

**UNIVERSITY OF GAZIANTEP
GRADUATE SCHOOL OF
NATURAL & APPLIED SCIENCES**

**DEVELOPMENT OF EMBEDDED SYSTEM FOR MONITORING
TEMPERATURE OF BLOOD BANK**



**M.Sc. THESIS
IN
ELECTRICAL AND ELECTRONICS ENGINEERING**

BY

KAIWAN SABER ISMAEL

MAY 2016

**Development of Embedded System for Monitoring Temperature of
Blood Bank**

M.Sc. Thesis

in

Electrical and Electronics Engineering

University of Gaziantep

Supervisor

Prof. Dr. Ergun ERÇELEBI

By

Kaiwan Saber ISMAEL

May 2016



© 2016 [Kaiwan Saber ISMAEL]

REPUBLIC OF TURKEY
UNIVERSITY OF GAZIANTEP
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES
ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT


Name of the thesis: Development of Embedded System for Monitoring Temperature of
Blood Bank

Name of the student: Kaiwan Saber ISMAEL
Exam date: May 16, 2016

Approval of the Graduate School of Natural and Applied Sciences


Prof. Dr. Metin BEDİR
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of
Science.


Prof. Dr. Ergun ERÇELEBİ
Head of Department

This is to certify that we have read this thesis and that in our consensus/majority opinion it
is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.


Prof. Dr. Ergun ERÇELEBİ
Supervisor

Examining Committee Members:

Signature

Prof. Dr. Ergun Erçelebi



Assoc. Prof. Dr. Ahmet Alkan



Assist. Prof. Dr. Mete Vural





I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Kaiwan Saber ISMAEL

ABSTRACT

DEVELOPMENT OF EMBEDDED SYSTEM FOR MONITORING TEMPERATURE OF BLOOD BANK

ISMAEL, Kaiwan Saber
M.Sc. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ergun ERÇELEBI
May 2016
Pages 99

The human blood is analyzed for three main components as plasma, platelets and red blood cell. Those components of blood must be safekeeping inside refrigerators. There are many refrigerators inside blood bank center. In this thesis, we have developed a embedded hardware and software for monitoring of temperature of 24 refrigerators inside blood bank center. Embedded hardware is implemented with microcontroller, oscillator circuit, CAN integrated circuit for CAN bus, and LCD. Due to the security of locations in the blood bank hall and difficulty of monitoring of each refrigerator separately, this work proposes a solution to monitor temperatures of all the blood bank refrigerators in one location. CAN-bus system has been used because it has many advantages. Especially it has been preferred due to easy in use, low cost, providing a reduction in wiring, fast to repair and easily expanding. PIC18F458 microcontroller has been used in embedded hardware design to execute embedded software. As a temperature sensor, LM35 integrated circuit was utilized. Embedded software's have been developed by using C language on MPLAB IDE. The designed system has ability of detecting temperature in the range between -55°C and $+150^{\circ}\text{C}$. The temperature sensed by sensor is compared with the setting value by the user and if the temperature goes beyond the preset temperature, then buzzer and a red LED will be active. Also, the address of each refrigerator is sent to LCD display so operator knows which refrigerator fails.

Keyword: Monitoring blood bank center, Control Area Network (CAN), PIC microcontroller, MPLAB IDE.

ÖZET

KAN BANKASININ SICAKLIĞININ İZLENMESİ İÇİN GÖMÜLÜ SİSTEMİN GELİŞTİRİLMESİ

ISMAEL, Kaiwan Saber

Yüksek lisans tezi: Elektrik-Elektronik Mühendisliği

Tez Yöneticisi: Prof. Dr. Ergun Erçelebi

MAY 2016

99 sayfa

İnsan kanı üç ana bölümden oluşan plazma, trombositler ve kırmızı kan hücresi için analiz edilir. Kanın bu bileşenleri buzdolabı içerisinde saklanmalıdır. Kan bankası merkezi içinde birçok buzdolapları vardır. Bu tezde, kan bankası merkezinde bulunan 24 buzdolabının sıcaklıklarının gözlemlenmesi için gömülü donanım ve yazılım geliştirdik. Gömülü donanım mikro denetleyici, osilatör devresi, CAN veri yolu için CAN tümleşik devresi ve LCD ile gerçekleştirildi. Kan bankası koridorlarındaki yerlerin güvenliği ve her buzdolabının ayrı ayrı izlenmesinin zorluğu nedeniyle bu çalışma tek yerden tüm buzdolapların sıcaklıklarının gözlenmesi için çözüm sunmaktadır. CAN veri yolu bir çok avantajından dolayı kullanıldı. Özellikle kolay kullanımı, düşük fiyatı, kablo gereksinimini azaltması, hızlı tamiri ve kolay genişletilir olması nedenleriyle bu çalışmada tercih edildi. Gömülü yazılımı koşturmak için gömülü donanım tasarımında PIC18F458 mikro denetleyicisi kullanıldı. Sıcaklık algılayıcısı olarak LM35 tümleşik devresi kullanıldı. Gömülü yazılımlar MPLAB platformunda C yazılım dili kullanılarak geliştirildi. Tasarlanan sistem -55°C ve $+150^{\circ}\text{C}$ arasındaki sıcaklıkları algılayabilecek kabiliyete sahiptir. Algılayıcı tarafından algılanan sıcaklık değeri belirlenen sıcaklık değeri ile kıyaslanır ve şayet bu sıcaklık değeri belirlenen sıcaklık değerinin ötesine geçerse LED ve sesli ikaz aktif hale geçecektir. İlaveten, operatör hangi buzdolabının hatalı olduğunu bilmesi için buzdolaplarının adresi LCD ekranına iletilir.

Anahtar kelimeler: gömülü donanım, gömülü yazılım, kan bankası merkezi, CAN veri yolu, mikro denetleyici.



Dedicated to

*My dears mother, father, my wife and children
my brother and my sister*

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious, the most Merciful. First of all I would like to thank to Allah for all His guidance and giving while I was preparing, doing and finishing this master thesis.

I would like to express my gratefulness to my supervisors Prof. Dr. Ergun ERÇELEBI for his guidance, patience, kindness, and encouragement throughout this research.

I would like to express my thanks to the staff members of the Department of Electrical and Electronics Engineering in the University of Gaziantep and my thanks to all other friends for their helping me in preparing this research.

Finally, my grateful thanks my parent's, wife and children for their great patience and help to accomplish this research.

TABLE OF CONTENT

	Pages
ABSTRACT	v
ÖZET	vi
ACKNOWLEDGEMENTS	viii
TABLE OF CONTENT	ix
LIST OF FIGURES	xv
LIST OF TABLES	xviii
LIST OF SYMBOLS/ABBREVIATIONS	xx
CHAPTER 1	1
INTRODUCTION	1
1.1 GENERAL INTRODUCTION.....	1
1.2 OVERVIEW OF THE CHAPTERS	2
CHAPTER 2	4
LITERATURE SURVEY	4
2.1 INTRODUCTION.....	4
2.2 MICROCONTROLLER.....	4
2.2.1 MICROPROCESSOR VERSUS MICROCONTROLLER	5

2.2.2	TYPE OF MICROCONTROLLER	6
2.2.3	PIC MICROCONTROLLER	6
2.2.3.1	PIC families.....	6
2.2.3.2	PIC18F458	7
2.2.3.2.1	Vdd (Vcc).....	8
2.2.3.2.2	VSS (GND)	8
2.2.3.2.3	Oscillator	8
2.2.3.2.4	MCLR	8
2.2.3.2.5	PIC18F CONFIG1H register.....	9
2.2.3.2.6	PIC CONFIG2L register	10
2.2.3.2.7	Watchdog Timer (WDT).....	11
2.2.4	ANALOG TO DIGITAL CONVERTER (ADC).....	11
2.2.4.1	ADC Resolution	12
2.2.4.2	Digital data output of A/D converter	13
2.2.4.3	Configuration of the A/D converters register and bit control	13
2.2.4.3.1	ADRESL and ADRESH registers.....	14
2.2.4.3.2	A/D ACQUISITION REQUIREMENTS	15
2.2.4.3.3	ADC CLOCK PERIOD	15
2.2.4.3.4	ADCON0 Control register 0	16
2.2.4.3.5	ADCON1 register.....	17

2.2.5	CLOCK OSCILLATOR	18
2.2.5.1	Internal oscillator	18
2.2.5.2	External oscillator	19
2.2.6	ECS-100AC 4MHZ(QUARTZ CRYSTAL OSCILLATOR)	22
2.2.7	LM35 TEMPERATURE SENSOR.....	22
2.2.8	CONTROLLER AREA NETWORK (CAN) BUS SYSTEM.....	23
2.2.8.1	Introduction	23
2.2.8.2	CAN protocol	24
2.2.8.3	Layers of CAN	25
2.2.8.4	CAN Message Frames.....	25
2.2.8.4.1	Data frame	25
2.2.8.5	The CAN bus	27
2.2.9	LIQUID CRYSTAL DISPLAY (LCD)	29
2.2.9.1	LCD Display pins.....	30
2.2.9.2	LCD screen.....	32
2.2.9.3	LCD memory	32
2.2.9.3.1	DDRAM (Display Data RAM) memory.....	32
2.2.9.3.2	CGROM	33
2.2.9.3.3	CGRAM	33
2.2.9.4	LCD Basic Commands.....	34
2.2.9.5	LCD Connecting	35

2.2.10 LED DIODE	35
2.2.11 MCP2561 CONTROLLER AREA NETWORK (CAN) TRANSCEIVER	36
2.2.12 LM7805C VOLTAGE REGULATOR	37
2.2.13 ULN2003APG SEVEN CHANNEL DARLINGTON SINK DRIVER	38
2.2.14 BUZZER	38
2.2.15 RELATED WORK	39
2.2.16 SCOPE OF THE WORK	40
CHAPTER 3	41
SOFTWARE AND CODE.....	41
3.1 INTRODUCTION.....	41
3.2 MPLAB IDE.....	41
3.3 PICKIT 3.....	42
3.4 PROGRAMMING STEPS IN MICROCONTROLLER.....	42
3.5 C PROGRAMMING VERSUS ASSEMBLY	43
3.6 PIC COMPILERS	43
3.7 DOWNLOADING MPLAB C18 C COMPILER	44
3.8 MPLAB IDE CREATE PROJECT WIZARD CHOOSING PIC18F458 WITH C LANGUAGE	45
3.9 MPLAB C18 C COMPILER LIBRARIES.....	48
3.9.1 XLCD DELAY FUNCTIONS	48
3.9.2 EXTERNAL LCD FUNCTIONS	49
4.9.3 PIC18F458 ADC C CONFIGURATION	52

