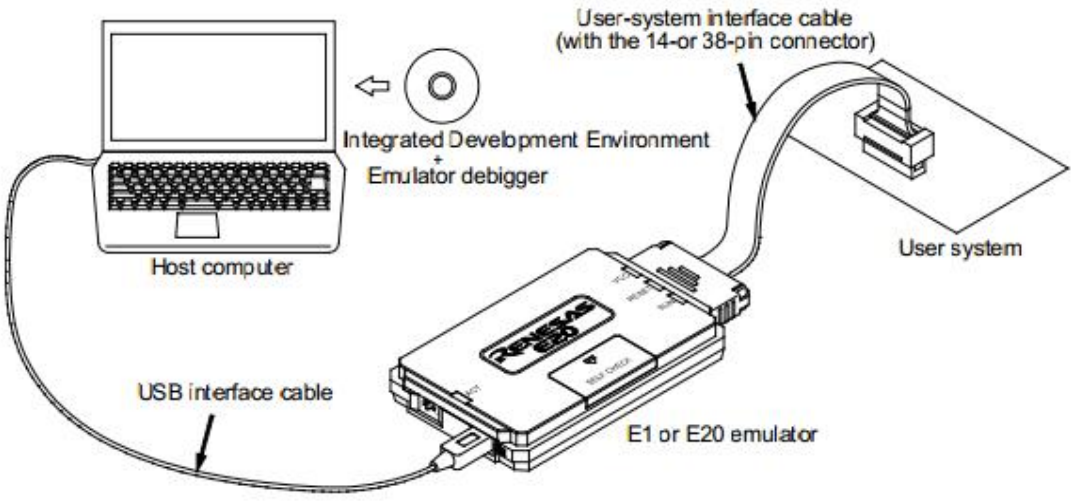

Figure 4.7: System configuration

4.9 Introduction to Software tools

This section contains basic information of all software used in this project for different purpose.

