PROFIBUS PROTOCOL IMPLEMENTATION OVER mINTS

Major Project

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

Apurva V Patel
(12MECE13)

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

May 2014

PROFIBUS PROTOCOL IMPLEMENTATION OVER mINTS

Major Project

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

Apurva V Patel
(12MECE13)

Under the guidance of

External Project Guide:

Mr. Milap Patel
Manager,I/O group r&d ,
Masibus Automation and Solutions,
Gandhinagar.

Internal Project Guide:

Prof. Vijay Savani
Assistant 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 2014

Declaration

This is to certify that

1. 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.

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

- Apurva V Patel

Certificate

This is to certify that the Major Project entitled **"Profibus Protocol implementation over mINTs"** submitted by **Apurva V Patel (12MECE13)**, towards the partial fulfilment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by him 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. Vijay Savani
Internal Guide

Mr. Milap Patel
External Guide

Dr. N.P. Gajjar
Program Coordinator

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

Dr. K. Kotecha
Director, IT

Company Profile

Masibus is head-quartered in Gandhinagar, Gujarat, since 1991, Masibus offers industrial automation & instrumentation solutions to customers within the country & across the globe. Recognized as one of the premier industrial automation solution provider, Masibus has close to 10,000 customers, in about 50 vertical industrial segments, offering products, solutions & services through 8 regional offices & a wide network of Dealers & System Integrators.

Masibus′*s* goal is to exceed the expectations of every customer by offering innovative industrial automation solutions, outstanding customer service & greater value addition, there by optimizing process functionality and improving operation efficiency of the industry. Our team is distinguished by their functional and technical expertise combined with their hands-on experience, thereby ensuring that our customers receive the most effective & value based solutions for their requirements.

Masibus has been quick to adopt & incorporate new technologies like Ethernet & Wireless into the industrial automation solutions that are offered. Today we have many proven track records of every conceivable product and solution in SCADA/PLC/DCS platforms that have been successfully installed at several sites.

Masibus has been the preferred supplier of products and solutions to the power sector also. We have recently tied up with Siemens-EDEA to offer complete sub-station monitoring & automation solutions. This tie up enhances Masibus′*s* offerings to the power sector and we now have products and systems to offer for Power Generation, Transmission & Distribution segments too. Masibus has recently become exclusive dealers for world class Calibrators from Beamex, Finland & Vibration Products from VMI, Sweden. The promise of quality has helped Masibus in becoming a globally accepted company with increasing exports to countries world over. Masibus′*s* global journey has begun with the opening of an office in Sharjah.

Acknowledgements

I would like to express my gratitude and sincere thanks to Dr. P.N.Tekwani, Head of Electrical Engineering Department, and Dr. N.P.Gajjar, PG Coordinator of M.Tech Embedded Systems program for allowing me to undertake this thesis work and for his guidelines during the review process.

I am deeply indebted to my thesis supervisors **Prof. Vijay Savani,Assistant Professor**, E.C.Dept., Nirma University and Mr. Milap Patel,R&D manager at Masibus Automation and Solution for their constant guidance and motivation. I also wish to thank Mr. Ashok Rabadiya, Engineer, Masibus Automation and Solution, Mr. Suhant Raval,R&D manager and all other team members at Intel for their constant help and support. Without their experience and insights, it would have been very difficult to do quality work.

I wish to thank my friends of my class for their delightful company which kept me in good humor throughout the year.

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.

- Apurva V Patel
12MECE13

Abstract

Industrial field work required monitoring and attaching of electrical heavy equipments, measurements and analysis of electrical functions and systems. Evaluatoin of equipment and communicating data security are required. mINT I/O is qualified for all requirements.

mINTs are innovative simple low cost solution for distributed I/O requirements. The I/O system consists of stand alone Digital and Analog - Input and Output modules which are connected together on RS 485 two wire multi-drop network. The modules communicate using the MODBUS-RTU protocol All (Digital and Analog) IO modules plug directly onto an industry standard.

Modbus and Profibus are popular field communication protocols.Now a days Profibus DP technique is widely applied in the field of industrial control.Profibus supports factory and process automation as well as drive applications with the same consistent communication protocol named Profibus DP. which is lake in modbus loaded mINT devices. This enables mixed (hybrid) applications, where continuously running processes. A Profibus system uses a bus master to poll slave devices distributed in multi-drop fashion on an RS485 serial bus. This thesis shows a construction of modbus-RTU to Profibus-DP converter. with the details of its initialisation software.

Abbreviation Notation and Nomenclature

CRC ..... Cyclical Redundancy Check
CS ..... clock synchronisation
DDLM ..... The direct data link map
DP ..... descrete periferals
DXB ..... data exchange broadcast
ESD ..... Electro Static Discharge
FMS ..... Field Message Specification
GSD ..... GerteStammaDaten: equipment master data
IEC ..... International Electrotechnical Committee
ISO ..... Isochronous mode
JTAG ..... Joint Test Action Group
mINT IO ..... masibus intelligent IO
POR ..... Power on reset
RSTIN ..... Reset In
RTU ..... Remote Terminal Unit
SAP ..... Service Access Point
SPI ..... Serial Periferal Interface
xPEC ..... external protocol eccess control

Contents

Declaration iii

Certificate iv

Company Profile v

Acknowledgements vi

Abbreviation Notation and Nomenclature viii

List of Tables xi

List of Figures xii

1 Introduction 1
- 1.1 mINT I/O . . . 3
- 1.2 Thesis Organization . . . 4

2 Profibus-DP protocol and Modbus RTU protocol 6
- 2.1 Modbus RTU Protocol . . . 6
  - 2.1.1 Frame format . . . 7
  - 2.1.2 Modbus data model . . . 10
- 2.2 Profibus Protocol . . . 11
  - 2.2.1 Structure of Profibus protocol . . . 12
  - 2.2.2 Profibus Algorithm . . . 13
  - 2.2.3 Frame Structure . . . 16
  - 2.2.4 GSD file description . . . 16
  - 2.2.5 How profibus Does work ? . . . 18
- 2.3 Modbus to Profibus . . . 19
- 2.4 Summary . . . 19

3 Hardware design 20
- 3.1 Physical structure design . . . 20
- 3.2 Hardware block diagram . . . 21
  - 3.2.1 netX 100 controller . . . 21
  - 3.2.2 Modbus slave . . . 22
  - 3.2.3 Configuration Software . . . 23
  - 3.2.4 Power on reset . . . 23
  - 3.2.5 Profibus interface circuit . . . 24

3.2.6 Serial flash . . . 25
3.3 Summary . . . 27

4 Firmware Functional details **28**
4.1 Blockdiagram . . . 28
4.1.1 Initialize ports . . . 28
4.1.2 Modbus Slave . . . 29
4.1.2.1 Initialize Modbus slave parameters . . . 29
4.1.2.2 Start Modbus save . . . 29
4.1.2.3 Communication with Configuration software . . . 29
4.1.3 Modbus Master . . . 30
4.1.3.1 Initialize Modbus master parameters . . . 30
4.1.3.2 Start Modbus master . . . 30
4.1.3.3 Receive response of slave and collect data . . . 30
4.1.3.4 RAM communication Buffer block . . . 30
4.1.4 Profibus Slave . . . 31
4.1.4.1 Initialize Profibus . . . 31
4.1.4.2 Initialize Profibus global firmware parameters . . . 31
4.1.4.3 Start Profibus . . . 31
4.1.4.4 Detect frame request from Profibus master . . . 31
4.1.4.5 Response to the master . . . 32
4.2 Communication Block . . . 32
4.3 Summary . . . 34

5 Softwares **35**
5.1 Hitex HiTOP . . . 35
5.2 mINT Profibus . . . 35
5.3 Profisim . . . 39
5.4 Summary . . . 42

6 Result **43**
6.1 Profibus to Modbus conversion . . . 43
6.2 Configuration software implementation . . . 45

7 Conclusion and Future work **46**
7.1 Conclusion . . . 46
7.2 Future Scope . . . 46

References **47**

List of Tables

2.1 Modbus data model 10
2.2 DAP and SAP description 17

6.1 Modbus Query to send 44

List of Figures

1.1 mINT I/O [3] . . . 3

2.1 Mater Slave query Response[5] . . . 7
2.2 Modbus frame . . . 9
2.3 Profibus structure [6] . . . 13
2.4 Profibus structure . . . 14
2.5 Profibus frame structure . . . 16
2.6 Profibus working process[8] . . . 18

3.1 Profibus Hardware Blockdiagramn . . . 20
3.2 Profibus Hardware Blockdiagramn . . . 21
3.3 Configuration software . . . 23
3.4 Power On Reset[9] . . . 25
3.5 Profibus Interface Circuit . . . 26
3.6 Serial flash communication[9] . . . 26

4.1 Software functional flow diagram . . . 28
4.2 Data communication process block . . . 33

5.1 HiTOP main screen . . . 36
5.2 HiTOP menu screen . . . 36
5.3 netX configuration tool . . . 37
5.4 mINT profibus thirdparty configuration . . . 38
5.5 mINT profibus download and upload configuration . . . 39
5.6 mINT profibus download and upload firmware configuration . . . 39
5.7 Profibus master simulator . . . 40
5.8 Open GSD file . . . 41
5.9 Search slave . . . 41

6.1 Profibus Result . . . 43
6.2 Modbus Result . . . 44

Chapter 1

Introduction

With the rapid development of modern science and technology, the automation technology changes each and every passing year. In the field of industry, the performances of automation systems are principally regarded as the key points to improve the quality and increase the speed of the production. The properties of the communication network directivity influences the performances of the automation systems. Where, various types of field devices are mounted on the network, and large number of the data among them are exchanged and transmitted according to the realistic demands. It should be appreciated that, the field bus technology contains lots of subjects which are digital communication, computer networks, automatic control and intelligent instruments etc., with the field bus technology , together intelligent instruments and equipments, they possess a huge market prospect. Difference with the traditional field instruments, intelligent ones themselves can generate analog and digital conversion in nature and digital signals are directly uploaded to upper pc.[1]

mINT IO (masibus intelligent IO) modules area innovative which provides a standalone digital and analog input / output modules which are connected together on RS485 two wire multi drop network. The modules communicate using the MODBUS RTU (Remote Terminal Unit) protocol. A 16-Bit controller is used in the modules to provide high speed data processing and fast communications turnaround times. Multiple baud rates are selectable from 9600 to 115200 baud. This the analog and digital, input and output kind of different kind of intelligent devices works with the Modbus RTU protocol, which is a de facto standard communication protocol. It is

common available industrial device protocol. Reasons for the extensive use of Modbus in industrial environment are that is developed with industrial applications in mind, and openly published and royalty free. This is also easy to deploy and maintain.
Because of establishment in 1970s, lake of numbers of data types, Modbus have limitations of large binary object support. There is also not flexibility to find description of slave nodes on the field. Since Modbus is a master/slave protocol, there is no way for the field device to report any exception during communication, here master poll s all the slaves to look changes in data. This consumes bandwidth and network time in applications where bandwidth may be expensive over lower bit rate. This protocol provides no security against unauthorized queries (commands) or interception of data.
Profibus automation technology provides the requirements for the communication between intelligent devices and their widely use in many other field equipments of automation. It permits full use of the advantages of digital communication such as improved resolution, remote diagnostics and setup or configuration. With the provisions of more security with the help of GSD (GerteStammaDaten: equipment master data) file, Profibus is more convenient than Modbus.[2]
Implementation of Profibus protocol for the field communication instead of Modbus protocol over mINT IO can reach intelligent IO feature to upper level. Thus, to reduce further expenses, conversion device from Modbus to ProfibusDP can be described in further topics. This conversing device works as a Modbus master and a Profibus slave.