CHAPTER 4	55
SYSTEM DESIGN AND IMPLEMENTATION	55
	xii
4.1 INTRODUCTION.....	55
4.2 SYSTEM DESIGN	56
4.2.1 BIT TIMING PARAMETERS.....	56
4.2.2 THE IMPLEMENTED ALGORITHMS	58
4.2.2.1 Display node	58
4.2.2.1.1 Configuration CAN Module	60
4.2.2.1.2 Implementing the Node feature.....	60
4.2.2.2 Red Blood Cell (RBC), Platelet and Plasma Node	61
4.2.2.2.1 CAN module configurations	62
4.2.2.2.2 Implementing the Node feature.....	62
4.2.2.3 Hardware structure	63
CHAPTER 5	67
RESULTS AND DISCUSION.....	67
4.1 INTRODUCTION.....	67
4.2.....	67
CHAPTER 6	71
6.1 CONCLUSION	71
6.2 SUGGESTIONS FOR FUTURE WORK	71
REFERENCES	73

APPENDIX - A	75
APPENDIX-B	85
APPENDIX - C xiii	90
APPENDIX - D	92
APPENDIX-D	94
APPENDIX-E	96
APPENDIX-F	98



LIST OF FIGURES

	Pages
Figure 2.1 Microcontroller Architecture.....	4
Figure 2.2 Microprocessor and Microcontroller block diagram.....	5
Figure 2.3 PIC18F458 chip.....	7
Figure 2.4 PIC18F458 Component Architecture.....	8
Figure 2.5 PIC18F458 Pin Diagram.....	8
Figure 2.6 PIC18F458 Basic Connection.....	9
Figure 2.7 Microcontroller connection to Sensor via ADC.....	12
Figure 2.8 PIC18 family 10-bit ADC [8].....	12
Figure 2.9 Analog to digital converters register.[8].....	14
Figure 2.10 Right justified A/D converter data storing(ADFM bit =1).....	14
Figure 2.11 Right justified A/D converter data storing (ADFM bit =0).....	15
Figure 2.12 (10-bit) ADC expense period.....	16
Figure 2.13 ADCON0 register.....	16
Figure 2.14 ADCON1 register.....	17
Figure 2.15 ADCON0 channel selection.....	18

Figure 2.16 Internal and external oscillator of microcontroller.....19

Figure 2.17 External oscillator (EC) mode.....19



Figure 2.18 Quartz crystal oscillators and block diagram connection.....	20
Figure 2.19 Ceramic oscillator.....	21
Figure 2.20 RC oscillator and connection with a microcontroller.....	21
Figure 2.21 RCIO oscillator and connection with a microcontroller.....	22
Figure 2.21 (a) EC-100AC waveform 2.21(b) Pin connections	
(c) Oscillator package	
.....	22
Figure 2.23 LM35 temperature sensor.....	23
Figure 2.24 The Layered ISO 11898 Standard Architecture.....	24
Figure 2.25 Standard CAN (11-Bit Identifier) [10].....	26
Figure 2.26 Physical Bus Connections.....	27
Figure 2.27 CAN bus speed and bus length.....	28
Figure 2.28 Twisted Pair CAN Bus and termination connection.....	28
Figure 2.29 (a) CANH and CANL bus voltage (b) CANH and CANL Signals.....	29
Figure 2.30 2×16 LCD DISPLAY.....	30
Figure 2.32 LCD pin description	31
Figure 2.33 LCD 5x8 pixel matrix	32
Figure 2.35 CGROM memory map.....	33
Figure 2.38 Lighting Emitting diode(LED).....	36

Figure 2.39 (a) MCP2561 chip (b) MCP2561 CAN Transceiver Pin Diagram	37
Figure 2.40 Four Node CAN Bus Block Diagram.....	37
Figure 2.41 Voltage Regulator Circuit	37
Figure 2.42 (a) 7-Ch Darlington Sink Driver Pin Diagram (b) 1-ch sink driver circuit (c) 7-ch Sink Driver chip	38
xvi	
Figure 3.1 MPLAB IDE	41
Figure 3.2 PICkit 3 package	42
Figure 3.3 PICkit 3 Programmer Connector piout.....	42
Figure 3.4 PIC Microcontroller programming.....	43
Figure 3.5 choosing C programming download location	44
Figure 3.6 C downloading guide.....	45
Figure 3.7 MPLAB IDE with main program	46
Figure 3.8 MPLAB IDE Creating C project.....	47
Figure 3.9 MPLAB IDE choosing specific PIC	47
Figure 4.1 Blood Bank Center designed layout.....	55
Figure 4.2 Display Node Schematic Diagram.....	59
Figure 4.3 Display Node Schematic Diagram.....	61
Figure 4.4 RBC, Platelet, and Plasma Node schematic Diagram.....	62
Figure 4.5 Flow Chart For CAN Based RBC, Platelet and Plasma Node.....	63
Figure 4.6 Four node circuit diagram of the four node monitoring system based CAN system.....	65

Figure 5.1 displays three different temperatures in three different nodes
and alarm address..... 68

Figure 5.2 The Actual hardware structure of the project system.....70



LIST OF TABLES

	Pages
Table 2.1 8-bit CONFIG1H register	9
Table 2.2 8-bit CONFIG2L register	10
Table 2.3 PIC voltage level configuration	10
Table 2.4 bit versus resolution of the ADC.....	13
Table 2.5 Configure ADC Clock period.....	15
Table 2.6 ADCON0 channel selection.....	17
Table 2.7 PIC18F458 Oscillator Frequency Choices and Capacitor Range	20
Table 3.1 LCD delay function.....	48
Table 3.2 LCD Busy function.....	49
Table 3.3 Open LCD	49
Table 3.4 put string to the LCD	49
Table 3.5 function reads the address byte from the Hitachi HD44780 LCD controller	50
Table 3.6 Reading Data from the LCD.....	50
Table 3.7 Character Generator Address function.....	51
Table 3.8 Display Data Address Function	51
Table 3.9 Writing Command to the LCD.....	51
Table 3.10 ADC Busy function	52

Table 3.11 Close the ADC function.....	52
Table 3.12 Start Conversion Function of ADC.....	53
Table 3.13 General Configuration of ADC function	53
Table 3.14 Reading Data function of ADC.....	53
Table 3.15 Channel Selection of ADC Function	54
Table 5.1 Practical Result of Implemented Board for monitoring Three Different temperatures in Three Nodes and Reaction of LED s and Buzzer.....	68

LIST OF SYMBOLS/ABBREVIATIONS

RBC	Red blood cell
WBC	White blood cell
PLT	Platelet
PIC	Peripheral Interface Controller
LCD	Liquid Crystal Display
PC	Personal Computer
ADC	Analog to Digital Converter
EEPROM	Electrical Erasable Random Only Memory
MPLAB IDE	Microchip LAB Integrated Development Environment
Soc	Small Computer
OTP	One Time Programmable
RAM	Random Access Memory
RF	Radio Frequency
OSC	Oscillator
DLL	Data Link Layer

CHAPTER 1

INTRODUCTION

1.1 General Introduction

A blood donation center is a reserve or bank of blood or blood parts, assembled as an aftereffect of blood gift or accumulation, put away and safeguarded for later use in blood transfusion. The expression "blood donation center" normally alludes to a division of a clinic where the capacity of blood item happens and where appropriate testing is performed (to decrease the danger of transfusion-related antagonistic occasions). Notwithstanding, it some of the time alludes to a gathering focus, and for sure a few clinics additionally perform accumulation [1].

Blood is an exceptionally concentrated tissue made out of more than 4,000 various types of parts. Four of the most critical ones are red cells(RBC), white cells (WBC), platelets(PLT), and plasma.

Every blood segment is utilized for an alternate sign; along these lines, the part partition has augmented the utility of one entire blood unit. Distinctive segments need diverse capacity conditions and temperature prerequisites for remedial viability. An assortment of types of gear to keep up suitable surrounding conditions amid capacity and transportation are in vogue [2].

With the progress of development, the techniques are ending up being progressive Complex. As a result of this development in capriciousness, for beneficial examination of strategy, the amount of parameters required for data acquiring also increases. Data Acquisition is basically the social event of information around a structure or methodology.

It is the procedure of gathering information in a computerized design from analog and digital measurement sources, for example, sensors and gadgets under test. Prior

to the PC age, most information was recorded physically or on strip graph recorders. Numerous new era information securing items has been created because of the rise of the microcontroller that empowers ongoing social ongoing, investigation.

Temperature is one of the fundamental parameters to control in the greater part of the manufacturing industry, food processing, pharmaceutical and so forth. In these sorts of industries, some item requires the obliged temperature to be kept up at most elevated need the item will come up short. So the temperature controller is most generally utilized in almost all the industries [3].

In this thesis, a monitoring of temperatures for a particular application has been planned. The framework works around the well-known PIC18 family. The framework outlines and created to quantify the temperature with the assistance of temperature sensors. A Microcontroller based temperature information has been developed for measuring the temperature at various input channels of analog to digital converter (ADC). For real time displaying, An LCD display has been used for a continuous showcase of Data procured from different sensors.

1.2 Overview of the Chapters

A brief review of the contents of this thesis is given as follows:

Chapter 2: It mentions about Microcontroller and the configuration of it, also, analog to digital converter background and using it with Microcontroller as well as mentions about CAN BUS System for communication and using it with Microcontroller, also mentions about liquid crystal display (LCD) and basic operation of it.

Chapter 3: It mentions about MPLAB IDE, using software and creating projects, also mentions about the C language, especially C library for PIC18F458, also mentions about PICKit 3 programmer tool for writing C Code to EEPROM of the PIC via MPLAB IDE.

Chapter 4: In this chapter, an account is given for the practical development of monitoring of 24 refrigerators. This process includes the design and implements CAN bus system and Microcontroller for collecting temperature data, Also, implementing the LCD (liquid crystal display) to display the temperature.