1. ORCAD Tool for PCB Layout & Circuit designing.
2. CUBESUTIE+ IDE For code Development

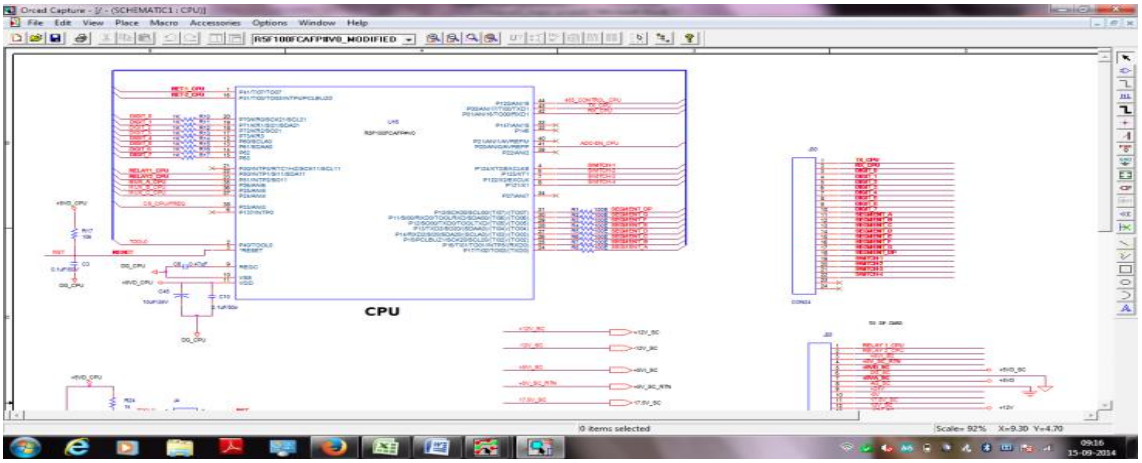

Figure 4.8: ORCAD Capture tool

layout.png

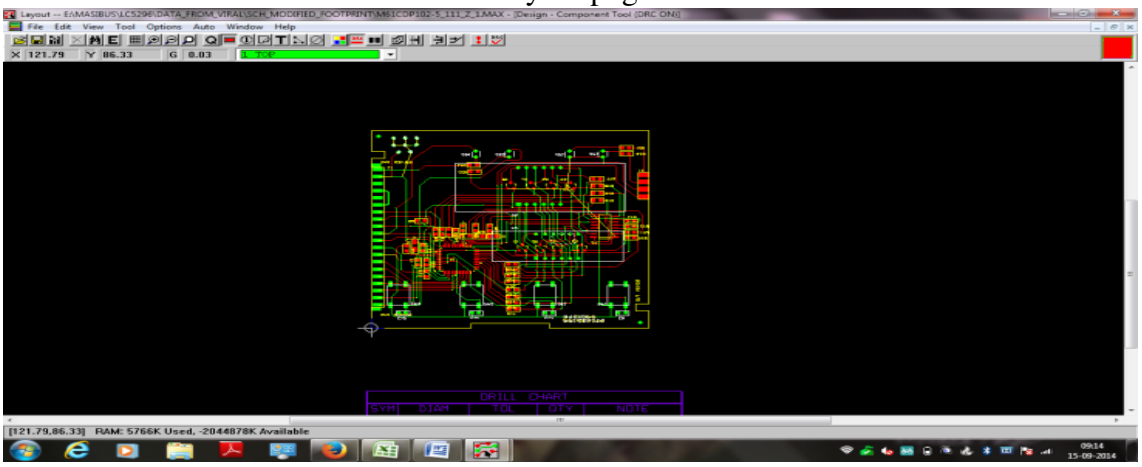

Figure 4.9: ORCAD Layout tool

3. Modbus tester for checking the Modbus communication protocol.

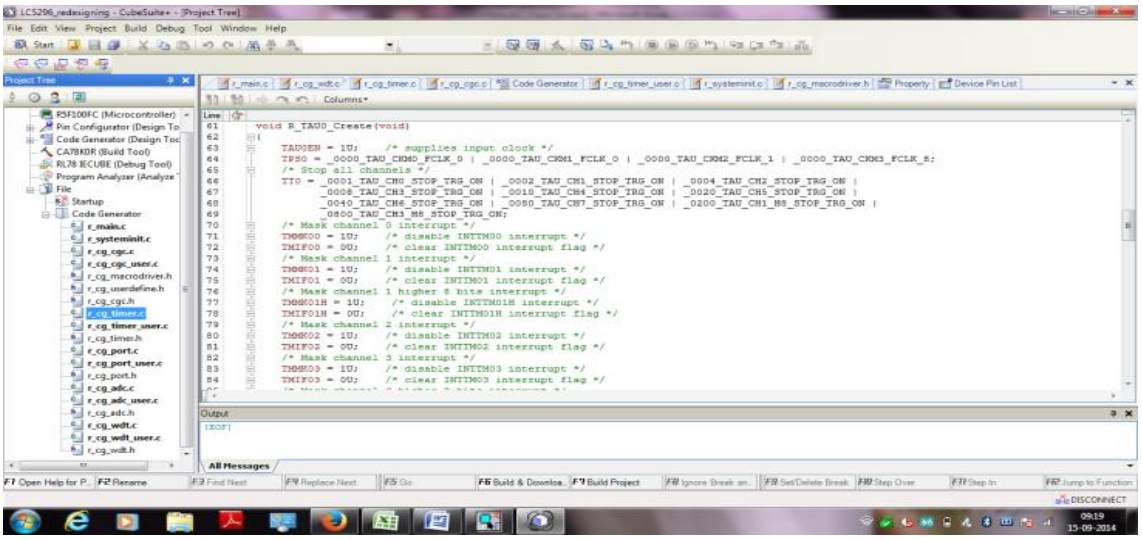

Figure 4.10: Cubesuite+ IDE

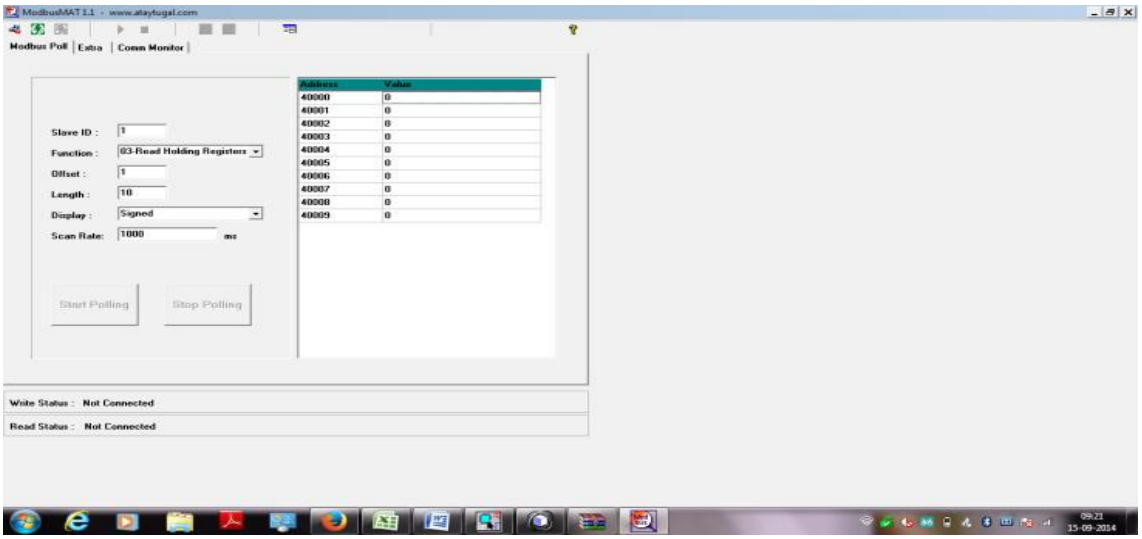

Figure 4.11: Modbus tester screen

4.10 Cubesuite+

As Renesas MCU has been used Cubesuite+ IDE has been choose for a coding. Cubesuite+ is an intelligent IDE in which is having advanced facility for ease of coding, like code generator. Integrated development environment CS+ offers, the ultimate in simplicity, usability, and security. For the repetitive editing, building and debugging that typifies software development. Easy to install and operate, CS+ offers a highly user-friendly development environment featuring significantly,shorter build times and graphical debug functions. The robust lineup of expanded functions and user support functions ensures a dependable environment for all users. [3]

4.11 Features of CS+

1. Project tree based on MCU development flow

CS+ positions the project tree to match the flow of MCU development. This sophisticated function enables the user to simply click on a node to move to the desired task. [3]

2. Single-page configuration window

Detailed settings for all items are consolidated in a unified control property panel. Simply select a node from the project tree and the display switches to facilitate the convenient search and configuration of settings. [3]

3. Simple Code Generation

CS+ is equipped with a "code generation function" that uses GUI settings to automatically generate a device driver program that controls MCU peripheral functions (timer, UART, A/D, etc.). User-friendly peripheral function setup facilitates smooth and problem-free program development, even when programming new MCUs (may not be available for some MCUs). [3]

4. Simple Pin List Creation

Configure the processing of dual-use pins using the pin list while confirming pin layout in the pin configuration diagram. Handy features include pasting the pin configuration di-