1.1 mINT I/O

Figure 1.1: mINT I/O [3]

mINT is a product of masibus which works intelligently to control I/Os mINT IO modules are innovative which provides a simple low cost solution for distributed I/O requirements. The IO system consists of stand-alone Digital and Analog - Input/output modules which are connected together on a RS485 two wire multi-drop network. The modules communicate using the MODBUS RTU protocol. A 16-Bit controller is used in the modules to provide high speed data processing and fast communications turnaround times. Multiple baud rates are selectable from 9600 to 115200 baud.

Four kind of different I/Os are of mINT product

mINT DI-16 The IO-16DI module is a 16 channel digital input module. The inputs are isolated from the logic by bi-directional Opto-couplers. The common is connected internally to either the + volts or - volts. The inputs have internal counters associated with them. These counters are 32 bit Counters allowing a count value from 0 to 4294967295. The count value can be cleared by writing a zero to the associated registers or preset to any other value using the same method.

mINT DO-16 This module has 16 open collector (NPN) digital outputs. The outputs may be used to drive Lamps or external relays when more drive capability is required. The outputs are isolated from the logic and they share a common

negative terminal. We can configure this module with mINT Plus configuration software or any Modbus Master device. This module has three output configuration modes such as normal output, single pulse and continuous pulse output mode. Pulse frequency and high low time is adjustable.

mINT AI-08 This module is supplied with Thermocouple, RTD, Voltage and current Inputs. All 8 channels are supplied with user selectable universal Inputs. We can configure this module with software. All user Zero values and Span values of connected sensors are configurable through the mINT Plus Software.

mINT AO-08 This module is supplied with 04mA - 20mA or 02-10Vdc analog output. All 8 channels are supplied with Fixed type of analog Outputs either current or voltage. We can configure this module with mINT Plus configuration software or any Modbus Master device. All user Zero values and Span values of connected load are configurable through the mINT Plus Software.

1.2 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, *Literature survey*, describes the details of field communication protocols,Modbus-RTU and Profibus-DP

Chapter 3, *Hardware Block diagram*, describes blocks communication, functionality and requirement of each block component.

Chapter 4, *Software Functional Blockdiagram*, describes various functions of the firmware flow.

Chapter 5, *Software*, presents the software required for implementing and testing protocol and communication.

In **chapter 6**, *Result*, result of communication implememtationusing modbus protocol as well as Profibus rotocol.

Finally, in **chapter 7** concluding remarks and scope for future work is presented.

Chapter 2

Profibus-DP protocol and Modbus RTU protocol

2.1 Modbus RTU Protocol

Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become a de facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices. Modbus enables communication between many (approximately 240) devices connected to the same network, for example a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from its use in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact. MODBUS is an application layer messaging protocol for client/server communication between devices connected on different types of buses or networks.

At the message level, the Modbus protocol still applies the masterslave principle even

though the network communication method is peertopeer. If a controller originates a message, it does so as a master device, and expects a response from slave device. Similarly, when a controller receives a message it constructs a slave response and returns it to the originating controller.[4]

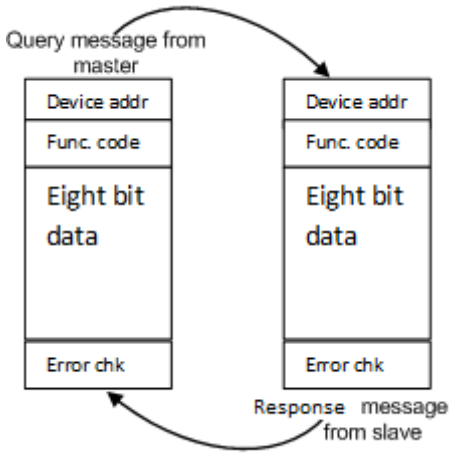

Figure 2.1: Mater Slave query Response[5]

the query : The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, function code 03 will query the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

The Response: If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error. The error check field allows the master to confirm that the message contents are valid.

2.1.1 Frame format

The format for each byte in RTU mode is:

Coding System 8bit binary, hexadecimal 09, AF
Two hexadecimal characters contained in each
8bit field of the message

Bits per Byte 1 start bit
8 data bits, least significant bit sent first
1 bit for even/odd parity; no bit for no parity
1 stop bit if parity is used; 2 bits if no parity

Error Check Field Cyclical Redundancy Check (CRC)

In RTU mode, messages start with a silent interval of at least 3.5 character times. This is most easily implemented as a multiple of character times at the baud rate that is being used on the network (shown as T1-T2-T3-T4 in the figure below). The first field then transmitted is the device address.
The allowable characters transmitted for all fields are hexadecimal 0-9, A-F. Networked devices monitor the network bus continuously, including during the silent intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device.
Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval.
The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.
Similarly, if a new message begins earlier than 3.5 character times following a previous

message, the receiving device will consider it a continuation of the previous message. This will set an error, as the value in the final CRC field will not be valid for the combined messages. A typical message frame is shown below.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4	8-BITS	8-BITS	N X 8 BITS	16 BITS	T1-T2-T3-T4

Figure 2.2: Modbus frame

Address field: The address field of a message frame contains eight bits (RTU). Valid slave device addresses are in the range of 0 - 247 decimal. The individual slave devices are assigned addresses in the range of 1-247. A master addresses a slave by placing the slave address in the address field of the message. When the slave sends its response, it places its own address in this address field of the response to let the master know which slave is responding.

Function field: The function code field of a message frame contains two characters (ASCII) or eight bits (RTU). Valid codes are in the range of 1 255 decimal. Of these, some codes are applicable to all Modicon controllers, while some codes apply only to certain models, and others are reserved for future. Function codes are as under.
01 Read Coil Status
02 Read Input Status
03 Read Holding Registers
04 Read Input Registers
05 Force Single Coil
06 Preset Single Register
15 (0F Hex) Force Multiple Coils
16 (10 Hex) Preset Multiple Registers

Data field: The data field is constructed using sets of two hexadecimal digits, in the range of 00 to FF hexadecimal. These can be made from a pair of ASCII characters, or from one RTU character, according to the networks serial transmission mode. The data field of messages sent from a master to slave devices contains additional information which the slave must use to take the action defined by the function code. This can include items like discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field.

The error checking field: When RTU mode is used for character framing, the error checking field contains a 16bit value implemented as two 8bit bytes. The error check value is the result of a Cyclical Redundancy Check calculation performed on the message contents. The CRC field is appended to the message as the last field in the message. When this is done, the loworder byte of the field is appended first, followed by the highorder byte. The CRC highorder byte is the last byte to be sent in the message.

2.1.2 Modbus data model

MODBUS bases its data model on a series of tables that have distinguishing characteristics. The four primary tables are:

The distinctions between inputs and outputs, and between bit-addressable and word-addressable data items, do not imply any application behaviour. It is perfectly acceptable, and very common, to regard all four tables as overlaying one another, if this is the most natural interpretation on the target machine in question.
For each of the primary tables, the protocol allows individual selection of 65536 data items, and the operations of read or write of those items are designed to span multiple consecutive data items up to a data size limit which is dependent on the transaction function code.
Its obvious that all the data handled via MODBUS (bits, registers) must be located in

Table 2.1: Modbus data model

Primary tables	Object type	Type of	Comments
Discrete input	Single bit	Read only	This type of data can be provided by an i/o system
Coils	Single bit	Read/Write	This type of data can be alterable by an application program
Input register	16- bit word	Read only	This type of data can be provided by an i/o system
Holding register	16- bit word	Read/Write	This type of data can be alterable by an application program

device application memory. But physical address in memory should not be confused with data reference. The only requirement is to link data reference with physical address. MODBUS logical reference numbers, which are used in MODBUS functions, are unsigned integer indices starting at zero.[5]

2.2 Profibus Protocol

Profibus is the German national standards DlNl9245 (in 1991), and European standard EN50170's (in 1996) field bus and International Electrotechnical Committee IEC 61158 Parts I, II and III (1999). A " qualified certification system " ensures that the quality assurances of Profibus products are totally independent of any manufacturer. It is one of the most popular field bus technologies. Its products are widely used in industry, electricity, energy, transportation, and other automated fields. According to the characteristics of its applications, Profibus is divided into three types: Profibus-DP, Profibus-FMS and Profibus-PA. Profibus-DP is suitable for high-speed communication between decentralized peripherals, which is more and more popular between automation devices and smart instrumentation industries.[6]

2.2.1 Structure of Profibus protocol

Profibus technology is mainly used in manufacture and process automation. It includes three compatible editions: Profibus-DP (Decentralized Periphery), Profibus-PA (Process Automation) and Profibus-FMS (Field Message Specification). Profibus-DP is mainly used in high-speed data communication between automatic control systems and dispersive I/Os or field intelligent devices. Profibus-DP protocol adopts the reference model of ISO/OSI; includes only the first, second layer and the user interface; it doesnt include the third to the seventh layer, for the purpose of high-speed data communication. The physical layer of Profibus-DP is accord with the EIA (Environment Impact Assessment) criterion.
Physically, Profibus is an electrical network based on a twisted pair or an optical network based on a fiber optic cable. The data link layer describes the normal formation, security mechanism and available communication services of data transmission messages. The only task of the Profibus-DP is to define how the user data transmit from one station to another through field bus, without the evaluation of the data. The direct data link map (DDLM) program supports the accessing right to the second layer. The user interface defines application functions of devices based on Profibus-DP and the characteristics of all kinds of systems or field devices. According to OSI reference model, the second layer defines the bus accessing control, the data transmission security, the management of communication protocol and messages. The second layer is called FDL (Field Data Link). The structure of the Profibus-DP protocol is shown as below (Figure:2.3).

According to Profibus DP standard, a DP system contains following stations: DP master (class 1), DP master (class 2) and DP slave. DP master (class 1) is a device that controls actual tasks such as reading and writing data from the slave and DP master(class 2) is a device which is programmed such that diagnostic unit or management device that provides diagnostic and service functions. DP slave is a device in the field area that reads in or output signals. DP slave can be modular or compact. In

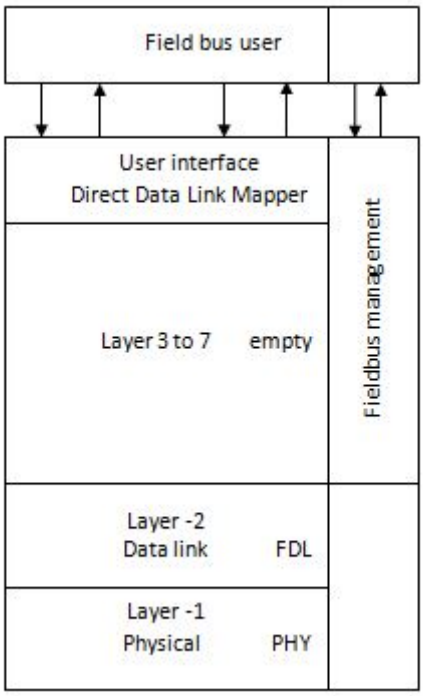

Figure 2.3: Profibus structure [6]

practical DP master (class 2) is optional.