Chapter 5: In this chapter proffers the results of the LCD display and laboratory instruments, also presents the measurement and test the result of all nodes. Also, the result of monitoring of temperature obtained with the actual temperature is compared.

Chapter 6: It mentions the thesis conclusion and future work.



CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

It mentions about Microcontroller and the configuration of it, also mentions about CAN Bus System for communication and using it with Microcontroller

2.2 Microcontroller

A microcontroller is a little PC (SoC) on a solitary incorporated circuit containing a processor center, memory, and programmable input/output peripherals. Program memory as Ferroelectric RAM, NOR flash or OTP ROM is regularly included on chip and an ordinarily little measure of RAM. Microcontrollers are intended for inserting applications, as opposed to the microprocessors utilized as a part of PCs or other general purpose applications comprising of different discrete chips, as shown in figure 2.1 [4].

All devices in the PIC family come with a many kinds of development tools, are easy to obtain, remain relatively cheap, and have very good documentation [5].

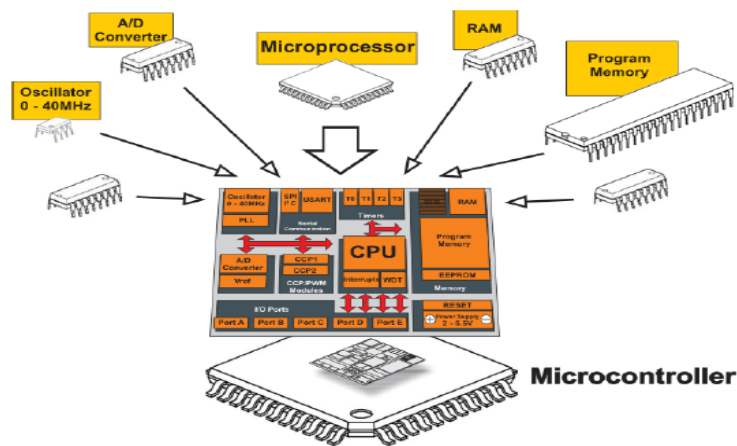


Figure 2.1 Microcontroller Architecture

2.2.1 Microprocessor versus Microcontroller

1. Speed of the Central Processing Unit (CPU): speed of the Microcontroller is lower than Microprocessor due to clock frequency. In other hand, Microcontroller has more reliable than a processor.
2. Protection: on the off chance that you programmed the microcontroller it is difficult to get the program from the Rom by different clients. The Rom will be bolted and it is difficult to recover the program from the ROM of microcontroller. Processor won't give that much security to its program.
3. The Design Time: The configuration an application microcontroller will take less time when contrasted with the processor. The interfacing between the peripherals and programming group will be simple when compared with the processor.
4. Cost: Microprocessor expensive than Microcontroller. Also, Implementing of Microprocessor costlier than Microcontroller if we compare together.
5. Applications: processors are primarily utilized as a part of calculation system, communication of the Network, and so forth Microcontrollers are principally utilized as a part of the inserted application like watches, cell phone, mp3 player, and so on.

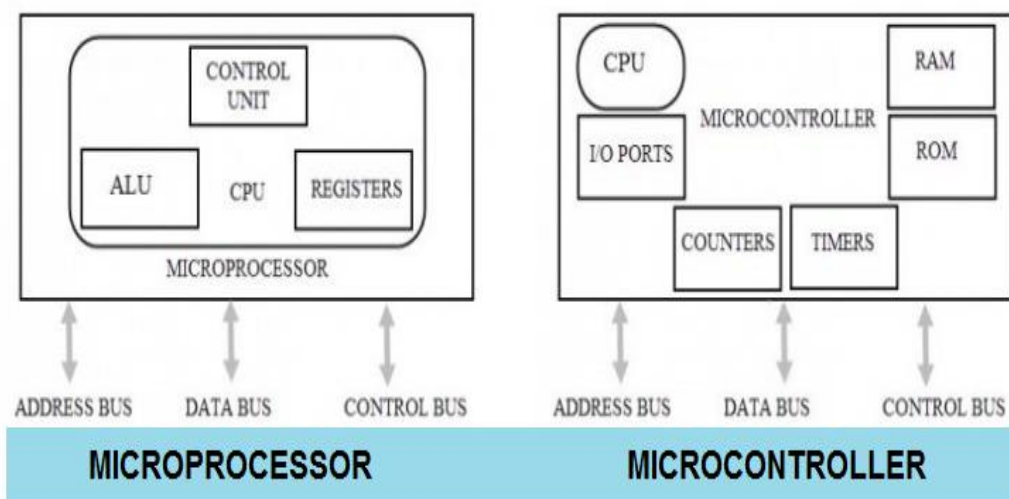


Figure 2.2 Microprocessor and Microcontroller block diagram

2.2.2 Type of Microcontroller

Starting 2008, there are a few dozen microcontroller designs and sellers including:

- ARM core processors (numerous sellers)
- Atmel AVR
- Intel 8051, additionally produced by NXP Semiconductors, Infineon and numerous others
- Infineon: XC800 for 8-bit, XE166 for 16-bit , XMC4000 for 32-bit
- MIPS
- Microchip Technology PIC, (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24), (32-bit PIC32)
- NXP Semiconductors LPC(1000, 2000, LPC3000), LPC4000 (32-bit).
- Renesas Electronics: RL78 for16-bit.
- Texas Instruments TI MSP430 for16-bit, MSP432 for 32-bit, C2000 for32-bit.
- Toshiba TLCS-870 for 8-bit and for 16-bit.

2.2.3 PIC Microcontroller

Microchip Technology is an American maker of microcontroller, memory and memory and analog semiconductors. Products of this manufacture include Microcontrollers (PICmicro (peripheral interface controller), dsPIC/PIC24, PIC32), Serial EEPROM gadgets, Serial SRAM gadgets, KEELOQ gadgets, Radio Frequency (RF) device, thermal, management of the battery or power of the analog device, and also interface and mixed signal devices. A percentage of the interface device consists USB, Controller Area Network (CAN), ZigBee/MiWi, and Ethernet [6].

2.2.3.1 PIC families

PICmicro chips are composed with a Harvard design and are offered in different device families. The pattern and mid-range families utilize 8-bit wide data memory, and the top of the line families utilize 16-bit data memory. The most recent

arrangement, PIC32MX is a 32-bit MIPS-based microcontroller. Guideline words are in sizes of 12-bit (PIC10 and PIC12), 14-bit (PIC16) and 24-bit (PIC24 and dsPIC). Representation of binary in the machine instruction is different due to change of family.

2.2.3.2 PIC18F458

PIC18F458 is a 40-pin chip created by Microchip technology; it has five ports, PORTA, PORTB, PORTC, PORTD, and PORTE. You can use all of them as input or output according to your application, it must be programmed. Also, Microcontroller operating frequency starts from a few kilohertz to 40 MHz, it additionally has Typical Data RAM has 1536 Bytes, For EEPROM it has 256 Bytes and 34 Input/output Pins, 8 Channel (10-bit) ADC (Analog to digital converter), 4 Timer. Whatever is left of the pins are assigned as Vdd, GND (Vss), OSC1(Oscillator 1 pins), OSC2(Oscillator 2 pins), and MCLR (master Clear reset), as shown in figure 2.5.



Figure 2.3 PIC18F458 chip

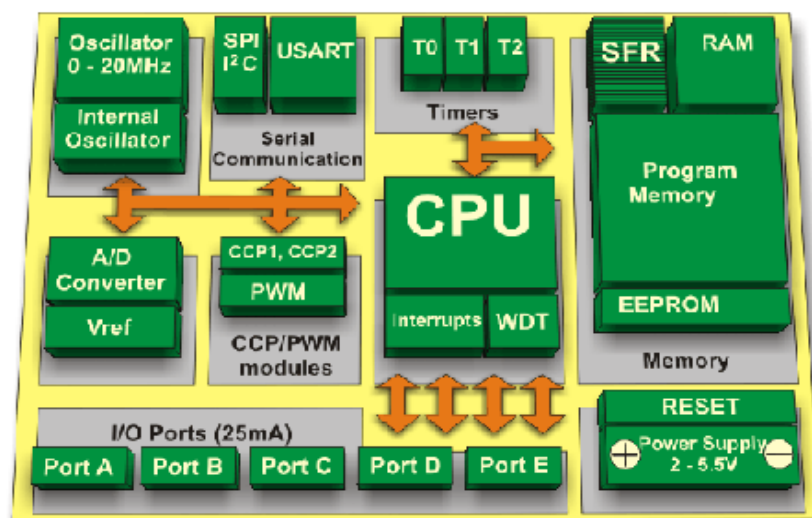


Figure 2.4 PIC18F458 Component Architecture

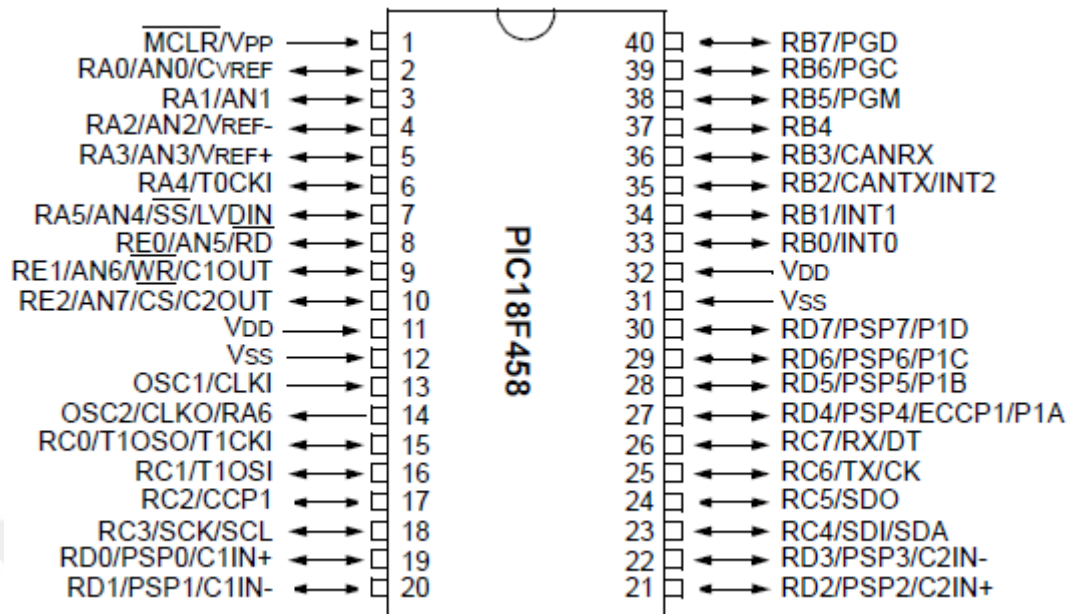


Figure 2.5 PIC18F458 Pin Diagram

2.2.3.2.1 Vdd (Vcc)

Two pins are utilized to give a voltage source to the PIC18F458. Normally the voltage source of this chip is +5V.