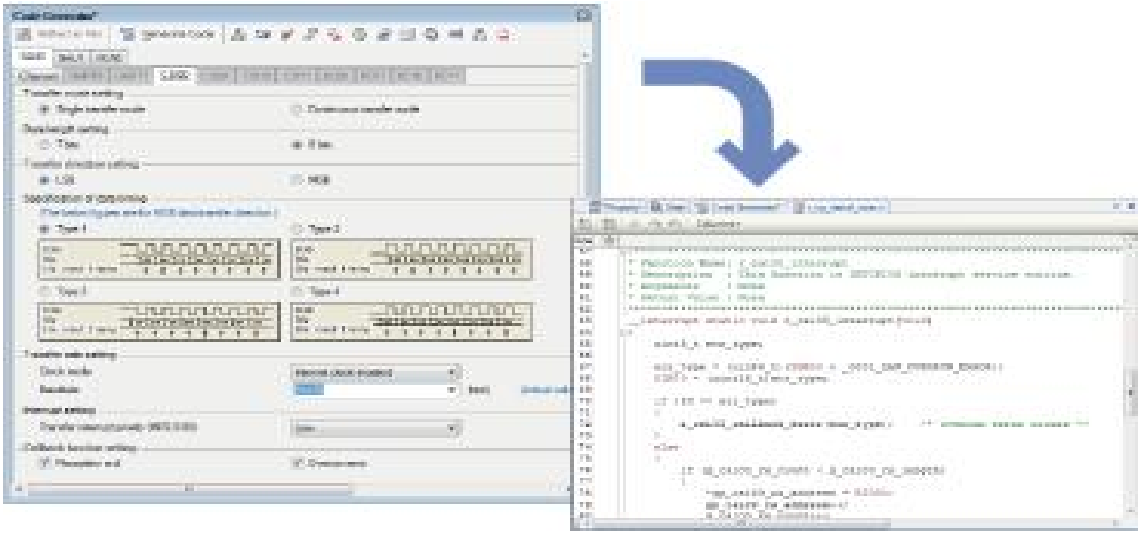

Figure 4.12: Code Generator Utility

agram in the design documents or outputting the pin list in an EXCEL file (may not be available for some MCUs). [3]

5. Reduced Build Time

Rapid build enables greatly reduced build time Conventional development environments require the user to edit all source files first, and then execute a build for the entire program, resulting in a lengthy build time.CS+ features a "Rapid Build" function that automatically runs the build function in the background each time a source file is changed or saved, greatly reducing overall build time. [3]

6. Action events eliminate the need for embedded debug printf

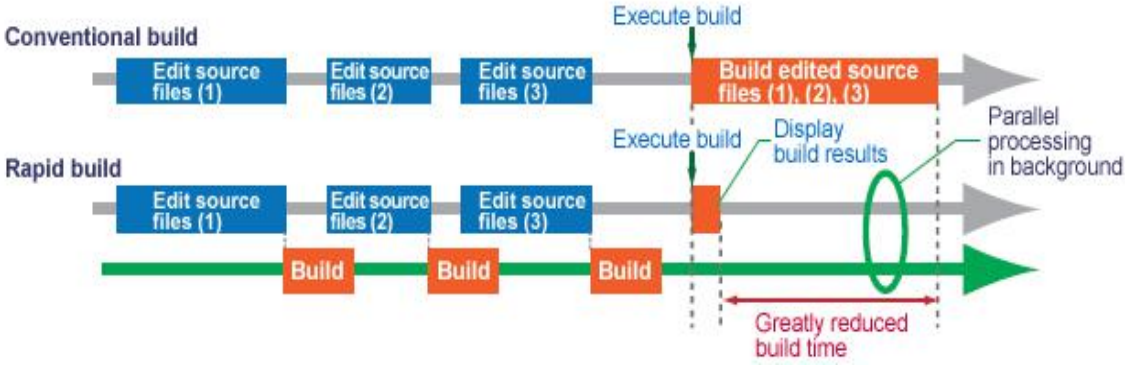

Figure 4.13: Comparison of Conventional build & Rapid build

Conventionally, debugging generally included a method for displaying the value of predefined variables through an embedded printf function; requiring a build operation each time the printf function was embedded. CS+ is equipped with the "action event function," for displaying variable values each time a program is executed at a specified address. Simply select the variable display function with a quick right click. Less time is eaten up by multiple build operations, greatly accelerating and enhancing the debug process. [3]

7. Graph function clearly displays variable- and function-related information

The ability to display variable values during program execution or graphs like an oscilloscope makes it possible to develop programs while performing monitoring of analog variations, as is necessary for sensor applications and the like. You can display the ratios of execution times among functions or a call graph to track down the processing associated with high loads. This makes it possible to improve overall system performance in a very efficient manner.

Other analysis functions include a source code display for functions and variables, CSV output of information, and the ability to search for all occurrences of specific definitions or declarations. These capabilities simplify administration and management even when the program structure is comparatively large-scale and complex. [3]

8. Repetitive tasks automated in script

CS+ embeds the "Python Console" function for automating repetitive operations executed before and after program download and after breaks; describing operations in the script enables the user to develop programs with greater ease and no operational mistakes. [3]

4.12 Basic understanding of Program

The program dumped in to the MCU can be divided in to several module, some of the important modules are discussed here.

1. Display, switches & LED read write functions

2. Processor initialization function for Timer, ADC, Watch dog counter & Ports

3. Modbus communication stacks implementation
4. Menu implementation

Chapter 5

Future-Scope

The Product is based on Modbus RTU Protocol if instead of that Modbus TCP/IP or another wireless network protocol will used then it will provide access over the network. Modbus TCP/IP let users to access and monitors the facilities at the remote site. It can be easily mounted on to a DIN-rail mounting rack, and it supports screw terminal block interface for power-supply and serial interface. With these merits for a true communication Data Gateway between Ethernet and Modbus, it is possible to integrate Ethernet and Modbus network within factory for resource sharing and better network integration.

Interconnection

Interconnection of previous work with new work can be described as follow: In first phase, the designing of embedded Product development for Modbus protocol was aimed to expand to study about Modbus TCP/IP network Protocol, which is very similar to IEEE 802.3 Ethernet protocol.Initially the work assigned in second phase is also about the study Architecture & working of Ethernet standard, which was similar to the work which was intended to do.Then in phase-two work is assigned on Ethernet switches Architecture & configuration options, which is led to the network switch simulator. Gradually the need of automation is detected so project is converted in to the automation of network switch simulator for several configuration options.

Chapter 6

Networking & Switching

In this chapter basic network topologies, networking jargon's & role of Broadcom in the world of networking & Switching is discussed, which also include the flow of work done in second phase at broadcom.