2.2.2 Profibus Algorithm

Following state machine helps us to illustrate how Profibus reacts with respect to slave. The four main states are: power ON reset, Parameterisation, I/O Configuration, and Data exchange.

The master uses the following general telegram sequence during startup(see figure 2.4):

- Request diagnostics
- Change station address (optional)
- Parameterize the slave
- Configure the slaves

- Request diagnostics again before data exchange to ensure that system startup was OK
- Data exchange
- Global control(optional)

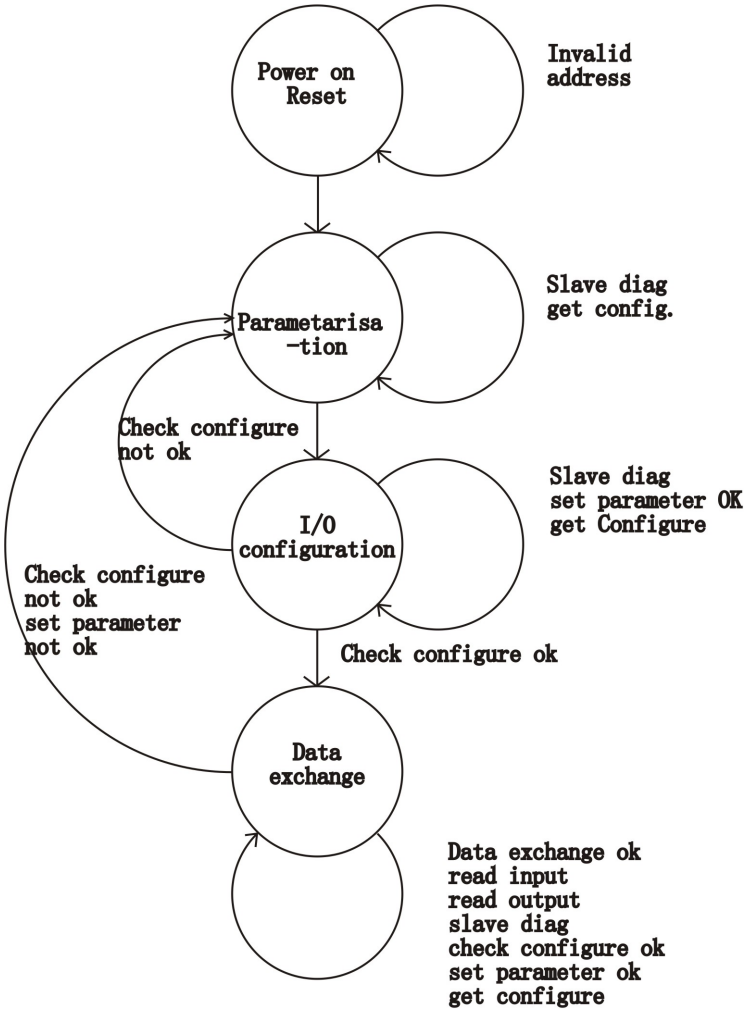

Figure 2.4: Profibus structure

Power ON/Reset State: The power on/reset state is the initial state following power up for the DP slave. In this state, the slave may receive a telegram from a class 2 master to change its station address. A slave will be held in this state if it does not have a valid address from 0-125. After completion of its power-on initialization

routine and if the slave has a valid station address, the slave will proceed to the Wait for Parameterization state.

Parameterization State: In this state, the DP slave awaits the parameterization telegram from the master which identifies the slaves master and the mode the slave is to operate in. A slave in this state will reject all other telegrams except a request telegram for diagnostics or configuration. After its parameters have been set, the slave will proceed to the I/O Configuration State. I/O Configuration State: In this state, the slave awaits a configuration telegram that specifies the number of input and output bytes that are to be exchanged in each data telegram cycle with the slave. The configuration telegram also causes the slave to check the configuration which was sent against the stored configuration. A slave in this state will accept a request telegram for diagnostics or configuration, or a set parameters telegram.

Data Exchange: State after parameterization and configuration have been accomplished, the slave cyclically exchanges I/O data with the master. This is a cyclic transfer of I/O data and possible diagnostic information. Fail Safe Operation: A Profibus master runs in two modes: Operate and Clear. With respect to the master of a Profibus DP system, the term fail-safe simply refers to whether the class 1 master sends 0 length data, or data set to 0, when it is in Clear Mode. With respect to a DP slave device, the term failsafe refers to whether the slave will process output telegrams with zero length data, or not. Whether the combined master/slave system is considered fail-safe depends on the actions taken by the slaves if the master fails, or if the master switches to Clear Mode. Ideally, the failure of a master should not cause errors in any of its slaves and the slave outputs should go to a predictable (defined) state. Using a fail-safe mode, the slave outputs can automatically switch to a fail-safe state in the event of master failure, or when the master switches to Clear Mode.

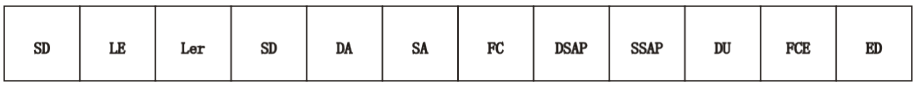

Figure 2.5: Profibus frame structure

2.2.3 Frame Structure

SD: Start Delimiter

LE: Net Data Length + DA, SA, FC, DSAP, SSAP

DA: Destination Address

SA: Source Address

FC: Function Code (FC=0A in Response Signals Diagnostic Data)

DSAP: Destination Service Access Point

SSAP: Source Service Access Point

FCS: Frame Checking Sequence

ED: End Delimiter

All of the data communication services are executed the SAPs (Service Access Points) through the upper protocol. Profibus-DP only uses the subclass of the services, namely SRD and SDN service. In Profibus-DP, every SAP is defined with a definite function. The default SAP is for data exchange. The FCs handles following tasks: (1) the transfer of process output data from a specified data area on the S7 CPU to the field device; (2) the entry of process input data read from the field device into a specified data area on the S7-CPU; (3) the handling of monitoring and diagnostic jobs. Following table shows the values of SSAP and DSAP for different functions.[7]

2.2.4 GSD file description

The Profibus equipments have the different performance characteristic, the difference of characteristic lies in the difference of the existing function (i.e. I/O signal quantity and diagnostic message) and bus parameters, such as baudrate and each kind of mon-

Table 2.2: DAP and SAP description

	DP Master		DP Slave	
Function	SSAP	SA	DSAP	DA
Data exchange	-	xx	-	Xx
Read input	62	xx	56	Xx
Read output	62	xx	57	Xx
Slave diagnostics	62	xx	60	Xx
Set parameter	62	xx	61	Xx
Check configuration	62	xx	62	Xx
Get configuration	62	xx	59	Xx
Global control	62	xx	58	Xx
Set slave address	62	xx	55	Xx

itoring time. These parameters have the difference to each kind of device type and the production manufacturer, to achieve Profibus to have the "plug and play" functionality, these parameters must be illustrated in the equipment database file, namely in the GSD document. The GSD document is to describe these parameters by one kind of accurate definition format; the production manufacturer has a GSD document to each kind of equipment. GSD file format is an accurate description of the definition, manufacturers of each device has a GSD file. In the future, if we want to set up a network, we can use the configuration software, such as COM Profibus Software as long as the equipment's GSD files are copied to the appropriate directory, we can easily put this device in the network. GSD file often includes three parts. 1) Overall explanation In this part, it mainly includes the names of vendors and equipment, hardware and software version number, supported baudrate, the possible monitoring time interval. 2) The relevant provisions of the DP main equipment It includes all parameters only suitable for the DP main equipments such as the maximum number of connected slave equipments, the load and unloading ability and so on. 3) The relevant provisions

of the DP slave equipment It includes all provisions relevant to the slave unit, such as: data and type of I/O channel, specification of diagnosis testing and compatibility information of I/O data etc. [1]

2.2.5 How profibus Does work ?

Profibus is also a master-slave type protocol like Modbus (see Figure 2.6) but with an additional token ring protocol to allow for multiple masters. Also, unlike Modbus, all devices go through a startup sequence during which they join the network. Each slave maintains a failsafe timer. If the master does not talk to it within a certain time limit, the slave goes into a safe state; the master must then go through the startup sequence again before further data exchange can occur. This, in combination with a watchdog timer in the master, ensures that all communication occurs every bus cycle with a certain time value. The general bus scan would happen as shown in Figure 7. Master A receives the token, which gives it control of the bus. It will then exchange data with each of its slaves, and when complete, pass on the token to the next master (if there is one). The requirement for detailed diagnostics from each slave is also built into the protocol. During normal data exchange, a slave can alert the master that it has diagnostics, which the master will then read during the next bus scan.[8]

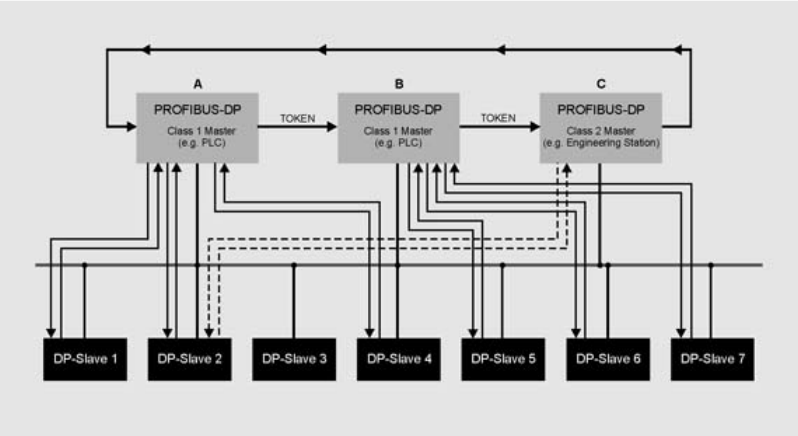

Figure 2.6: Profibus working process[8]

2.3 Modbus to Profibus

A Profibus solution permits full use of the advantages of digital communication such as improved resolution, remote diagnostics and set-up/configuration.
Limitations of Modbus

- Since Modbus was designed in the late 1970s, the number of data types is limited to those understood by PLCs at the time. Large binary objects are not supported.
- Since Modbus was designed in the late 1970s, the number of data types is limited to those understood by PLCs at the time. Large binary objects are not supported.
- Modbus transmissions must be contiguous which limits the types of remote communications devices to those that can buffer data to avoid gaps in the transmission.
- Modbus protocol provides no security against unauthorized commands or interception of data

Profibus is a very robust protocol that was designed to automate entire plants. It works extremely well in multi-vendor applications, with modems, and has detailed diagnostics. When connecting a controller to one smart device in a point-to-point configuration, or if there is only one remote site, Modbus is an easy solution. For situations where there are more points, where different vendors are involved, or where there is a hazardous environment, Profibus is a better solution.

2.4 Summary

when point to point communication is required Modbus protocol execution is simplest.for multiple masters , profibus protocol is required with more security and exceeded data rate .

Chapter 3

Hardware design

3.1 Physical structure design

Profibus physical network is shown in following figure. This shows a Modbus based devices can connect with the interface slave card and creates a combination of a profibus slave. Here single intelligent device in connected. But multiple Modbus devices can also connect. With multiple devices, memory management could be different.