2.2.3.2.2 VSS (GND)

PIC18F458 has two pins for grounding. This will decrease the Noise is called ground bounce.

2.2.3.2.3 Oscillator

PIC18F458 has OSC1, OSC2 and it has more alternatives for the Clock source. A quartz Crystal oscillator is a most often case for choosing to connect the PIC. The quartz crystal oscillator connected to the pin 13 and 14 on this chip additionally need two capacitors. One of them is associated with the ground.

2.2.3.2.4 MCLR

The pin one of the PIC18F458 is called Master Clear Reset (MCLR), this pin must be high at normal operation (+ 5V across it). Also to activate the MCLR must be

applied low pulse to this pin, after doing that the activity of Microcontroller will reset. Figure 2.6 shows the basic connection of the voltage source, MCLR, and oscillator on the PIC18F458.

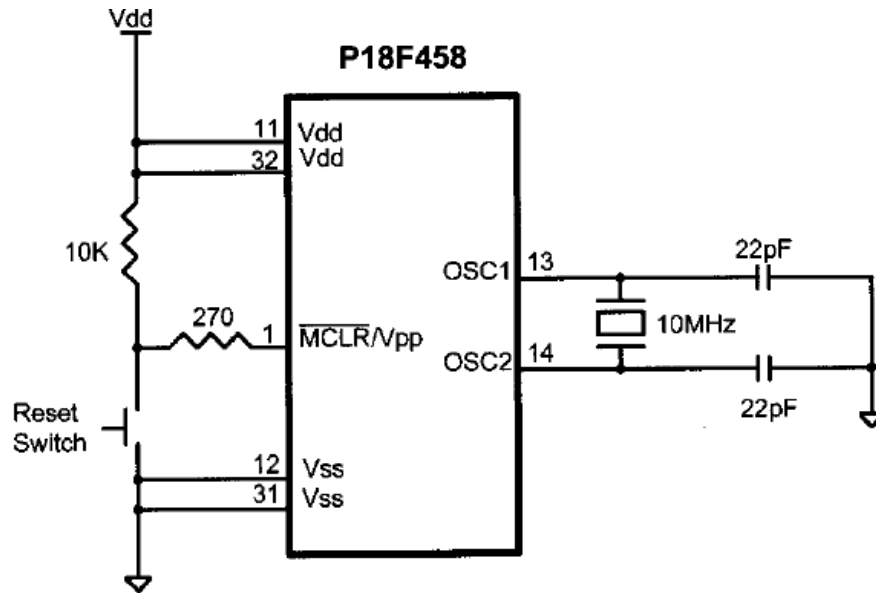


Figure 2.6 PIC18F458 Basic Connection

2.2.3.2.5 PIC18F CONFIG1H register

The CONFIG1H register in PIC18f458 is 8-bit (bit 0 –bit 7) register and it is used to configure the clock oscillator, as shown in table 2.1

Table 2.1 8-bit CONFIG1H register

U-0	U-0	OSCSSET	U-0	U-0	FOSC2	FOSC1	FOSC0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

U-0 does not used (Read as “0”), from bit (0 to 2) is used for Oscillator selection, for example: for using RC Oscillator, mean is that FOSC2: FOSC0 (111), for using HS (high-speed Oscillator), FOSC2: FOSC0(010).

Bit 5 is OSCSEN is used for enabling and disabling system Oscillator Clock Switch

OSCS = 1 is disabled.

OSCS = ON (oscillator switch is disabled). In my thesis enabled OSCS.

OSCS = 0 (oscillator switch is enabled).

OSCS = OFF (oscillator switch is enabled).

2.2.3.2.6 PIC CONFIG2L register

CONFIG12L register is used for the purpose of stability of the clock frequency and voltage during a reset. See Table 2.2. There are two internal timers inside PIC microcontroller, (OST) Oscillator start-up timer and (PWRT) Power-up timer, these two internal timers help to reduce the delay associated with the voltage source and frequency oscillator during the process of power-up.

Table2.2 8-bit CONFIG2L register

U-0	U-0	U-0	U-0	BORV1	BORV0	BOREN	PWRTEN
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Table 2.3 PIC voltage level configuration

BORV1	BORV0	Set Voltage (V)
1	1	2.0
1	0	2.7
0	1	4.2
0	0	4.5

U-0 is Unimplemented (Read as “0”)

BOREN: Brown Out Reset Enable bit

BOREN = 0 disabled

BOREN = 1 enabled

PWRT: Power –up Timer enable bit

PWRT = 1 enable

PWRT = 0 disable

In my project BOREN enabled, BORV is 2V, PWRT is disabled.

2.2.3.2.7 Watchdog Timer (WDT)

A watchdog timer is a clock associated with a totally separate RC oscillator inside of the microcontroller. We can use the watchdog timer to force the Microcontroller into a Known state of reset when the system is hung up or out of the control due to the execution of an incorrect sequence of codes.

WDT = ON (watchdog timer is enabled)

WDT =OFF (Watchdog timer is disabled)

In my project WDT is disabled.

2.2.4 Analog to Digital converter (ADC)

Analog-to-digital converters are among the most broadly utilized devices for data acquisition. Advanced PCs use binary (discrete) values, yet in the physical world, everything is analog (continuous). Temperature, pressure (wind or liquid), humidity, and velocity are a couple of illustrations of physical quantities that we manage each day. Converting a physical quantity to an electrical (voltage, current) signals using a device called Transducer. Transducers are additionally alluded to as a sensor. Sensor for velocity, temperature, light, pressure and numerous other natural quantities produce an output that is voltage (or current). Subsequently, we require an analog-to-digital converter to interpretation the analog signals to digital numbers so that the microcontroller can read and process them, as shown in figure 2.7 [7].

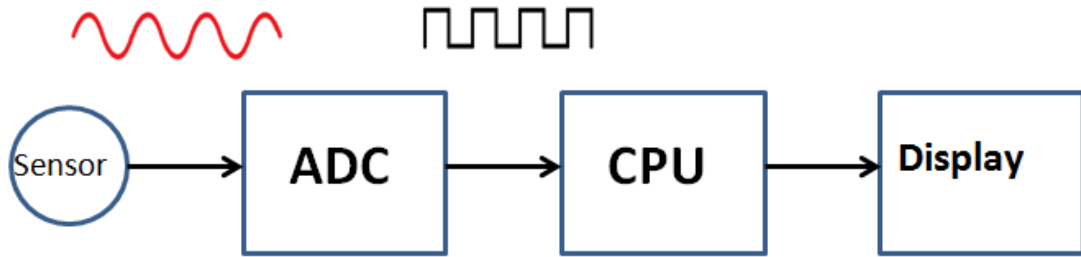


Figure 2.7 Microcontroller connection to Sensor via ADC

2.2.4.1 ADC Resolution

ADC of PIC Microcontroller has a n-bit resolution, 8-bit is a lower resolution and 24-bit is the highest resolution, generally n can be 8, 10, 12, 16, and 24-bit resolution. The smallest step size of the ADC depended on the n-bit of the ADC, then a large number of n provides higher resolution. Where to step size is the littlest change that can be recognized by an ADC. Resolution is the most important thing in an ADC.

For instance, If the reference voltage of ADC is 0 to 5v then 8-bit ADC will beyond the range in 256 divisions so it can measure it precisely up to $5/255v=19mv$ roughly. While the 10-bit ADC will beyond the range in $5/1024=4.8mv$ roughly. So we can see that the 8-bit ADC can't differentiate somewhere around 1mv and 18mv. PIC18F458 is a 10-bit resolution as shown in figure 2.8. Also, Table 2.4 illustrates the wide resolution of the ADC.

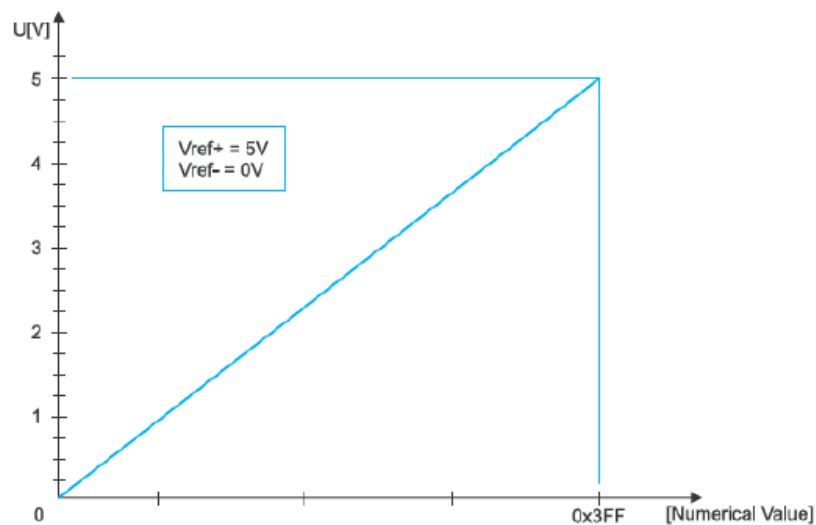


Table 2.4 bit versus resolution of the ADC

n-bit	Number of steps	Step size (mV)
8	256	5/256=19.53
10	1024	5/1024=4.88
12	4096	5/4096=1.2
16	65536	5/65536=0.076

2.2.4.2 Digital data output of A/D converter

In a 10-bit ADC we have a 10-bit digital data output of D0-D9 while in the 12-bit ADC the data output is D0-D11. We have the following formula to find the output voltage

$$D_{out} = \frac{V_{in}}{\text{step size}} \quad 2.1$$

2.2.4.3 Configuration of the A/D converters registers and bit control

There are four registers of A/D converter;

1. ADRESL Contains low byte of conversion result (storing result).
2. ADRESH Contains high byte of conversion result (storing result).
3. Control register 0 (ADCON0).
4. Control register 1 (ADCON1).