6.1 Background

Now a day's life without internet seems just impossible. Most important role of this internet service is played by switches & Routers. Network switch is used to connect two or more computers or devices within one local area network (LAN). Switches are incapable of connecting multiple networks or sharing an Internet connection. A router is a more sophisticated device than a switch. Traditional routers are designed to join multiple area networks (LANs and WANs). Routers serve as intermediate destinations for network traffic. They receive TCP/IP packets, look inside, decode its source & destination IP address for each packet, and then forward these packets as needed to ensure the data reaches to destination. Some latest device comes with switch & router together in one device.

Broadcom is also one of the company who is in switching technology, along with leading-edge software solutions, can be also found in a wide array of products for home and small business, data center, enterprise, and service provider networks. Broadcom has wide range of Switch Architecture but Broadcom StrataXGS® switches and XGS Core™ fabrics en-

able carrier class IP-based network solutions with optimal performance, quality-of-service and end-to-end subscriber bandwidth guarantees to client. These switches allow equipment providers to develop scalable, low power fixed and modular solutions for IP-based networks that provide carrier-class reliability for network. Broadcom StrataXGS® switch family having large number of chip which is provided to client as per their requirement with Software Development Kit (SDK) support. Hardware of switches is also known as 'Pizza box' which is customizable, as per customer's need based on Gigabit Ethernet (100-Gigabit Ethernet, 40-Gigabit Ethernet, 10-Gigabit Ethernet) or based on Fabrics (XGS Core® Fabrics, Dune Networks Fabrics). These switches are provided to client along with SDK. On the top of this SDK client can write their Application Program Interfaces. But if client wait till the physical availability of Switch for APIs then it may increase their time to market so to minimize the delay network switch simulator framework is provided to clients. This simulator can have exact & accurate behavioral model which looks & gives the feel like a real device. Network switch Simulator is also used by several internal teams (e.g. AE team: Application engineers team) .It has very complicated & tedious process of invocation. AE team has a responsibility to solve client's query regarding all switching device. Each time they need to set up all things in to lab for testing, these efforts can be minimized by using Network simulator.

6.2 Objective of work

To automate the process of network switch simulator initialization & invocation using script. Using this interactive script user will have the freedom to choose the different configuration options. SDK version will also be changed. Chip & SDK compatibility will also be checked. Based on user's input (with or without stacking option, number of chip, SDK & chip model, Debugging options) network switch simulator windows has been invoked and for debugging purpose GNU debugger (GDB) has been used.

6.3 Importance of work

AE team has to maintain average resolving time for customers' query, for that certain things which cost more time is identified & new techniques or use of certain tools which can help to reduce the time to resolve the customers query. The use of network switch simulator is also one of suggested option but initialization for a particular chip with particular configuration is very tedious & time consuming task. Some basic configuration option has been manually done for start up, but to avoid duplication of work & minimize time some task automation script was needed.

Chapter 7

Network Switch

In this chapter the basic of network switch is covered with its functionality & architecture of general network switch. Broadcom switch architecture with its SDK & customer APIs discussed in detail.

7.1 Introduction to Network Switch

Switch is a hardware device which uses to connect two device or Computers within a network. Function of Switch in network may implement power over Ethernet (PoE), which avoids the need for attached devices, such as a VoIP phone or wireless access point, to have a separate power supply. Since switches can have redundant power circuits connected to uninterruptible power supplies, the connected device can continue operating even when regular power supply fails.

7.2 Function of Switch at different layer

Layer 1: At layer 1 switches used as hub or repeater, the function at layer one is just to maintain end-to-end application or for "repeating" the traffic in network.
Layer 2: At Data link layer switch is used as "Network Bridge" which interconnects a small number of devices in small network. It is a trivial case of bridging, in which the

bridge learns the MAC address of each network connected device.

Layer 3: At network layer switch is known as “Router”, which can increase efficiency by delivering the traffic of a multicast group only to ports where the attached device has signaled that it wants to listen to that group.

Layer 4: At layer 4 switches used are mostly for load balancing, network address translator & TCP sessions are taken care. Device also responsible for including a stateful firewall, a VPN concentrator, or be an IPsec security gateway.

Layer 7: Layer-7 switches may distribute loads based on Uniform Resource Locator URL or by some installation-specific technique to recognize application-level transactions. A layer-7 switch may include a web cache and participate in a content delivery network.

7.3 StrataXGS® Switch

StrataXGS® III switches are the first to incorporate ubiquitous security, wire-speed IPv6 routing and wireless LAN support. Broadcom’s StrataXGS III switch architecture features an advanced multi-layer 72 Gigabit per second (Gbps) full-duplex packet processing architecture. StrataXGS III architecture enables seamless integration of a unified wireless and wired infrastructure [9].

Figure 7.1: StrataXGS Switch

7.4 Features

1. Enables unprecedented 10/40GbE single chip switch configurations 100+ 10GbE ports with flexibility to support up to 32 40GbE ports
2. First integrated switch delivers NVGRE and VXLAN L2oL3 transit and gateway switch technologies
3.Supports industry's highest equal cost multipathing-based fat-tree networking scale on a single chip
4. Greater FCoE network scale enabling true LAN/SAN convergence — up to 4X increase in forwarding entries
5. High port density with direct attach to SFP+/QSFP modules and KR Back-planes Integrated IEEE 1588 1-step timing solution
6. 3.2 Tbps multilayer Ethernet switching
7. Integrated low-power 25Ghz SERDES
8. Authoritative support for 25G and 50G Ethernet Consortium specification
9. Configurable pipeline latency enabling sub 400ns port-to-port
10. Supports high performance storage/RDMA protocols including RoCE and RoCEv2
11. BroadView instrumentation: provides switch- and network-level telemetry
12. High-density FleXGS flow processing for configurable forwarding/match/action capabilities
13. OpenFlow 1.3+ support using Broadcom OF-DPA
14. Comprehensive overlay and tunneling support including VXLAN, NVGRE, MPLS, SPB
15. Flexible policy enforcement for existing and new virtualization protocols
16. Enhanced Smart-Hash load balancing modes for leaf-spine congestion avoidance
17. Integrated Smart-Buffer technology with 5X greater performance versus static buffering
18. Single-chip and multi-chip HiGig solutions for top-of-rack and scalable chassis applications [9].