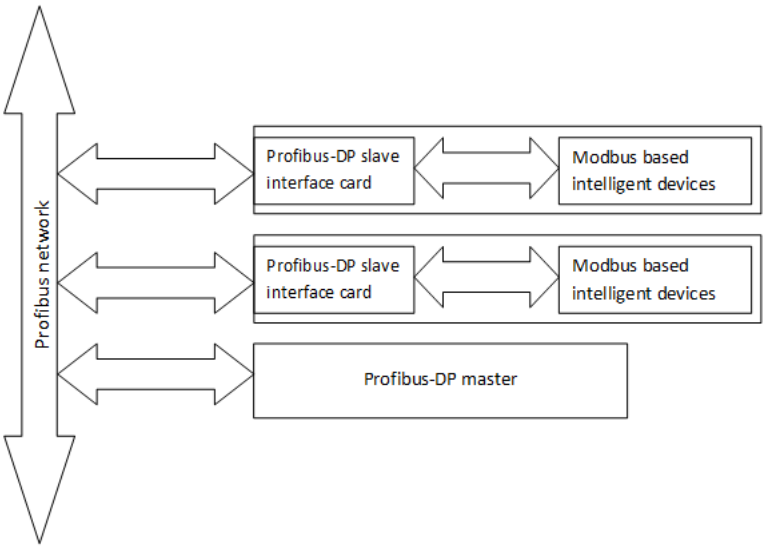

Figure 3.1: Profibus Hardware Blockdiagramn

3.2 Hardware block diagram

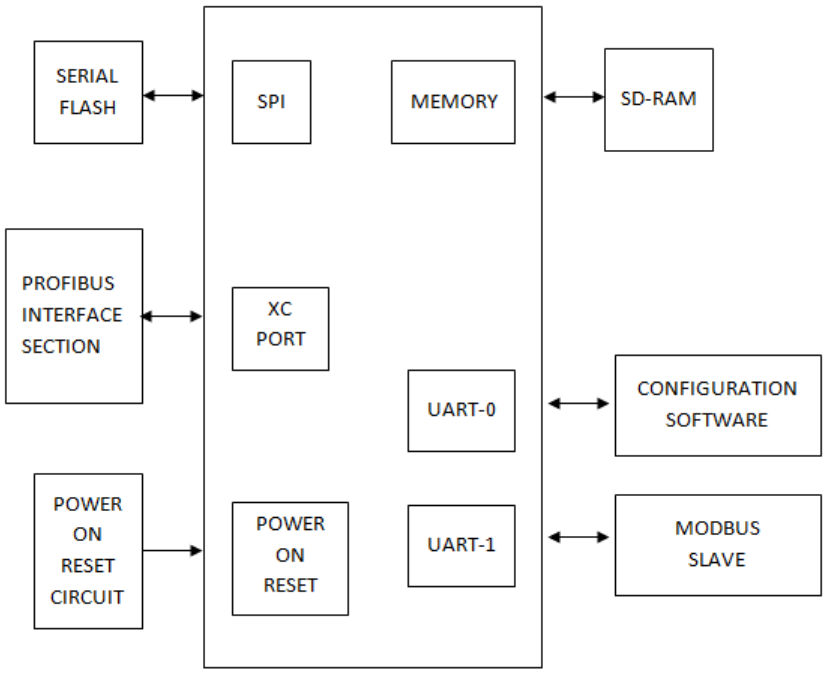

Figure 3.2: Profibus Hardware Blockdiagramn

3.2.1 netX 100 controller

Features of netX 100(next generation communication controller) controller:

1. Powerful 200 MIPS ARM 926 EJ-S CPU
2. Support for real-time Ethernet protocols etherCAT, Ethernet/IP, Powerlink, PORFINET
3. Fieldbus controller for AS interface, CAN, CC-Link, Profibus master and slave
4. 128 Kbyte internal RAM for small applications without external memory components
5. 16 Kbyte RAM with separate power supply to hold non volatile data
6. 32 Kbyte B0ot ROM
7. SDRAM controller for large memory
8. SRAM and FLASH interface without glue logic

9. Dual port memory interface for easy interface to host controller
10. Extension bus to add peripherals for standalone applications
11. USB interface, runs as host or device, eight pipes
12. 3 UARTs, 16550 compatible
13. SPI with separate input and output fifo, fully interrupt driven
14. I2C interface
15. JTAG Debug interface
16. Boundary scan
17. ARM Embedded Trace Macrocell for time tracing
18. Extended temperature range
19. Guaranteed 10 years lifetime
20. Boot option via FLASH, serial EPROM,MMC ,Dual-Port Memory or UART
21. Real time Kernel in Boot ROM
22. Software support for protocol stacks
23. Windows CE and Linux implementations available[9]

3.2.2 Modbus slave

Modbus slave block is the device such as AI AO DO and DI connected on the field working with the Modbus protocol. It gives data in the form of the Modbus data. This Modbus data are identified and are sending to the Profibus master in the form of the query response of the master. This gives input for the converter to convert Modbus protocol to the Profibus protocol. This block is work as Modbus master and it is initialise during the controller initialisation along with the Profibus slave. During this configuration required queries basic serial communication parameters are initialised.

3.2.3 Configuration Software

This block is dedicated for Profibus parameters configuration . which includes configuration of profibus slave communication baud-rate, profibus slave id, read and write data configuration.modbus queries and modbus master communication baud rate and modbus master id.This can be shown in figure 3.3.

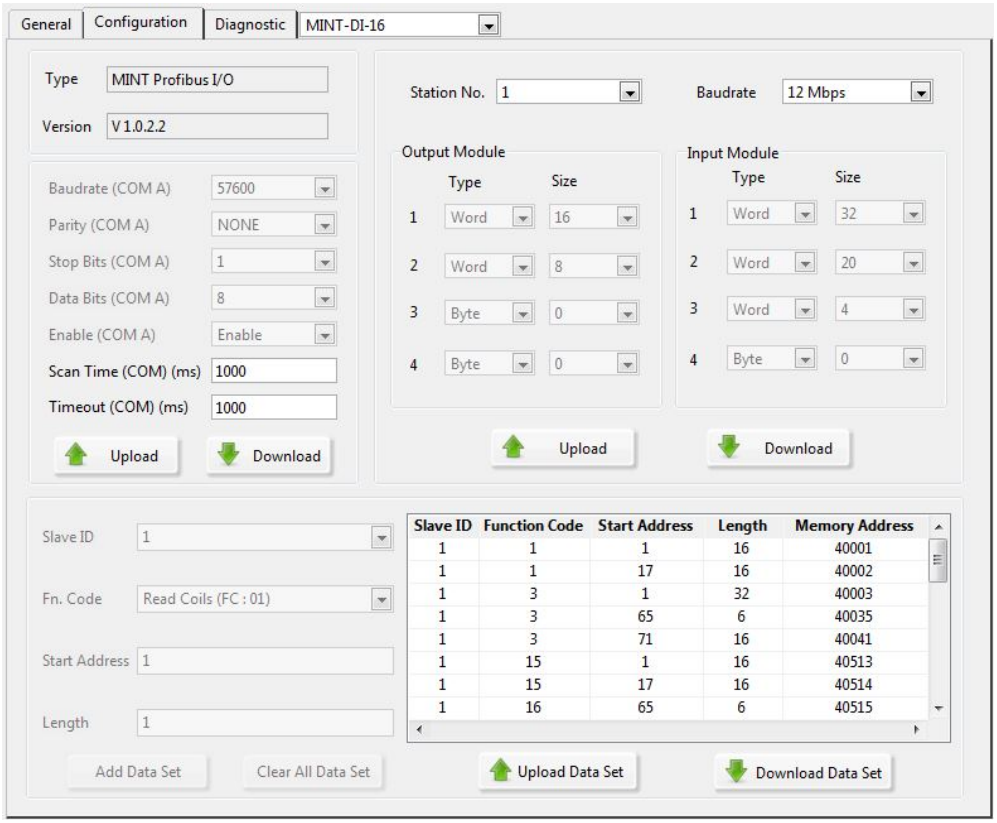

Figure 3.3: Configuration software

3.2.4 Power on reset

The netX50, netX100 and netX500 provide two inputs for reset signals, the Power On Reset (PORn) and the Reset In (RSTINn). While the use of the RSTINn is optional, the Power On Reset is mandatory. Since the PORn input is equipped with a Schmitt-trigger gate, it could basically be connected to a capacitor (other pin of cap. connected

to GND) and a pull-up resistor, however it is strongly recommended to connect this signal to the output of a reset generator with voltage supervision, to make sure, the netX will not be released from reset until the power supplies have reached sufficient and stable levels. Reset generator components are often available with either a push/pull or an open drain output. When the design will make use of the JTAG Interface of the netX, an open drain type should be selected, since this allows to simply connect the reset signal from the JTAG connector (which is also specified as open drain) to the output of the reset generator. Of course a pull-up resistor (e.g. 10 k) must be attached to the PORn signal when using open drain reset sources.
The optional RSTINn, which is commonly used by an external host processor to reset the netX, also provides a Schmitt-trigger gate. While the netX100/500 are not equipped with an internal pull-up resistor, the netX50 provides such an internal resistor (nominal 50k), hence on netX50 designs, this input could be left open when not used, however it is recommended to tie it to VDDIO (+ 3,3V), since this can improve EMC behaviour.
When placing the components during PCB design, the reset source(s) should be placed near the reset inputs of the netX, to keep the traces off the reset signals short. Routing reset signals all over the PCB may result in bad EMC behaviour of the design, since ESD may cause undesired resets of the chip.
Experience with several netX designs further has shown that a 1nF ceramic capacitor connected to GND and PORn, with the capacitor located close to the netX PORn pin, further improves resistivity against ESD.(see figure 3.4)[9]

3.2.5 Profibus interface circuit

Profibus interface is done with this block. One portion is connected with the netX controller at UART -2 which gives the output of the conversion frame of the Profibus from the Modbus data accepted from the slave device at the other UART-0 port.