Figure 2.9 shows the A/D register block diagram.

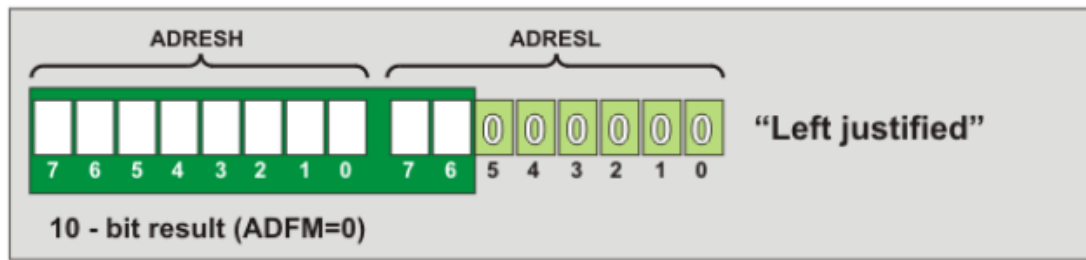


Figure 2.11 Right justified A/D converter data storing (ADFM bit =0)

Right justified used in my project.

2.2.4.3.2 A/D ACQUISITION REQUIREMENTS

Necessary to increase the accuracy of ADC, To get this accuracy must be determined the certain time delay between selecting specific analog input and measurement itself. This time is called 'acquisition time'. There is a function used to determine this time accurately, this time, must be greater than 20 μ S OR 11 times TAD.

2.2.4.4.3 ADC CLOCK PERIOD

The required time during conversion process for completing one bit is called TAD. This time, the period must be equal or greater than 1.6 μ S. For full conversion 10-bit, ADC required 11 TAD. According to using high frequency (short time period) in PIC microcontroller and for solving this problem, need to increase the time period without changing the clock source by configuring an ADCS1 and ADCS2 bits in ADCON0 register. Illustrates in Tables 2.5.

Table 2.5 Configure ADC Clock period

ADC Clock Source	ADCS1	ADCS0	Device Frequency (F_{osc})			
			20 Mhz	8 Mhz	4 Mhz	1 Mhz
$F_{osc}/2$	0	0	100 nS	250 nS	500 nS	2 μ S
$F_{osc}/8$	0	1	400 nS	1 μ S	2 μ S	8 μ S
$F_{osc}/32$	1	0	1.6 μ S	4 μ S	8 μ S	32 μ S
Frc	1	1	2 - 6 μ S	2 - 6 μ S	2 - 6 μ S	2 - 6 μ S

In my project I used $F_{osc}/32$ and 4 MHz Oscillator, So the Clock period is $8 \mu\text{s}$ (TAD), Thus total ADC time conversion is $88 \mu\text{s}$. As shown in figure 2.12.

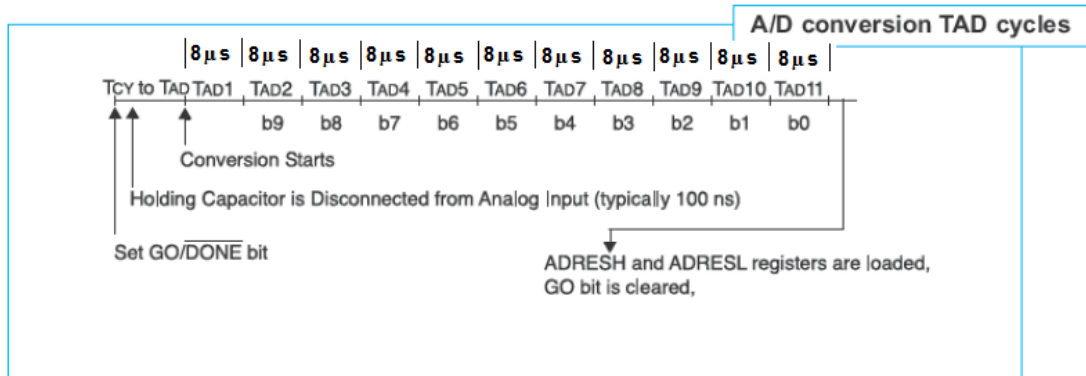


Figure 2.12 (10-bit) ADC expense period

2.2.4.3.4 ADCON0 Control register 0

There are 8-bit ADCON0, as shown in figure 2.13.

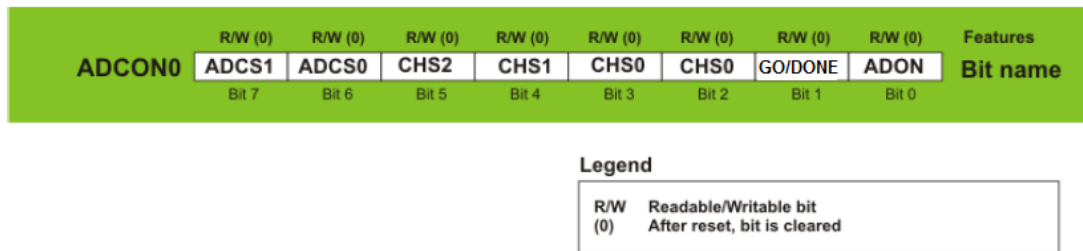


Figure 2.13 ADCON0 register

Bit 0 of ADCON0 used for enabling and disabling ADC, When ADON bit is (1) means is that the converter is enabled and vice versa.

Bit-2 is an A/D Conversion Status bit (GO/DONE), It is used to start conversion and monitor to see it, mean is that its use in determining the current status of the conversion, if (GO/DONE =1) mean is that conversion in progress, or (GO/DONE =0) mean is that conversion is complete, after completing the conversion ADON is cleared automatically by the hardware.

Bit-3, bit-4 and bit-5 (CHS0-CHS2) are analog channel select bits; these 3- bits are responsible for selecting a pin or an analog channel in the conversion process, Table 2.6 illustrates the channel selections.

Table 2.6 ADCON0 channel selection.

CHS2	CHS1	CHS0	CHANNEL SELECTION
0	0	0	CHAN0 (AN0)
0	0	1	CHAN1 (AN1)
0	1	0	CHAN2 (AN2)
0	1	1	CHAN3 (AN3)
1	0	0	CHAN4 (AN4)
1	0	1	CHAN5 (AN5) not implemented on 28-pin PIC18
1	1	0	CHAN6 (AN6) not implemented on 28-pin PIC18
1	1	1	CHAN7 (AN7) not implemented on 28-pin PIC18

The conversion time is set with ADCS0, ADCS1 bits in ADCON0 register and ADCS2 in ADCON1 register.

2.2.4.3.5 ADCON1 register

Figure 2.14 shows the ADCON1 register, PCFGs are analog to digital converter configuration control bit, this 4-bit are responsible for which pin is analog or digital and a voltage reference, as shown in figure 2.15.

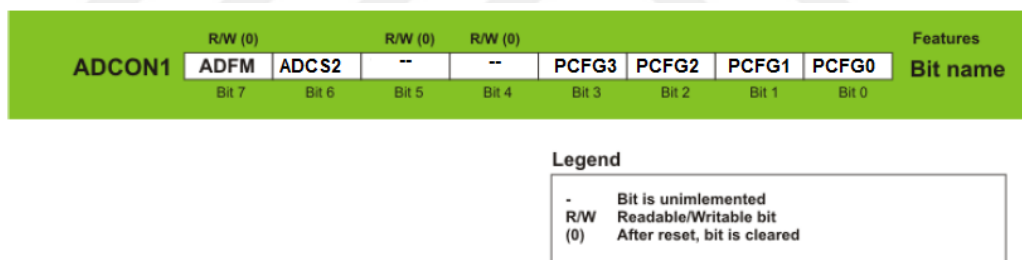


Figure 2.14 ADCON1 register

PCFGs	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	Vref+	Vref-	C/R
0000	A	A	A	A	A	A	A	A	Vdd	Vss	8/0
0001	A	A	A	A	Vref+	A	A	A	AN3	Vss	7/1
0010	D	D	D	A	A	A	A	A	Vdd	Vss	5/0
0011	D	D	D	A	Vref+	A	A	A	AN3	Vss	4/1
0100	D	D	D	D	A	D	A	A	Vdd	Vss	3/0
0101	D	D	D	D	Vref+	D	A	A	AN3	Vss	2/1
011x	D	D	D	D	D	D	D	D	-	-	0/0
1000	A	A	A	A	Vref+	Vref-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	Vdd	Vss	6/0
1010	D	D	A	A	Vref+	A	A	A	AN3	Vss	5/1
1011	D	D	A	A	Vref+	Vref-	A	A	AN3	AN2	4/2
1100	D	D	D	A	Vref+	Vref-	A	A	AN3	AN2	3/2
1101	D	D	D	D	Vref+	Vref-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	Vdd	Vss	1/0
1111	D	D	D	D	Vref+	Vref-	D	A	AN3	AN2	1/2

A = Analog input, D = Digital I/O

C/R = # of analog input channels / # of pins used for A/D voltage reference

The default is option 0000, which gives us 8 channels of analog input and uses the Vdd of PIC18 as Vref.

Figure 2.15 ADCON0 channel selection

2.2.5 Clock oscillator

Oscillator circuit is a very important thing to generate a clock of the microcontroller. The Clock is required for executing a program or program instructions. In generally; there are two parts of oscillators in Microcontroller:

1. Internal oscillator
2. External oscillator

2.2.5.1 Internal oscillator

Internal oscillator consists of two internal oscillators separately:

The high-frequency internal oscillator (HFINTOSC) which works at 8MHz. After using prescalers The Microcontroller can use this clock source in different frequencies. Also, low-frequency internal oscillator (LFINTOSC) which works on 31 kHz. SCSSEN bit decides to select which oscillator are used (internal or external) in the CONFIG1H register, the figure 2.16 shows the types of oscillators block.