7.5 Categories of Applications

There are wide ranges of area where XGS switches are used, all major area of applications are listed below, Carrier and Service Provider Data Center Enterprise Home and Small Business Software Software-Defined Networking Solutions [9]

7.6 Architecture

In general architecture of switch is combination of hardware and software.

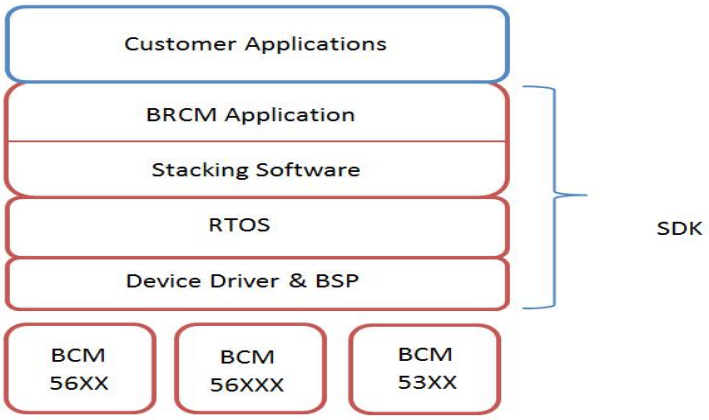

Figure 7.2: Architecture of Switch

7.7 SDK

SDK –Software Development Kit a Suite of Application Programming Interfaces (APIs) to facilitate the use of switching silicon.

7.7.1 Need of SDK:

1. Broadcom chips are complex, SDK is the component which makes integration of silicon with a customer's software achievable within a typical development cycle.

2. If the complexity of programming switching devices once “solve”, it can be can leverage that for every customer.
3. Once ported to Broadcom software, the advantages are two-fold Transition between current and future Broadcom chipset families is easier Transition away from Broadcom chipset families is harder

7.7.2 Components of SDK:

1. Device drivers & BSP:
All SDKs are not able to provide support to all the chips, this is the one of the control layer which makes this difference. As different chips are having different hardware so BSP for different family of hardware will be different from other families. Device drivers & BSP files are the basic difference between different SDKs which supports different chipsets or families of switches.
2. RTOS: A Multithreaded Real time Operating System is needed to take the control & handle of everything. Current versions of the SDK cannot function without the following operating system support: Threads, Semaphores, Recursive Mutexes, Timers and Dynamic memory. The SDK is written for system independence, so all hardware, PCI, and configuration specific information must be managed and provided to the SDK by the application, while the SDK requires an Operating System, it is designed to be independent of any particular one. The SDK dependency on the OS is abstracted through the use of a software interface layer known as the SAL; any underlying OS which can implement the requirements of the SAL can be used with the SDK. Our customers have successfully ported the SDK/SAL themselves to many different OS platforms like: QNX, Nucleus, BSD, Linux 2.6, eCos and some of Customer Proprietary Operating Systems. (As per customers Requirement different Operating System support is provided by SDK team.)
3. Stacking applications:
It is used to Provides dynamic topology discovery and management for stacked systems, where the more than one chipsets are involved. It is useful for multi CPU/PCB or dynamic

configuration kind of environment. This applications set makes SDK multi-chip and multi-CPU aware.

4. BRCM Applications:

This is the basic building block for systems which using Broadcom software, this applications are written to provide a high level programming interface covering all of the features of devices & Provide the same programming interface for each one of them, making them interchangeable from a software perspective.

7.8 Broadcom as a silicon Vendor

Customers applications:

This layer is not comes under SDK but can be considered as an essential part of switching Architecture. This is something that is written or codded by customer, which is written on the top of the SDK, which plays important role for the functionality of same chip with same SDK version. Due to this layer only switch becomes vendor specific. In the figure,

Figure 7.3: Broadcom as a silicon vendor

all the shown chips are vendor specific, which may function as per the code of customers end applications, but base of all this switches are same i.e. Trident2+.

Chapter 8

Automation of network simulator configuration

In this chapter the Purpose of Automation is described as well as also the whole methodology for automation is covered.

8.1 Introduction to Network Switch Simulator

The Network Switch Simulator at Broadcom is a framework and a collection of tools to allow the rapid specification and reconfiguration of interconnections of network components to prototype and validate architectures and software. This framework provides support to start and stop the simulation, to manage the simulation components and to manage the connections between the components. The most important parts of the simulator environment are the device models of Broadcom silicon. These are distributed as binary executable which communicate over sockets. These models have two interfaces.

1. CPU interface that communicates to a CPU emulation using the PCID protocol that has been distributed with the SDK.

2. All other ports communicate across a socket interface which encapsulates Ethernet and HiGig frames.

The block diagram show the logical components of The Network Switch Simulator frame-

work and two BCM devices connected other via simulator framework

Other simulation components may include custom packet handling device simulators, soft-