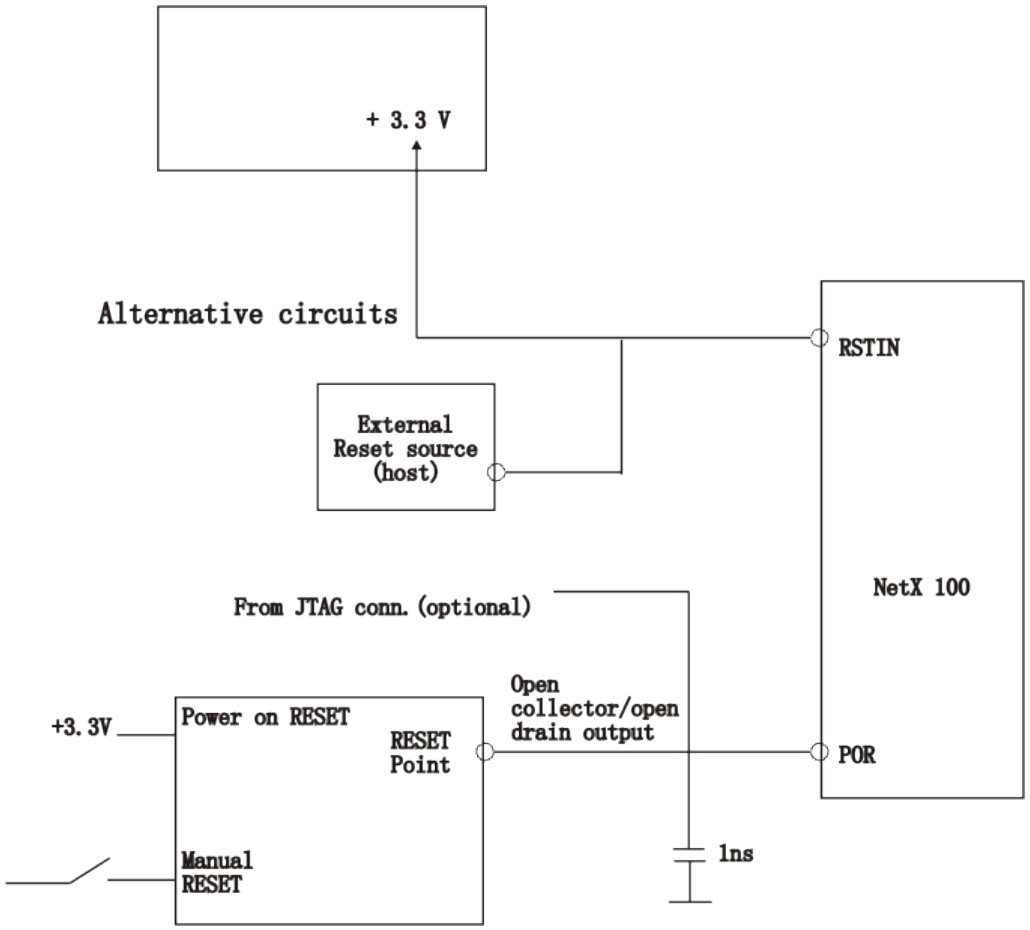

Figure 3.4: Power On Reset[9]

The other side of this circuit block db-9 connector is connected which gives the Profibus output following with the convertor of the conversion of RS 232 to Profibus. (see Figure 3.5)

3.2.6 Serial flash

To prevent lake of internal memory, the non volatile 32Mbits size of serial flash is connected. This serial flash is connected with SPI bus which is already supported with the hardware of the netX 100 controller. The SPI unit supports all four transfer modes with 16 different speeds a be configured as master and slave. As a master, it

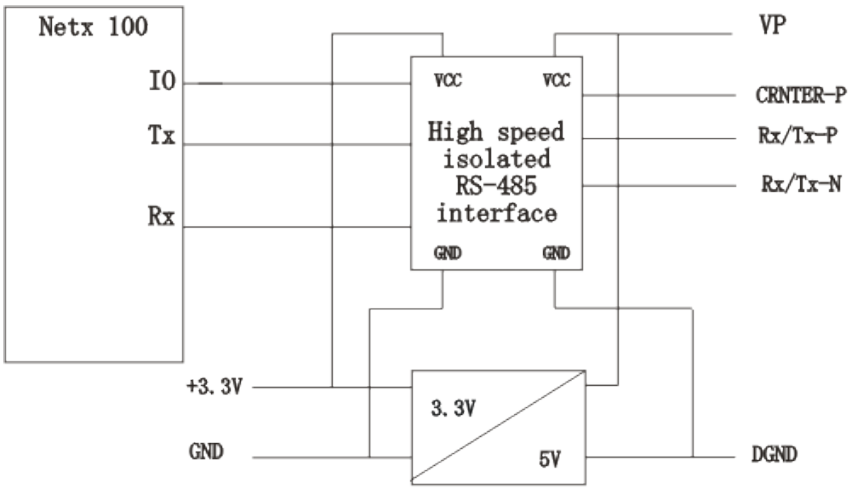

Figure 3.5: Profibus Interface Circuit

can support speed up to 25 MHz. The interface is completely interrupt driven and provides separate 16x16 bit FIFOs for incoming and outgoing data. And this can be read at any time. (see figure 3.6)[9]

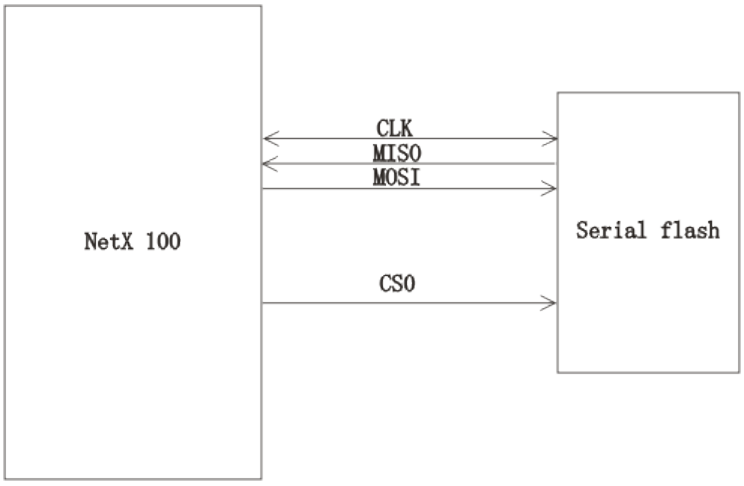

Figure 3.6: Serial flash communication[9]

3.3 Summary

Profibus to Modbus convertor module placement is between connection of profibus master and modbus slave. Data accusation Profibus slave from Modbus slave device is done with such design that , code can be load into serial flash. High speed large data communication helped by SD-RAM. Power on reset functionality provides safety to the controller.

Chapter 4

Firmware Functional details

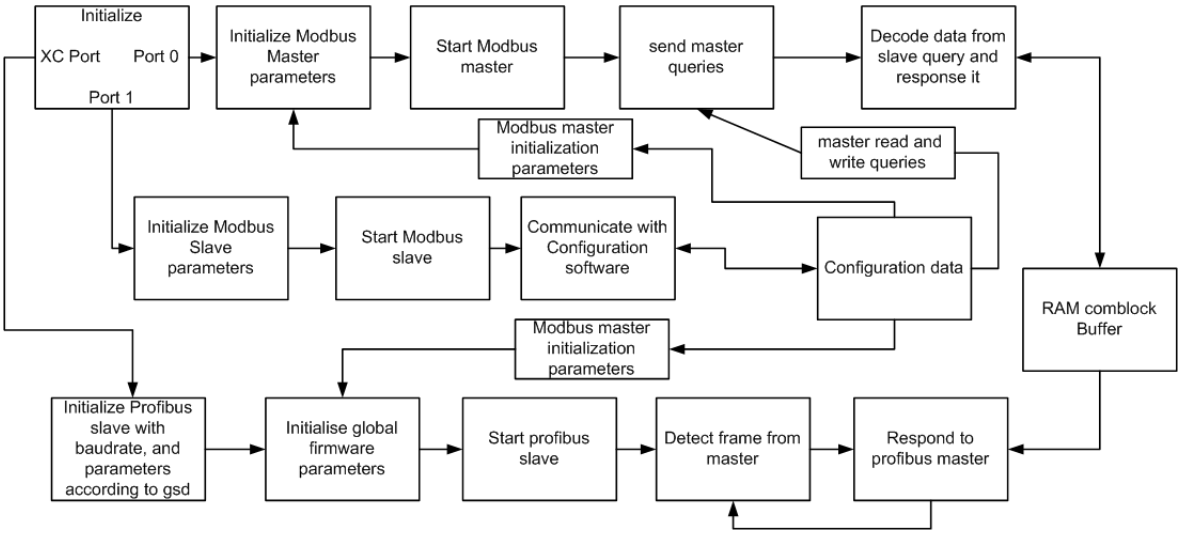

Figure 4.1: Software functional flow diagram

4.1 Blockdiagram

4.1.1 Initialize ports

Initialize all the communication uart ports 0,1, and XC Port for Modbus slave , Modbus master and Profibus slave respectively. Port 1 parameters for modbus master are vary

with the modbus slave device requirements. This can be set with configuration software mint-profibus. This block diagram is devided into 3 sections

- Modbus Slave
- Modbus Master
- Profibus Slave

4.1.2 Modbus Slave

4.1.2.1 Initialize Modbus slave parameters

Initialise Modbus slave with its slave id, and serial communication parameters such as start bit, data length, parity and stop bit. Which are 57600 baudrate, 8 bit data length, 1 stop bit and none parity.

4.1.2.2 Start Modbus save

Start Modbus slave for waiting of Modbus master query to write required parameter data.

4.1.2.3 Communication with Configuration software

Communication with configuration software gives data for Modbus master communication parameters. And Modbus queries created in configuration software. Profibus initialisation data such as input data and output data length, Profibus slave baudrate, and Profibus slave address. Received data are decoded in this block from Modbus frame and store it to RAM communication buffer block. Which initialise Profibus with their data such as Profibus station no, baudrate, no of read data size, no of write data size, module data length and data module according to GSD. And Modbus master data such as baudrate, parity, stop bit, and data length.

4.1.3 Modbus Master

4.1.3.1 Initialize Modbus master parameters

After initialise of the Modbus parameter, all the the parameters for the Modbus master required such as scan rate, master receive and transmit data length, frame receive flag, frame transmit flag and write query number flag. These data are gathered from configuration software.

4.1.3.2 Start Modbus master

Start Modbus master with sending the queries which are already predefined with the accessing data to transmit through Profibus. Here all the master queries are generated in the software Profibus configuration tool. In which we have to give data of slave address, function type, data length and starting address. All the queries are stored in flash memory consuming five bytes of data.

4.1.3.3 Receive response of slave and collect data

As a response of data received frame of the read data, data is decoded and save data to the RAM communication buffer register. All these data are used for the transmission with the Profibus slave to the master. And as a frame of the write data to the slave, data is collect from the RAM communication buffer register to send respected queries.

4.1.3.4 RAM communication Buffer block

This block contains all the data from the Profibus master to the slave, Modbus slave to the Modbus slave and of Modbus slave from the Modbus master and vice versa. Profibus master gives data to write it to the connected mINT device. And Modbus master stores all the data from the mINT analyzed from the field. Modbus slave stores all the parameter data which user want to initialize Profibus slave and Modbus slave.

4.1.4 Profibus Slave

4.1.4.1 Initialize Profibus

Initialize Profibus slave accordance with the serial communication with the parameters of baudrate, parity, start bit, stop bit and length. As per device description provided in the GSD file which is to be given to the Profibus master, all parameters such as device id (ident number), Profibus slave number, number of read data and number of write data. All these data are retrieved from the flash memory which is stored at particular address.

4.1.4.2 Initialize Profibus global firmware parameters

Parameters such as DXB (data exchange broadcast), CS (clock synchronisation),ISO (isochronous mode) , interrupts , communication blocks for buffering data, SAPs(service access point)values and xPEC (external protocol eccess control) are set to their initial values.

4.1.4.3 Start Profibus

Start Profibus protocol port with is described functionality, to avoid unwanted port block error. With the steps of confirming disability of port, clearing all pending interrupts and then start respective communication port.

4.1.4.4 Detect frame request from Profibus master

In the Profibus-DP protocol, there are many frame formats, and the length of these formats is different, which makes us hard to know which data belongs to a frame data. However, it is mentioned in the protocol that the master should wait for a moment when it sends the next frame. In other words, it is an interval between the two data frames to receive in the slave. If the interval of the frame could be identified, the frame will be known.Profibus communication is done with the process described in

figure 2.4. Depending upon this frame sections profibus slave gives its response and end of success full parametrisation and configurations, data communication starts.