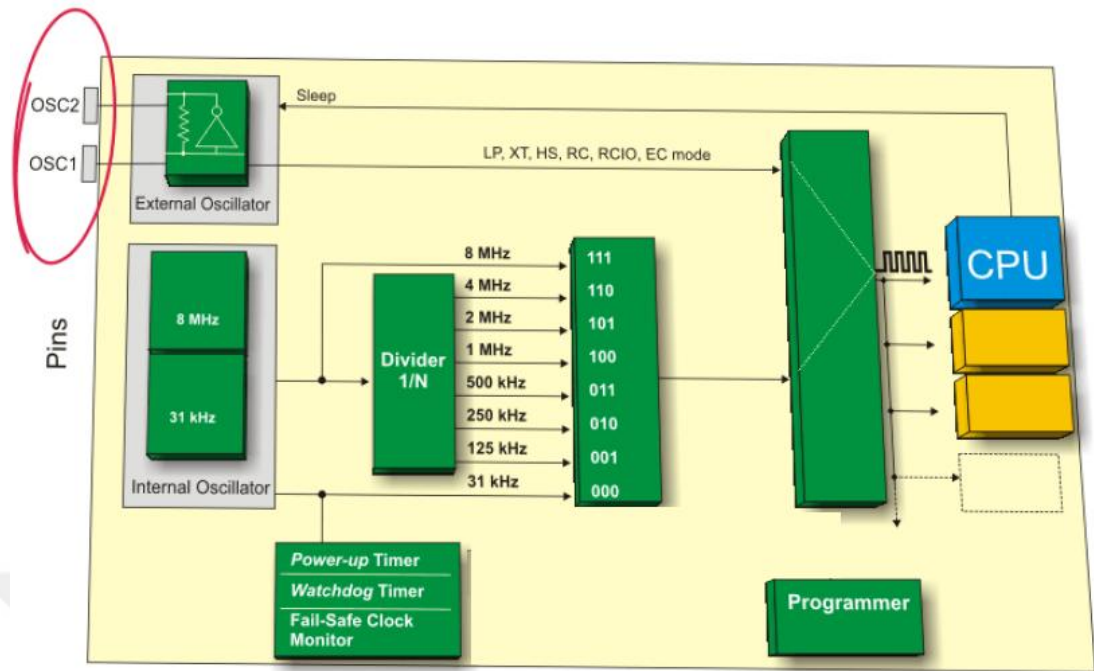


Figure 2.16 Internal and external oscillator of microcontroller

2.2.5.2 External oscillator

There are several modes of external oscillator clock, During the writing of the programming process of Microcontroller, the operation mode is selected by the user. The (EC) mode uses an external oscillator as a clock source. The limited range of this Oscillator restricted about 20 MHz as shown in figure 2.17.

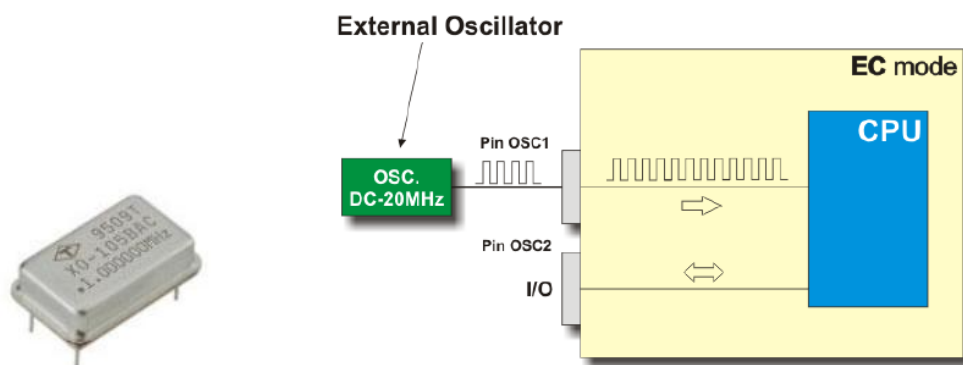


Figure 2.17 External oscillator (EC) mode

Quartz crystal or ceramic is used as an external clock to provide any mode of clock source (LP, XT and HS) modes.

In PIC18F458, the OSC1 and OSC2 pins are connected to the quartz crystal and ceramic resonator terminals and across the two capacitors for providing clock frequency as shown in figure 2.18. Also, the size of capacitors selects the frequency range of clock as shown in table 2.7.

Table 2.7 PIC18F458 Oscillator Frequency Choices and Capacitor Range

Osc choice	Crystal Freq	C1 range	C2 range
LP	32 kHz	33 pF	33 pF
LP	200 kHz	15 pF	15 pF
XT	200 kHz	47–65 pF	47–65 pF
XT	1 MHz	15 pF	15 pF
XT	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
HS	8 MHz	15–33 pF	15–33 pF
HS	20 MHz	15–33 pF	15–33 pF
HS	25 MHz	15–33 pF	15–33 pF

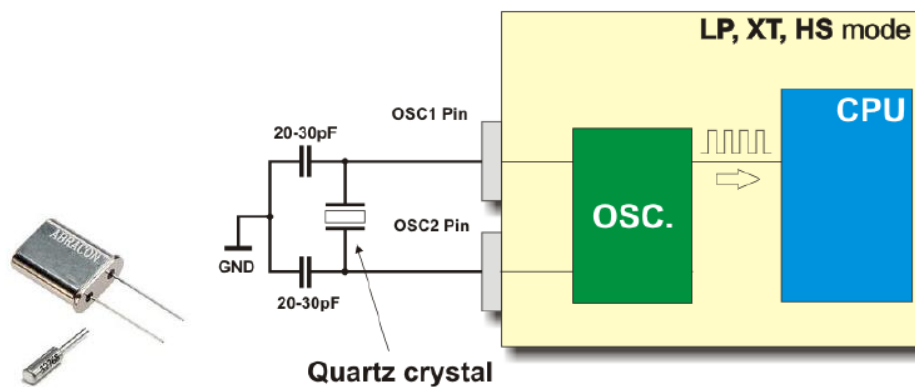


Figure 2.18 Quartz crystal oscillators and block diagram connection

The connection of Ceramic resonators to provide clock frequency is similar to crystal oscillator additionally ceramic resonator cheaper than a crystal oscillator due to not good characteristics. As shown in figure 2.19.



Figure 2.19 Ceramic oscillator

For the purpose of connecting RC mode OSC1 pin connected to RC circuit, another pin (OSC2) provides less frequency by 4 times the (OSC1/4). This frequency is useful for several applications like calibration, synchronization etc., as shown in figure 2.20.

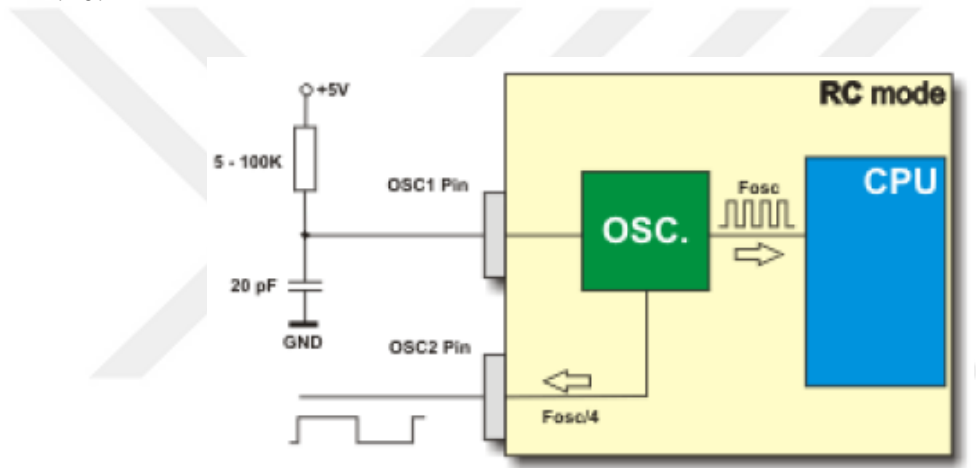


Figure 2.20 RC oscillator and connection with a microcontroller

Also in RCIO mode, it has the same way for connection OSC1 pin to the RCIO circuit, but another pin (OSC2) it's free to be used as input or output of general purpose. As shown in figure 2.21.

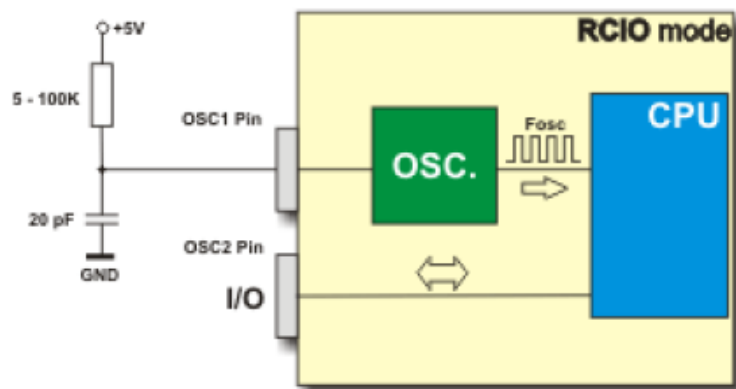


Figure 2.21 RCIO oscillator and connection with a microcontroller

2.2.6 ECS-100AC 4Mhz (quartz crystal oscillator)

The ECS-100 clock oscillator is fully compatible with TTL (transistor-transistor logic) circuitry. The metal package with pin-7 case ground acts as shielding to minimize radiation. Also, this type oscillator is low cost. Figure 2.22 illustrates the pin diagram and output waveform.

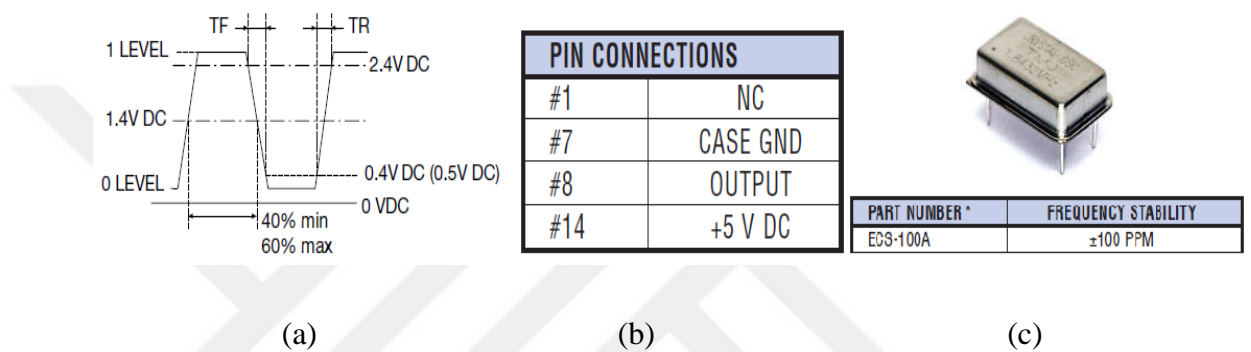


Figure 2.22 (a) ECS-100AC waveform 2.21(b) pin connections (c) Oscillator package

2.2.7 LM35 temperature sensor

The LM35 is an incorporated circuit sensor. As shown in Figure 2.23. It senses the temperature and changed to electricity in the output relative to the temperature in centigrade ($^{\circ}\text{C}$).