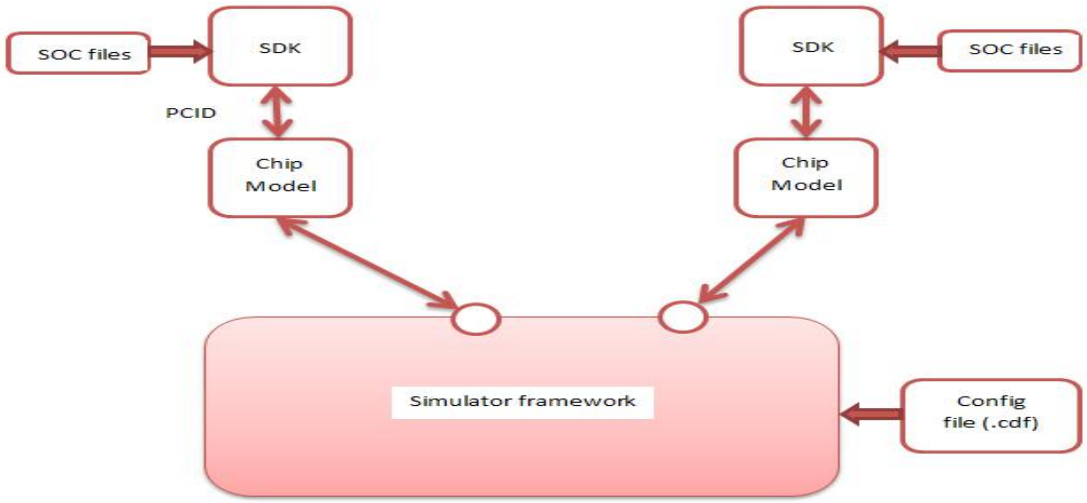

Figure 8.1: The Network Switch Simulator Framework

ware packet generators or interface shims to allow communication to network hardware. In addition to these components, there are two other important parts of the network switch simulator.One of these components reads the system configuration from a file and initiates the components for a simulation run. Other component is responsible for interconnections between simulation components.

8.2 Testing Environment

At AE team, for testing purpose whatever general setup they have to do in lab is shown here. Each time when new chipset or switch for particular configuration setup needs to set, all this exercise has been done. For this setup AEs have to go physically inside the lab, have to arrange all hardware & cables have to invest few hours.

Figure 8.2: Testing Environment

8.3 Ease of Access due to Simulation

The Network Switch Simulation provides the feel of real hardware so without taking all this efforts, for some of the testing event which is not related with performance of switch but related with behavior of switch can be configure using simulator. Instead of having physical switching devices, using Switching simulator, one can have exact & accurate behavioral model which looks & feels like a real device. Moreover one can easily do software implementation to start before the real silicon is available.

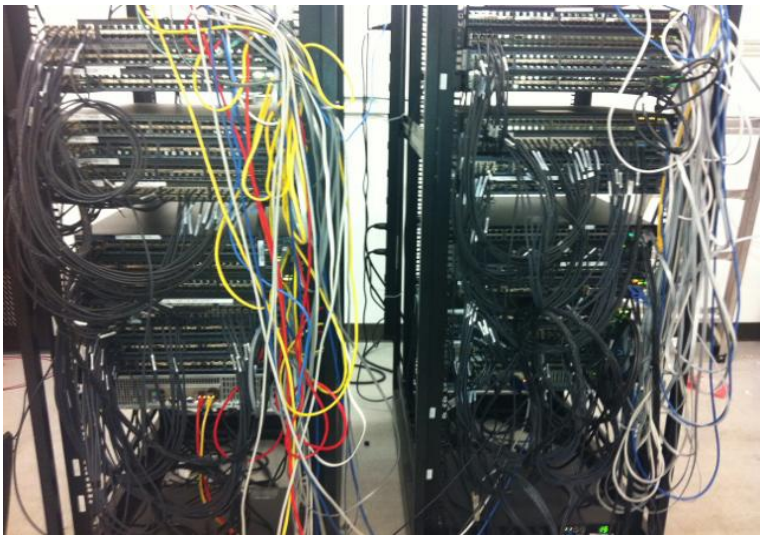

Figure 8.3: Back panel

8.4 Need of Automation

Using Network Switch Simulator, to set the particular switch configuration need to set several environment variables as well have to run several commands on Linux server farm. This process of use of Simulator is quite tedious & not much user friendly. To minimize user efforts automation with interactive script is needed which guide the user to abstract the other steps. To save efforts of understanding simulator from scratch, now just user need to add only chip model number, configuration details and debug options only, rest of things will be taken care by script. If this kind of script can be written then it will save time of understanding as well as implementation.

8.5 Scripting Language selection Criteria

To write such script which language should be used was a big question. The points to be considered before selecting a scripting language were:

1. It should be easy to embedded

2. It should be compatible with C++.

According to Criteria Ruby was the first language which comes in mind, as the network

switch simulator framework is written in ruby & C++. Shell was the second choice.

8.6 Scripting with Ruby

8.6.1 Ruby: As a Scripting Language

It is a dynamic, reflective, object-oriented and general-purpose programming language. Ruby embodies syntax inspired by Perl with Smalltalk-like features and was also influenced by Eiffel and Lisp. Alternative Ruby: There are a number of complete or upcoming alternative implementations of Ruby, including YARV, JRuby, Rubinius, IronRuby, MacRuby (and its iOS counterpart, RubyMotion), mruby, HotRuby, Topaz and Opal. Each takes a different approach, with IronRuby, JRuby, MacRuby and Rubinius providing just-in-time compilation and MacRuby and mruby also providing ahead-of-time completion.
Ruby gems:
A Ruby has package manager called “Gem” which is generally third party libraries or program applications such as IDEs. There are around 1.5 lacs gems hosted on its official website.

8.6.2 Intro to Ruby Gem: Cucumber

Cucumber is the testing tool which used for Behavioral driven development, Cucumber is a command-line tool. Cucumber has its own ubiquitous language. The easy readability of Cucumber tests draws business stakeholders into the process, helping you really explore and understand their requirements. It has nice structure which increases its readability, it describes its behavior in three steps: Features, Scenarios & steps definition. The scenarios are defined using Given, When and then by very simple plain text, which is also understood by non-technical persons. Next is step definitions, which is actual technical coding behind each Scenarios.