4.1.4.5 Response to the master

After starting of Profibus slave, it has to response the Profibus master queries such as getting configuration, slave diagnostics, write parameters and check configuration. All the description of frame is given in literature survey chapter. And then all the data frames communication response starts asked in requests.

4.2 Communication Block

Communication interface card have a memory region as shown in figure 4.2, which stores data from profibus and Modbus communication. Profibus protocol communicates data in the individual byte format. For this data-out from profibus master stores in the data and that data is to be used for create Modbus write query. If multiple write query is to be sending to the Modbus slave, interfacing card manages proper query with proper length. And read query response data to proper profibus address. Data formation for write data queries Modbus protocol is as under.

- Function code 5: write single coil
  Single coil contains single address for one bit data in Modbus slave. Profibus data are in byte format. So assigning every coil to every byte can generate lake of bytes for other functions. So, counting each single coil write queries and modulo with count of this queries byte assignment is consider. For communication each bit change of assigned byte is observed.

- Function code 15: write multiple coil
  Similar to the single coil data, multiple coil write should contain number of register are modulo of length to 8. This value can be generate during configuration.

- Function code 6 and 16: register write

  Holding register contains two bytes of data for single address. So two byte assigned for each holding register. For multiple register same logic is applied. Here lower profibus byte is assigned for most significant byte for holding register.

This is same for read coil query. In read section, single data byte can describe minimum of 8 coil status as per the configuration and can describe one higher or lower byte of register.

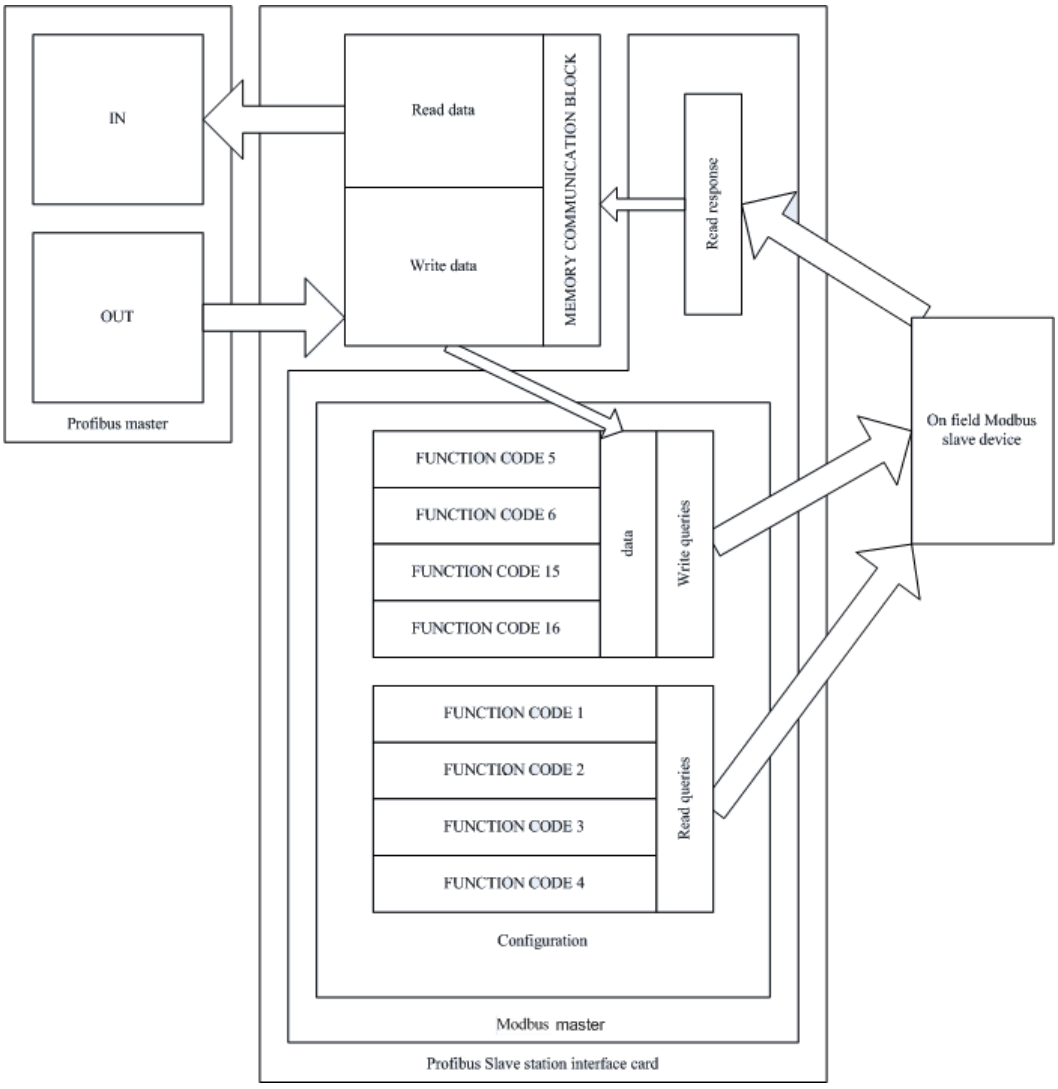

Figure 4.2: Data communication process block

4.3 Summary

Device is initialize for modbus slave, modbus master and Profibus Slave at its port1,port0 and XC port respectively. Initialisation from modbus slave from the configuration software used for modbus master and profibus slave communication parameters. Data read write gateway with the conversion from profibus data frame to modbus data frame should be done with appropriate data length.

Chapter 5

Softwares

5.1 Hitex HiTOP

We created a first project application. We will now load this into the development environment and run it on the evaluation board. The development environment is called HiTOP. HiTOP acts as an editor, make tool, project manager and debugger. The HiTOP debugger and its main windows are shown in figure 5.1

If you close a HiTOP window, it can be re-opened by moving the mouse cursor onto an unused section of the toolbar, right-clicking the mouse and then selecting the window you wish to re-open. This menu(figure 5.2) also allows you to enable and disable the various toolbars.

5.2 mINT Profibus

mINT Profibus is basically used for configure Profibus slave as well as modbus master parameters. It also used for initialising modbus master queries to access data from modbus slave.

To work with the mINT Profibus follow the steps described hereafter:

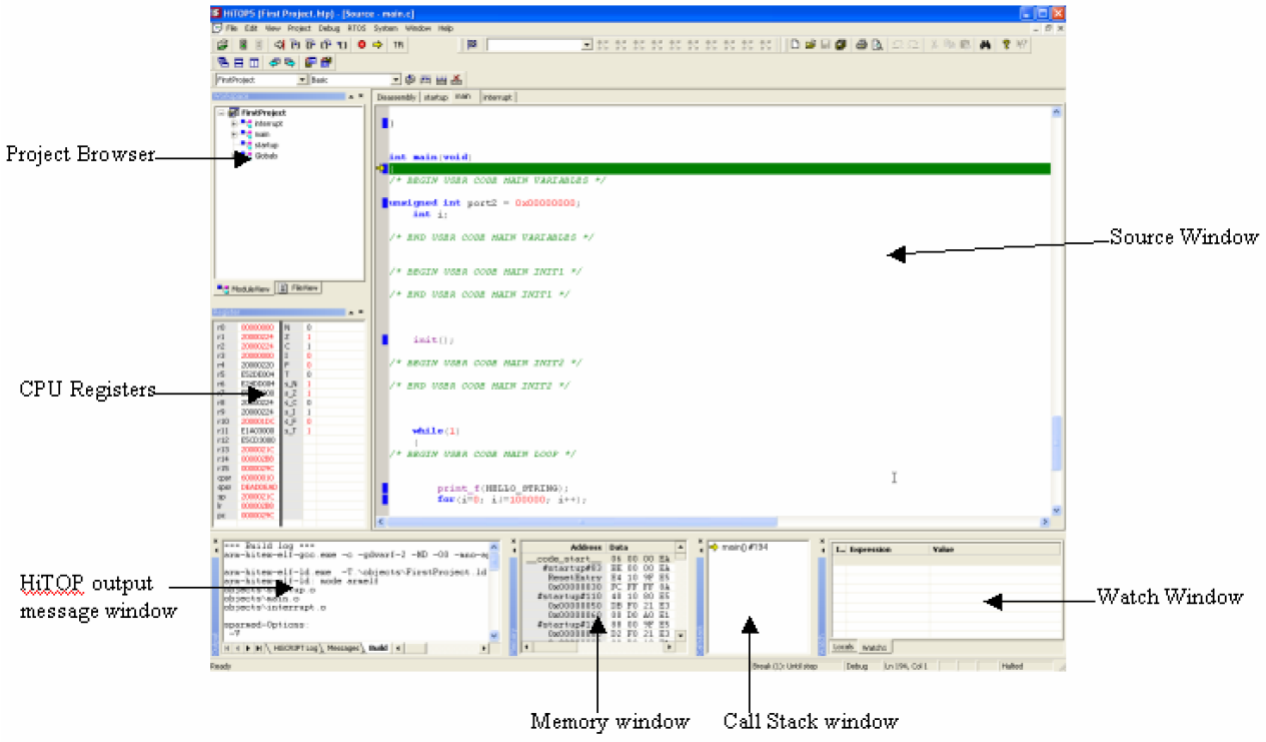

Figure 5.1: HiTOP main screen

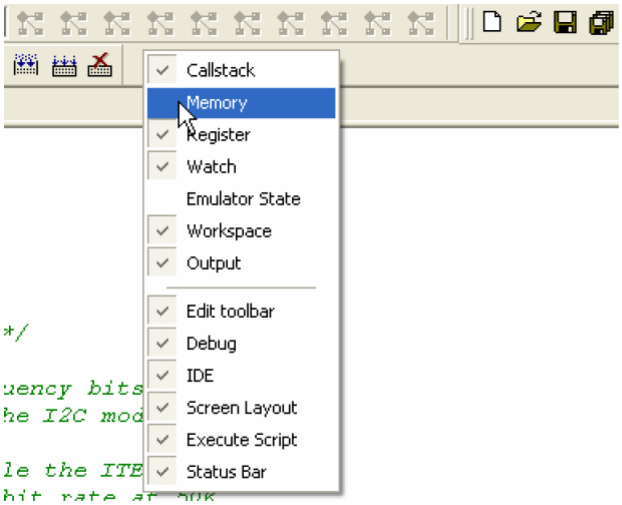

Figure 5.2: HiTOP menu screen

1. Make sure the device is correctly supplied with power and is operational.

2. Start mINT Profibus:

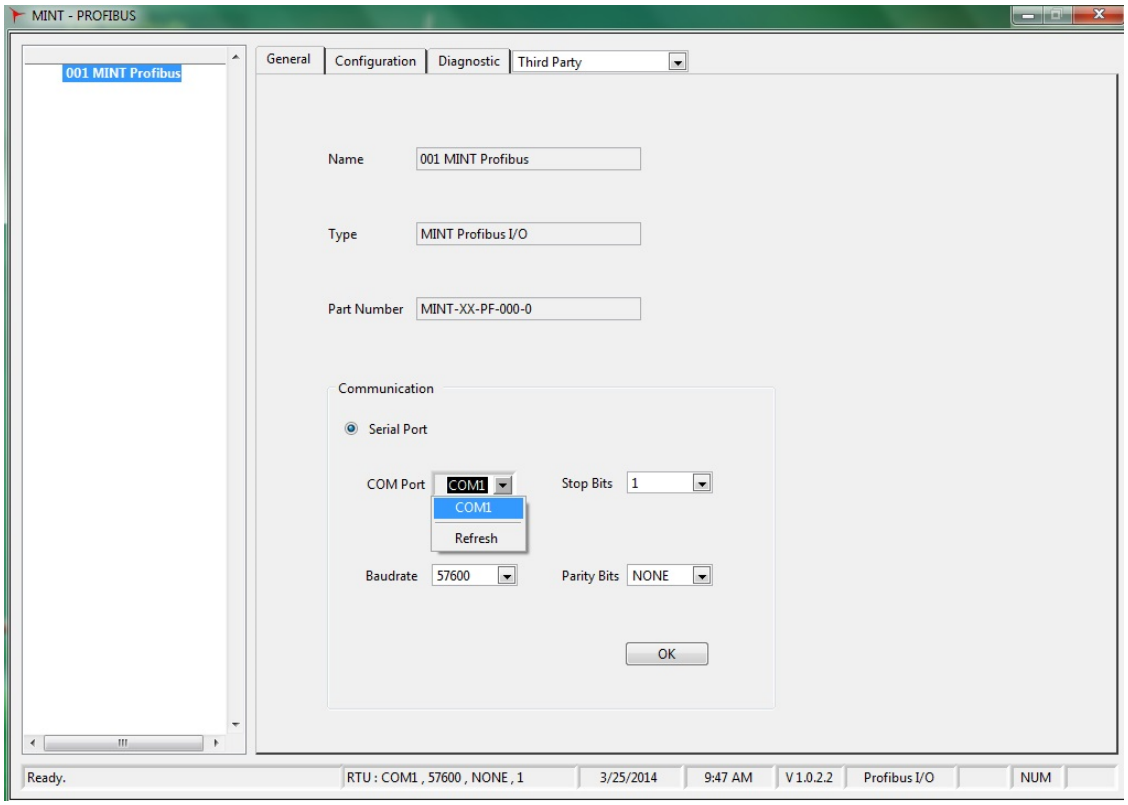

Figure 5.3: netX configuration tool

After the mINT Profibus has been started, manually select profibus module with selecting com port. Click on the OK button,from lower Right corner of the main screen from General Menu to communication with profibus module. After pressing this button,

the Device Identification shows the information about the connected device.such as device name, device type and product id. This can be elaborate in figure 5.3

3. Manually add I/O module:
   Select From the drop box, from upper Right corner of the main screen.

It will manually add the I/O module information. we can choose mint-AI , mint-AO , mint-DI , mint-DO and third party modbus RTU based devices.
Configuration is default set for selection of masibus intelligent devices.

4. Edit Third Party I/O configuration:
   User can add I/O module information by selecting the proper slave ID, Function Code, Address ,Length and click on **Add Data Set** and clear All data from selected by click on **Clear All Data Set** as shown in below figure 5.4

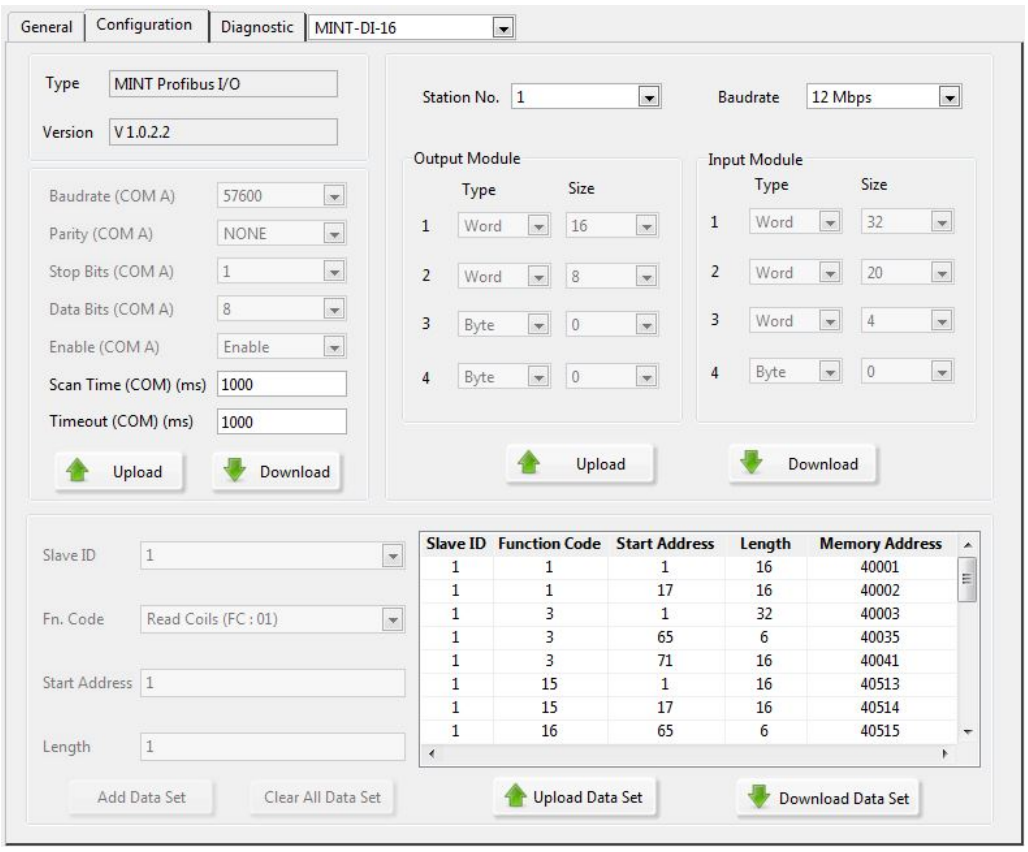

Figure 5.4: mINT profibus thirdparty configuration

5. Download and Upload data set
   Click on the Download Data Set, from lower right corner of the main screen from Configuration Menu to Download I/O Module configuration. As shown in figure 5.5 green box represents download button to configure device with the query

set. Same way Upload Data setbutton (red) brings data to the table from device where queries are already downloaded.

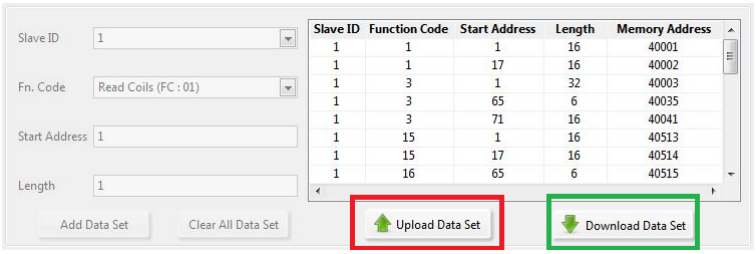

Figure 5.5: mINT profibus download and upload configuration

6. Download and Upload the Firmware and the Configuration:
   Click on the Download,(see figure 5.6) from Upper left-right corner of the main screen from Configuration Menu to Download Profibus salve configuration. similarly Click on the Upload, to bring data retrieve from device.

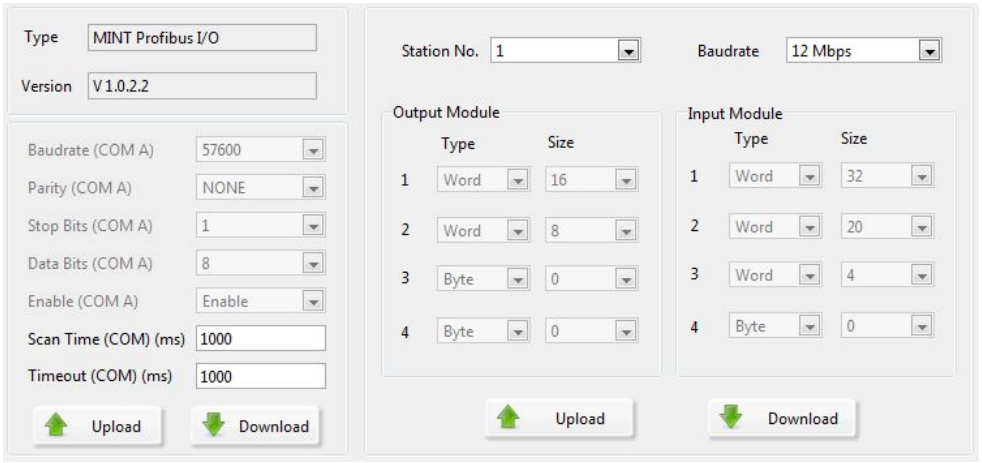

Figure 5.6: mINT profibus download and upload firmware configuration

5.3 Profisim

Profibus master simulator

This software is used for observing comunication between Profibus slave and Profibus

master as a computer device.

This software works in three phases:

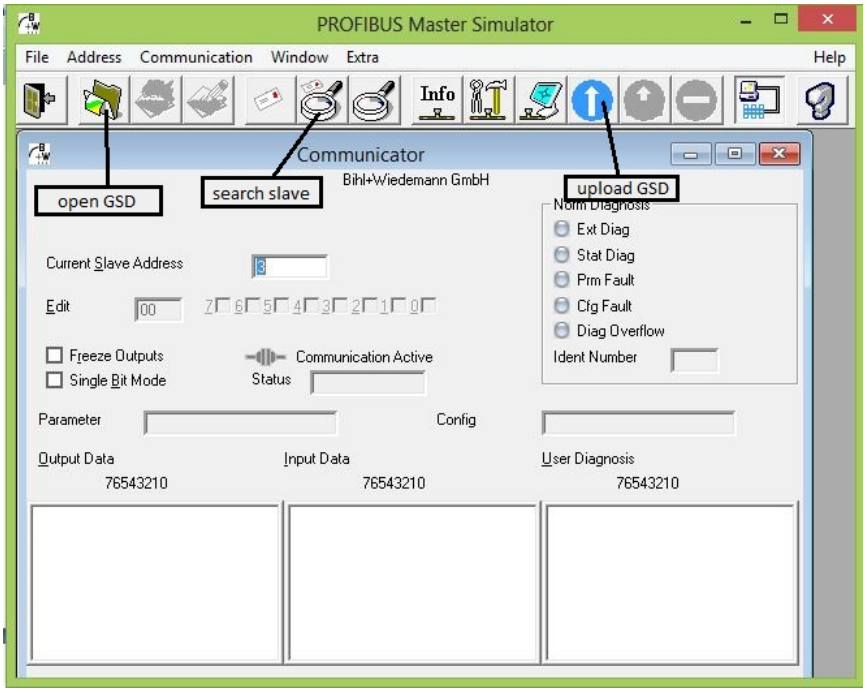

Figure 5.7: Profibus master simulator

1. Open GSD file

   Gsd file should be open for the slave configuration. File > open GSD

   this window can shown in figure 5.8

2. Search slave

   If any slave is connected with the master then it will be identified from the 128 slaves. This can be done with the steps of Address> start search Profibus slave address.(see figure 5.9)