The output voltage of this sensor higher than thermocouples consequently does not need it for amplification. The scale of voltage output versus temperature is 10 mV for each one degree ($10\text{mV}/^{\circ}\text{C}$). Also the important thing of LM35 is, does not need external calibration, Also it has good accuracy about $\pm 4.0^{\circ}\text{C}$ at room temperature, But at out the range of (0°C to 100°C) as its accuracy decreases to $\pm 0.8^{\circ}\text{C}$. Also the LM35 has very low power consumption about $60\mu\text{A}$. [9].

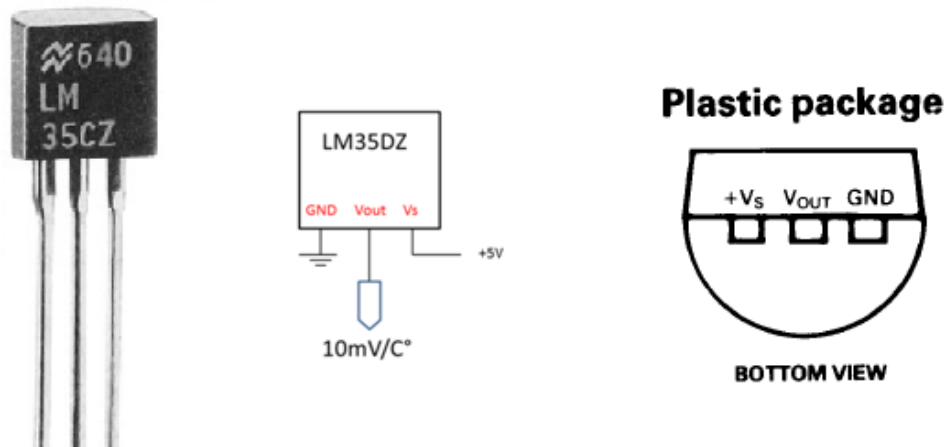


Figure 2.23 LM35 temperature sensor

2.2.8 Controller Area Network (CAN) bus System

2.2.8.1 Introduction

The CAN bus was developed by Robert Bosch in the 1980's and is used as a multi-master protocol that can have a maximum signal rate of 1Mbps(CIA). The idea of CAN came about to solve an ever-increasing demand for the multiple Electronic Control Module (ECM) networking. Unlike USB or Ethernet protocols, CAN uses smaller packet sizes to deliver messages throughout the CAN bus. For this reason, messages like temperature, humidity or even engine RPM(revolution per minute) could be sent from one node to another with accuracy and with little cost.

CAN is an International Standardization Organization (ISO) characterized serial communications bus. Initially created for the car business that helps to reduce wiring. For these elements, CAN's prevalence is utilized as a part of an assortment of commercial enterprises, including automation of building, medical, and factories. The communication of CAN protocols, ISO-11898 depicts how data are gone between devices on the network. The actual communication between gadgets associated with the physical medium is characterized by the physical layer of the model. The ISO 11898 architecture defines the Data Link Layer (DLL) and the Physical Layer to ensure compatibility between CAN transceivers and only the transceivers.

The CAN communications protocol, ISO-11898 describes how information is passed between devices on the network. Actual communication between devices connected by the physical medium is defined by the physical layer of the model. The ISO 11898 architecture defines the Data Link Layer (DLL) and the Physical Layer to ensure compatibility between CAN transceivers and only the transceivers.

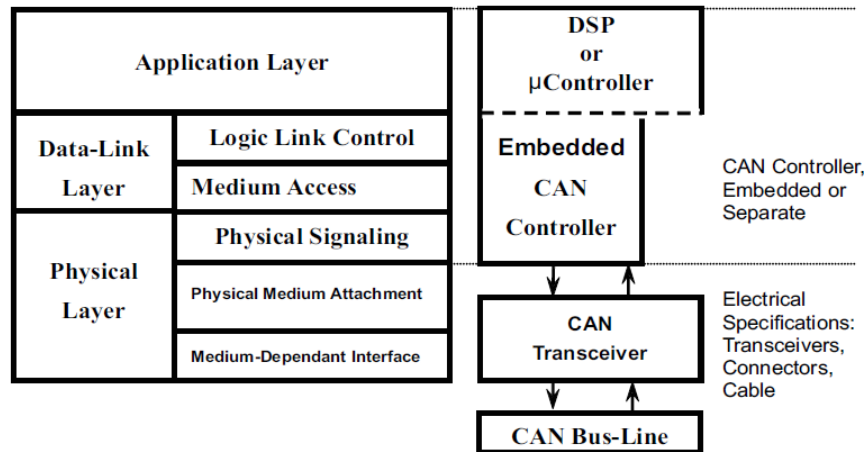


Figure 2.24 The Layered ISO 11898 Standard Architecture

In Figure 2.24, the DLL alongside the Medium Access Control (MAC) layer of the CAN protocol characterizes the non-damaging bit-wise arbitration, so the message with the most elevated earlier identifier will get sent first.

2.2.8.2 CAN protocol

- CAN is a serial protocol.
- It's a Multimaster, Multicast, protocol.
- It supports distributed real-time control.
- Currently, Controller Area Network (CAN) is not restricted to the auto industry.

Multi-master; when the bus is free any node might begin to send a message. Which node has the largest priority can send the message to the bus.

Multicast; numerous node might get and follow up on the message at the same time.

2.2.8.3 Layers of CAN

CAN divide into three layers:

- The Object Layer
- The Transfer Layer
- The Physical Layer

The Object and Transfer Layers include all administrations and elements of the Data Link layer characterized by the ISO/OSI model. Likewise, the physical layer performs the real exchange of bits between various nodes.

2.2.8.4 CAN Message Frames

Can message includes four sorts of frame:

1. Data frame: It conveys information from a transmitter to the collectors.
2. Remote frame: It is transmitted by a node to ask for the transmission of a Data frame with the same identifier.
3. Error frame: It is transmitted by any hub in the wake of distinguishing a bus mistake.
4. Overload frame: It is utilized to give extra defer between two information or remote frames.

2.2.8.4.1 Data frame

- Standard CAN (11-bit identifier)
- Extended CAN (29-bit identifier)

Data Frame consists a 7 field. As shown in figure 2.25.

Arbitration Field

S	11-bit Identifier	R	I	r0	DLC	0...8 Bytes Data	CRC	ACK	E	I
O		T	D						O	F
F		R	E						F	S

Field	Bits	Field	Bits
Start of frame	1	Data field 0 – 8 bytes	0 – 64
Identifier	11	CRC	15
Remote trans.	1	CRC delimiter	1
ID extension	1	ACK slot	1
Reserved	1	ACK delimiter	1
Data length code	4	End of frame	7

Figure 2.25 Standard CAN (11-Bit Identifier) [10]

- Start of Frame (SOF) has denoted the start of a data or remote frame. It comprises of one and an only dominant bit. All nodes need to synchronize with the main node of the begin of frame bit.
- Identifier- The 11-bit identifier establishes the priority of the message.
- RTR–The single remote transmission request (RTR) bit RTR is 0 means this node need to receive data from another node. Also, if RTR is 1 means this node need to transmit data to the other node.
- IDE–A dominant single identifier extension (IDE) bit implies that a standard CAN identifier with no extension is being transmitted.
- r0–Reserved bit.
- DLC–The 4-bit data length code (DLC) contains the number of bytes of data being transmitted. This field can be in lengths of 0-8 bytes.
- Data The data field contains the data to be transmitted.
- CRC– It’s a 16-bit field used to improve transfer reliability
- ACK– This field consists of 2-bits, 1-bit for the acknowledge slot and the other 1-bit is for the acknowledge delimiter. A transmitting message sends two recessive bits while a receiving node will send a dominant bit after a valid message.

- EOF – Each message is ended by a flag sequence, including of seven recessive bits.
- IFS–This 7-bit interframe space (IFS) consists the time required between messages that are being sent over the bus.

2.2.8.5 The CAN bus

The data link and physical signaling layers of Figure 2.27 are regularly straightforward to the system operator and are included with any controller that implements the CAN protocol. Connection to the physical medium is then is implemented through a line transceiver, such as MicroChip’s MCP2561 (MicroChip) which was the chosen Transceiver for my project. Figure 2.26 below shows the bus connections for any amount of nodes connected to a system. Here you will also see the data direction for which the controllers and transceivers send and receive messages.

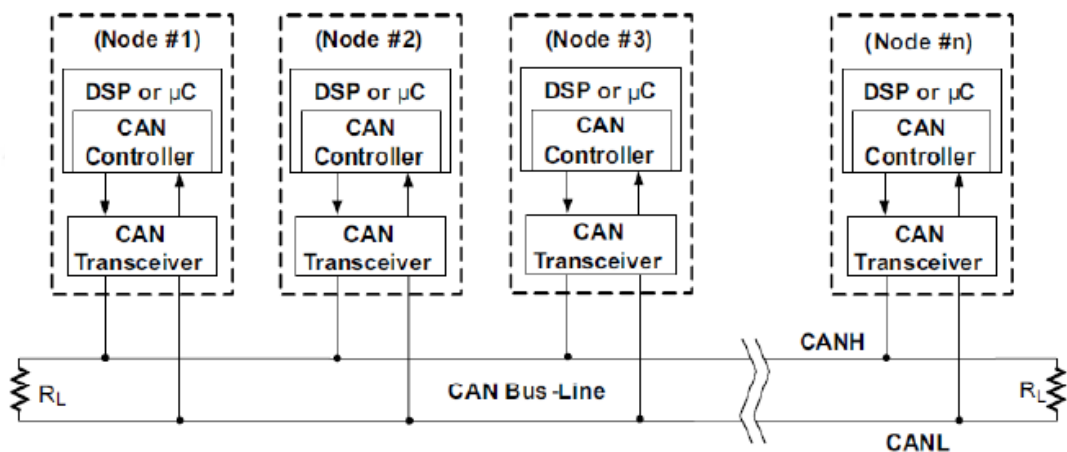


Figure 2.26 Physical Bus Connections

Messages are sent across the bus using differential signals. Differential signals offer superior in the reduction of noise immunity and allow for fault tolerance. This is done using a twisted-pair cable which helps reduce noise and allows for high signal rates. Since the cable is twisted, the signal line current flows in each line of pairs equally but in a different direction. The resulting field-canceling effect. Unwanted noise has the potential for data corruption and ultimately leads a system to be unstable.