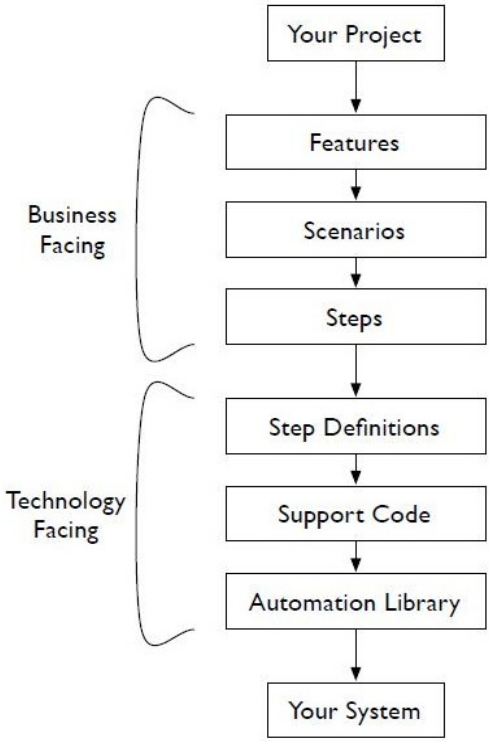

Figure 8.4: Cucumber Stack

8.6.3 RubyMine IDE

RubyMine is IDE for Ruby Projects, which provides test frameworks support with test runner GUI for some of ruby gems e.g Cucumber. It is graphical Ruby Debugger which supports for multiple project support.

8.6.4 Problems with RubyMine

RubyMine IDE has a nice GUI, which consumes very large amount of run memory as compared to terminal. The Network Switch Simulator which is also required large amount of memory at runtime, which results in exceeds limit of dedicated memory on server. In this case LSF used has maximum limit of 2 MB, but due to the use of this heavy IDE it reaches up to 6.5 MB. These problems resist the use of RubyMine & hence ruby gem cucumber is eliminated for scripting. Now the second option was simple terminal operated shell as a scripting alternative.

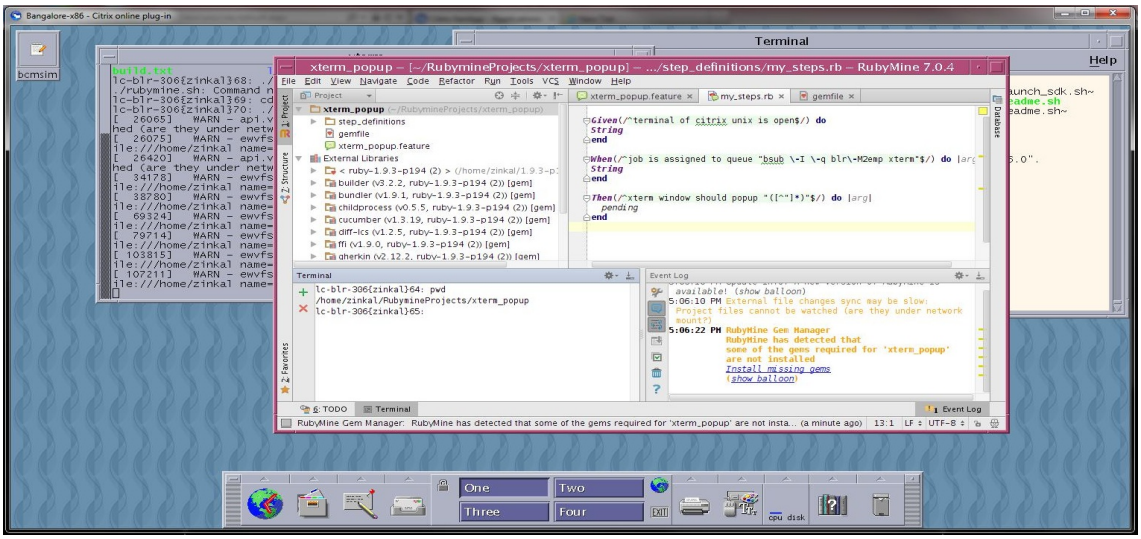

Figure 8.5: RubyMine IDE

8.7 Scripting with Shell

8.7.1 Shell: As a Scripting Language

Shell Scripting is a program written in text editor to run on UNIXTM or Linux shell, a command line interpreter.it is actually a list of command written in the order of execution which will run in shell environment of host OS.

8.7.2 About Shell script

Shell scripting is generally considered to be a glue language; it is the set of UNIXTM command putted inside the text file. Shell is ideal for creating small pieces of code that can be used to connect other tools together. When shell scripts can be used for more complex tasks, usually not the best choice. Shell Programming is:

- almost always special-purpose code
- often one-time code
- seldom used where speed is important
- often used to manipulate files

Shell Script has a structure like this:

#! /bin/sh # this is a comment

body of program

to continue a line append

this is the rest of the continued line

exit 0

From the text file shell script can be identify from its first line that is #! /bin/sh.

8.7.3 Comparison of Ruby & Shell

Here comparison shown is for this project work whatever data & information written is with respect to this project work.

Parameter	Which language is better? (Ruby/shell)
Less Memory consumption	Shell (< 2MB) (For ruby its 6.51 MB)
Speed	Ruby
Readability	Shell
Less complex	Shell
Flexibility	Ruby
Handle of terminal emulator	Shell

Figure 8.6: Comparison of Ruby & Shell

8.8 Implementation of Script

In shell script written for automation of the Network Switch Simulator, following options are implemented;

1. Dynamic SDK

User can provide the location of its own copy of built SDK, Script will take that Path; use that location and set environment location accordingly. 2. Stacking Options for chip

User can choose single or multiple chip for stacking option, for single chip user has to enter

1, for stacking has to enter 2 or 3.

3. GDB option for SDK

User can select GDB attachment option for SDK, if it is selected with GDB then with all other environment variable GDB is also set to 1, while invoking SDK.

4. GDB option for Simulator

User can select GDB option for simulator, for that there are some steps should be follow, for those steps one window will pop up to guide user that what he should do the next.

8.9 Troubleshooting

While implementing the script the following problems are faced.

1. Problem of runtime memory violation for LSF while using RubyMine IDE.

2. Getting control of particular window using script

3. Sequence of window popup

4. For multichip options SDK window is getting closed immediately after popup

5. Swapping the focus of current window with another to run particular command using script

8.10 Flow chart

The Flow chart of this script is shown below which included some file call, that file call are explained separately. The overall steps for automation of the network switch simulator for different configuration from invocation is described below.