3. Start communication

   Communication > start with GSD will communicate with the slave with the

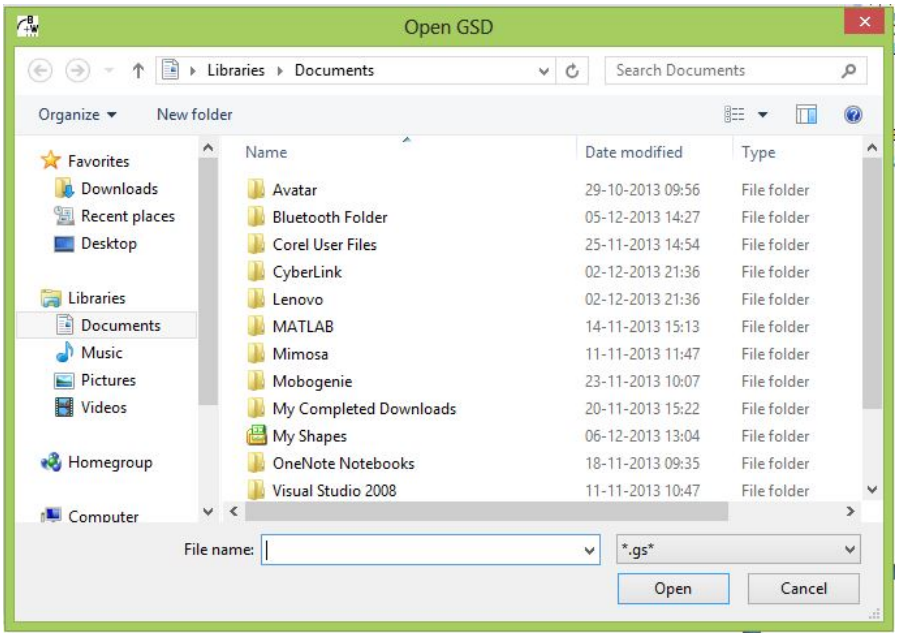

Figure 5.8: Open GSD file

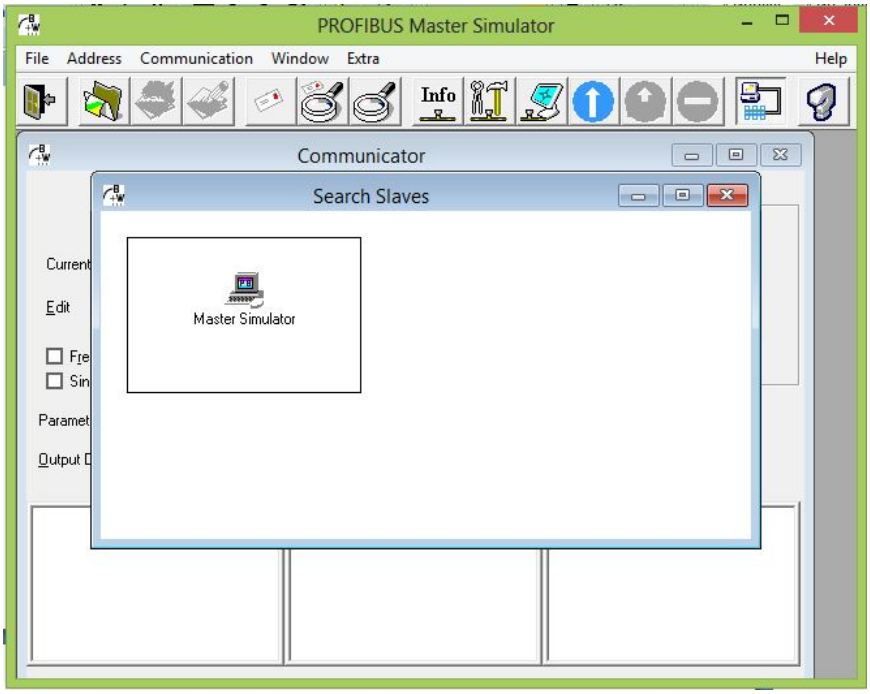

Figure 5.9: Search slave

generated queries of GSD file. And Communication > Easy start communicates without any GSD configuration (default). This communication can be shown in communicator window in fig 5.7. Read data and write data can be shown in communicator window, in output data and Input data portion.

5.4 Summary

Hitop, mINT Profibus and Profibus master simulator(profisim) softwares are used for programming, configuration and communication testing on profibus communication respectively.

Chapter 6

Result

6.1 Profibus to Modbus conversion

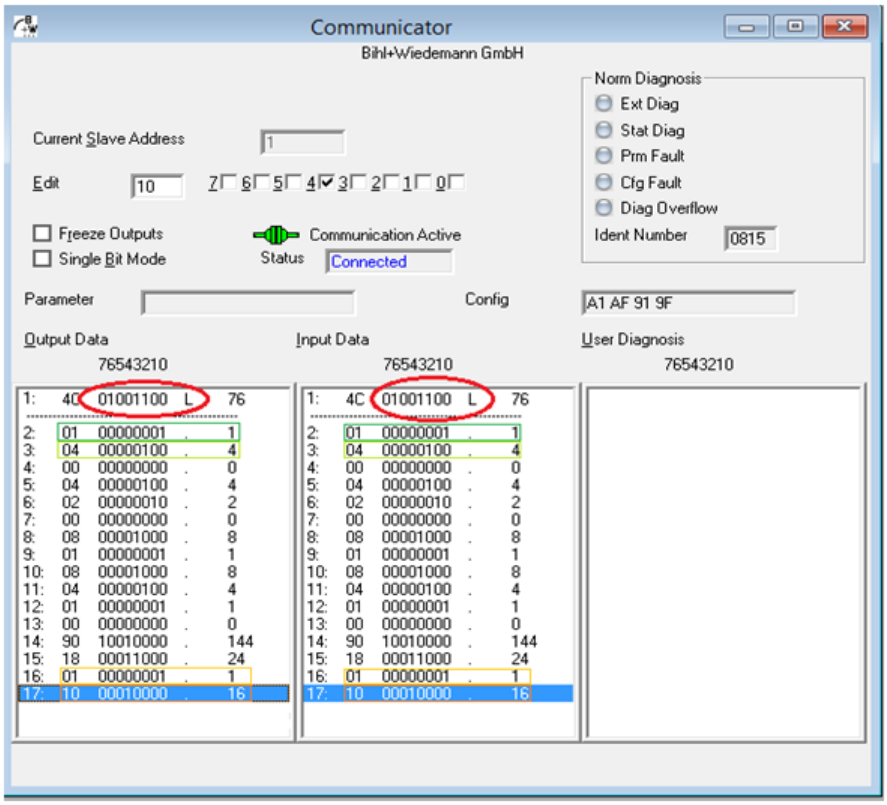

Figure 6.1: Profibus Result

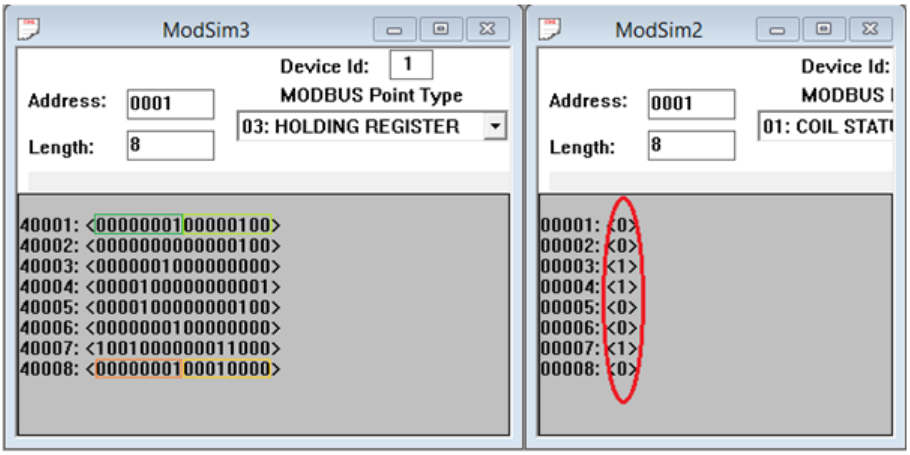

Figure 6.2: Modbus Result

Modbus queries for read and write coils, read and write holding registers are shown in following table. Configuration of profibus parameters is 1-bytes in, 1-bytes out,16-

Table 6.1: Modbus Query to send

Slave Id	Function Code	Starting Adderss	Length	Profibus Address
1	1	1	8	1(input)
1	3	1	8	2 to 17 (input)
1	15	1	8	1(output)
1	16	1	8	2 to 17(output)

bytes in and 16-bytes out, written value in output data section all 17 byte can be observe as same in input data section and Modbus slave as well. As per the table 2.2 , write data for mod bus coil in byte no 1. And same result response from the Modbus slave coil can be observed in Modsim2 window in figure 6.1 circled with red colour. Same data can seen in profibus simulator, where read data are at byte no 1. Same for holding registers which are in count of 8, so they are read and write from 2 to 17 bytes in profibus simulator (figure 6.2). Here, lower byte is higher byte for holding register. Byte 2 in profibus is higher byte of register in Modbus holding register 40001. This is shown in dark green rectangles. And byte 3 is lower byte for register 40001.

6.2 Configuration software implementation

Configuration software as modbus master is prepared in Labview software,so that configuration of mint profibus device can initialize and makes it able communicate with profibus master and modbus slave device.

Chapter 7

Conclusion and Future work

7.1 Conclusion

In this fulfilment of desetation, following conclusions can be made.
Profibus-DP protocol is implemented for the Modbus slave with the implementation of slave communication having features of frame detection, multiple baudrates support and auto baudrate detection.
Configuration Software is also implemented as Modbus master to configure mint profibus device. Profibus conversion chip is also able use into other modbus slaves other than mint products.

7.2 Future Scope

- Implementation of Ethernet communication with TCP/IP Modnet protocol in same device.
- Programming section implement with RTO rcx os. which is compatible with netX controllers.

Bibliography

[1] Panfeng Zhang,Yongzhe Shi "A Design of Slave Station Interface Circuit of Profibus-DP", "International Conference on Computer Science and Service System (CSSS)",June 2011,Volume3, paper 7,pp 1927-1930

[2] Modbus to Profibus [online],Available : *http://www.fieldbus-international.com/en/Gateway_Technology/Modbus+to+profibus.9UFRjK4R.ips* [Accessed:Aug. 2013]

[3] mINT I/O [Online],Available : *http://www.masibus.com/images/DAQ/mint-di-16.jpg* [Aug. 2013]

[4] Modbus [Online],Available: *http://en.wikipedia.org/wiki/Modbus* ,Feb. 13,2014,[Mar. 2014].

[5] Modcon Modbus Protocol Referance Guide[Online],Available: *http://www.modbus.org/docs/PI_MBUS_300.pdf* [Accessed:Mar.2014]

[6] Jun Xu and Yan-Jun Fang "Profibus Automation Technology and Its Application in DP Slave Development", "International Conference on Information Acquisition",Jun. 2004,sensor and instruments,paper 8,pp 155-159.

[7] Introduction to Profibus DP [Online],Available: *http://www.diit.unict.it/users/scava/dispense/II_270/ProfibusIntroduction.pdf*[Accessed:Aug. 2013]

[8] James Powell, Siemens, "Profibus and Modbus : comparision ",Oct. 2013,[Online], Available: *http://www.automation.com/automation-news/article/profibus-and-modbus-a-comparison*,[Accessed :March 2014].

[9] Technical Data Referance Guide # netx500 100, eddition 1.3 ,2008/07,[online],Available: *http://hilscher.com/files_design/8/netX500-100_Technical_Data_Reference_Guide_13.pdf* [Accessed:Aug 2013].