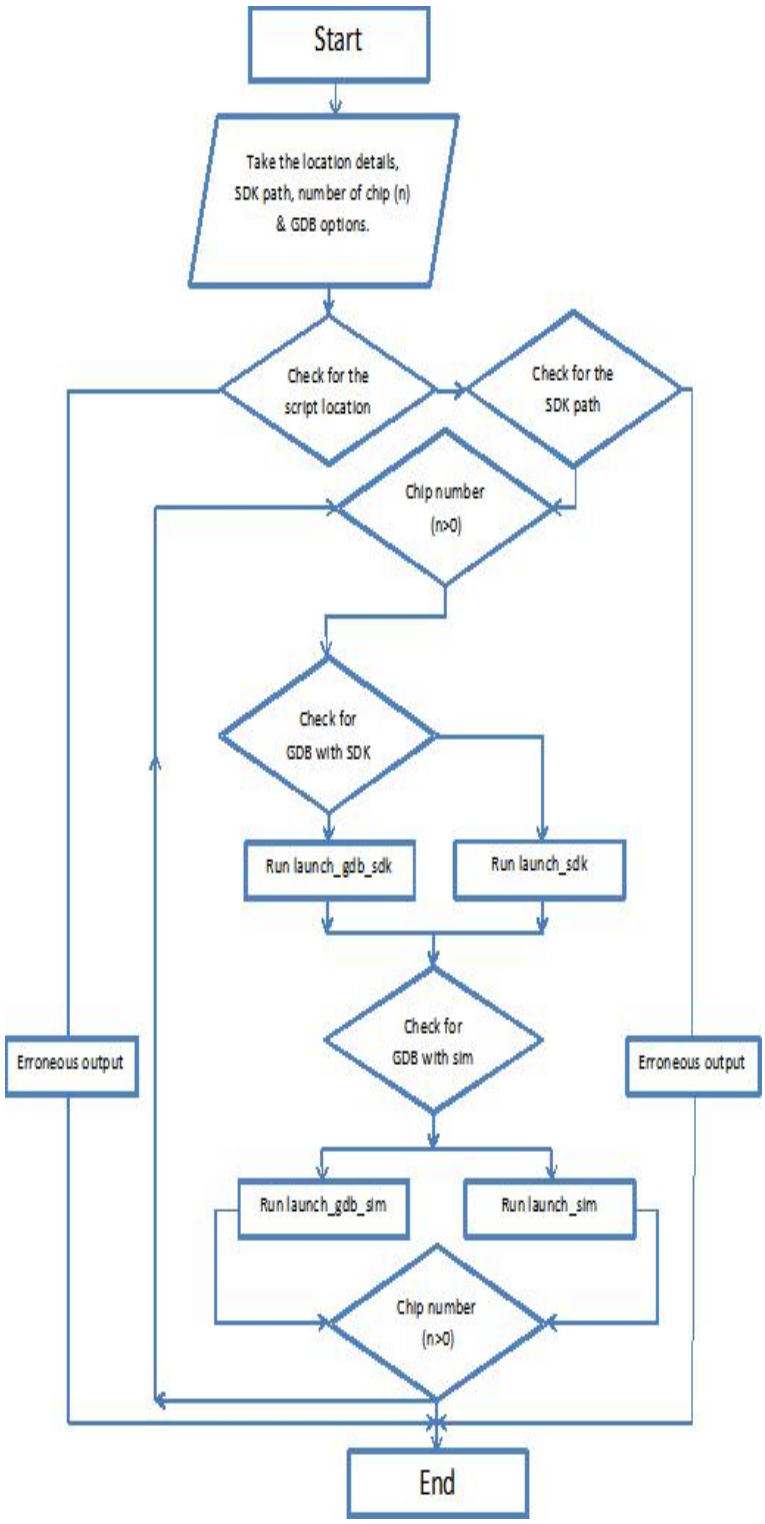

Figure 8.7: Flow chart

8.11 Different File calls

1. Launch_gdb_sdk

This file contains the environment variables which are required to set for SDK invocation with GDB, Also having some Flag details for this configuration.

2. Launch_sdk

This file contains the environment variables which are required to set for SDK invocation without GDB, Also having some Flag details for this configuration.

3. Launch_gdb_sim

This file contains the set of commands which required for running the simulator, which includes command which run ruby script with provided .cdf file and launch the sim executable with GDB attached to it.

4. Launch_sim

This file contains the set of commands which required for running the simulator, which includes command which run ruby script with provided .cdf file and launch the sim executable without GDB attached to it.

Chapter 9

Conclusion

This chapter will conclude the project work and also the future scope of this work is discussed.

9.1 Conclusion

Using this automation script within a fraction of minutes SDK & simulator for different configuration options can be invoked with least user interactions. This script is written in basic shell script, with several internal file calls which slower down the process, but as here this much speed is not required so shell turned out as a good option.

9.2 Future scope of work

There are many ways to implement & enhance script, Major where one can focus are:
1. Script initially aimed to be writing in ruby, but to some problem with its IDE RubyMine, that idea was quit due to time duration of project; but can be done with light weight less GUI featured IDE.
2. In Future if Simulator itself is enhanced & if one tries to get inside the code of simulator than some of the issues which is solved temporally can be fixed in better way.

References

[1] LC5296, 5006RN, LC5296H–On-Off Controllers:
www.masibus.com//index.php/products/controllers/on-off-controllers
/lc5296-dual-display-on-off-controller

[2] Control Station:
Instrumentation Text book

[3] Cubesuite–plus installation guide:
www.renesas.com/products/tools/ide/idecubesuiteplus/index

[4] Modbus details:
www.sena.com/modbus/detaills

[5] Merchant Silicon and Vendor Software – The Hype in 2012:
etherealmind.com/merchant-silicon-vendor-software-rise-lost-opportunity/

[6] The Network Makes or Breaks the Cloud:
www.broadcom.com/products/features/cloudscalenet.php

[7] Tear–Down of an HP ProCurve 2824 Ethernet Switch
blog.thelifeofkenneth.com/2013/02/tear-down-ofhp -procurve-2824-ethernet.html

[8] Switching:
www.broadcom.com/products/Switching

[9] Broadcom website:
http://www.broadcom.com