The ISO 11898 standard specifies that a signal rate of 1Mbps can have a maximum bus length of 40 meters. As the bus length increases the rate at which the bus can operate decreases. Figure 2.27 [11] below shows the correlation. In this figure, You can see that a 400-meter bus length will have a bus rate of 100Kb/s. The reason why the bus speed decreases because the capacitance builds up between the two wires.

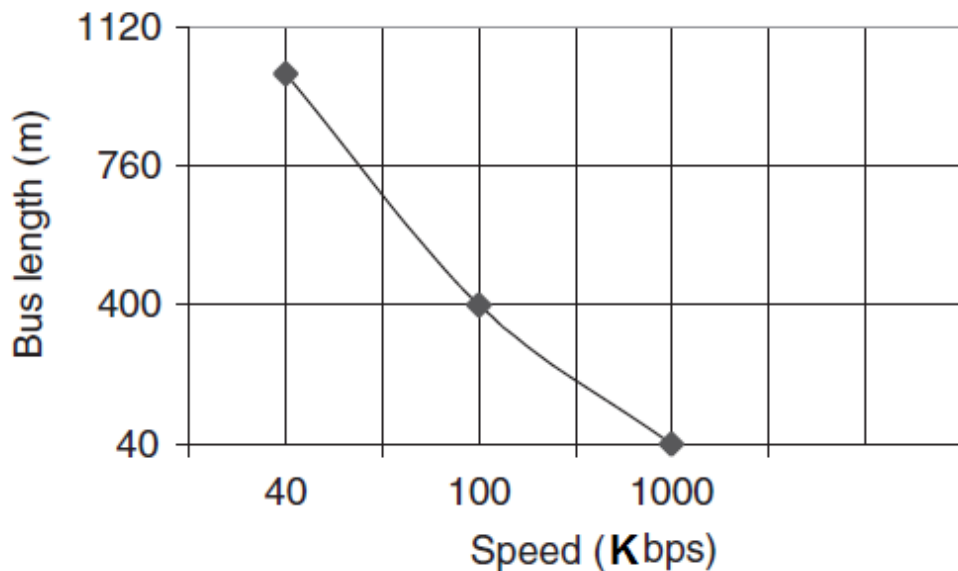


Figure 2.27 CAN bus speed and bus length [11]

The twisted wire or cable can be shielded or it can be unshielded depending on the location of the application. The standard also calls for the twisted-pair to be connected with a 120Ω resistor at each of pair ends which minimize signal reflection. As shown in Figure 2.28.

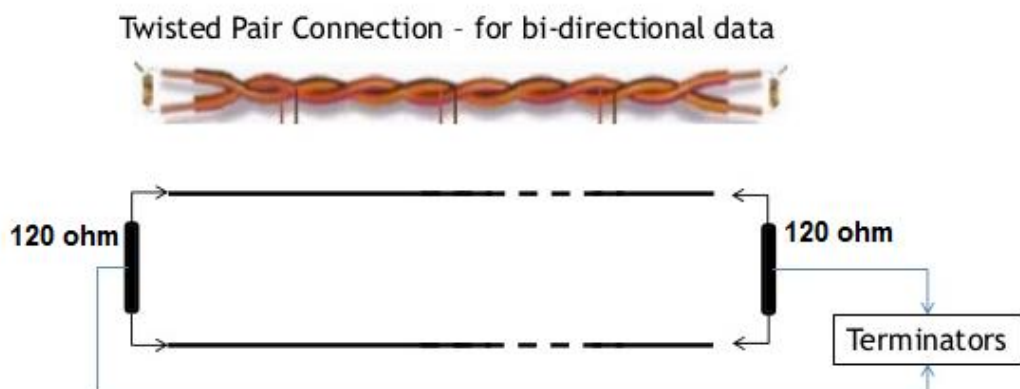


Figure 2.28 Twisted Pair CAN Bus and termination connection

The data on CAN bus is different and can be in two states: dominant(CANL) and recessive (CANH). The state of voltages on the bus. The bus defines a logic bit 0 as a dominant bit and a logic bit 1 as a recessive bit. as displayed in Figure 2.29.

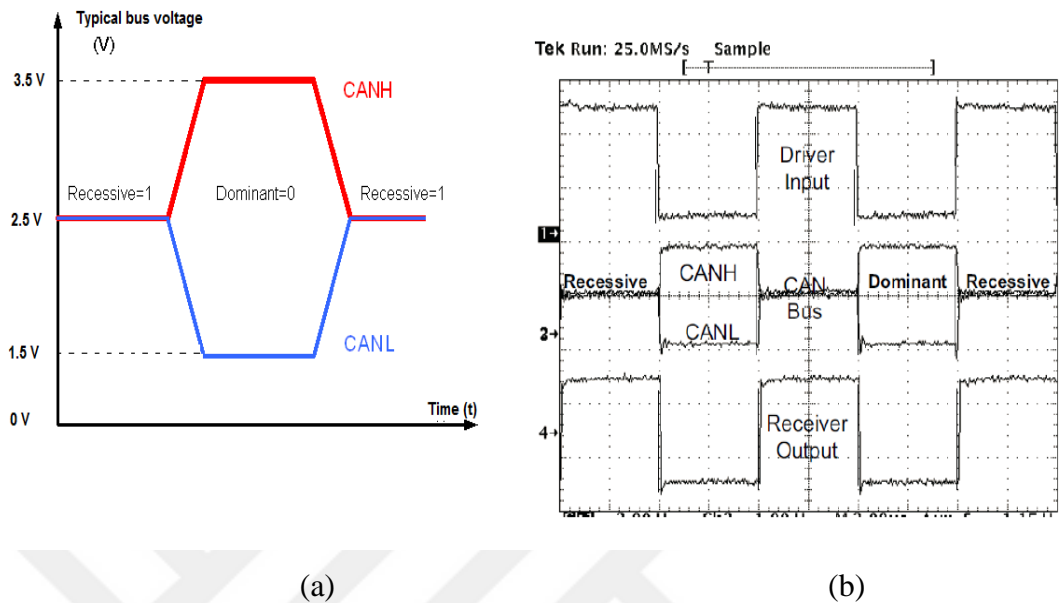


Figure 2.29 (a) CANH and CANL bus voltage (b) CANH and CANL Signals [10]

2.2.9 Liquid Crystal Display (LCD)

LCD is a very suitable instrument for displaying the standard map character. A (2x16) LCD consists 2 lines of character displays, Also the number of 16 means, This LCD has the capacity of 16 characters for displaying. An LCD is capable of interfacing with a microcontroller and normally used in various devices, as shown in figure 2.30. Additionally, it has more popularity than 7-segment and multi-segment LED displays. The advantages of LCD's are as follows

- Less expensive.
- It can program easily.
- Display long number of characters.
- Very light and compact.
- Power consumption is very low.
- Easy programming for displaying the characters and graphics [12].



Figure 2.30 2×16 LCD DISPLAY

2.2.9.1 LCD Display pins

Normally the LCD display has 14 pins some of them has extra 2 pins for the backlight of it. These pins used for communication with microcontroller one side of the small printed board of the LCD display there are pins that enable it to be connected to the microcontroller. There are all together of 16 pins as appeared in figure 2.31. Their function is portrayed in the figure 2.32.

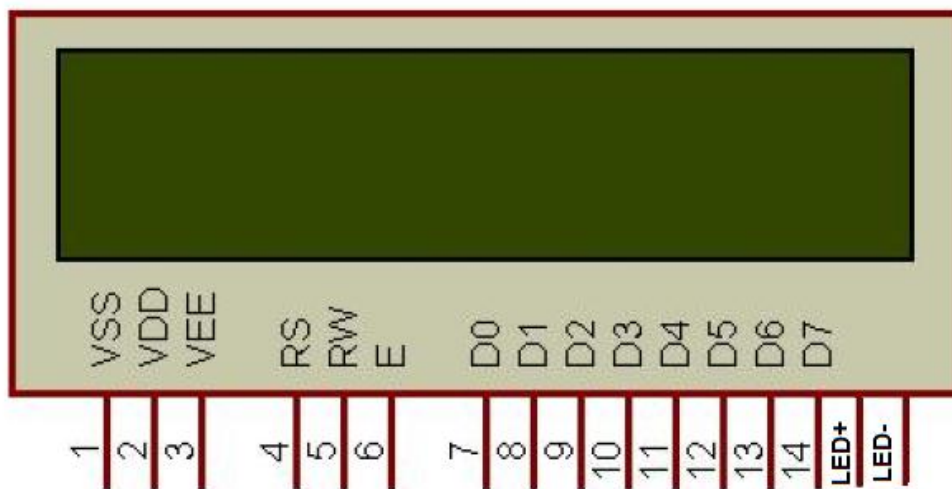


Figure 2.31 LCD Pin Diagram

Function	Pin Number	Name	Logic State	Description
Ground	1	Vss	-	0V
Power supply	2	Vdd	-	+5V
Contrast	3	Vee	-	0 - Vdd
Control of operating	4	RS	0 1	D0 – D7 are interpreted as commands D0 – D7 are interpreted as data
	5	R/W	0 1	Write data (from controller to LCD) Read data (from LCD to controller)
	6	E	0 1	Access to LCD disabled Normal operating
			From 1 to 0	Data/commands are transferred to LCD
Data / commands	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1
	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	13	D6	0/1	Bit 6
	14	D7	0/1	Bit 7 MSB

Figure 2.32 LCD pin description

Vcc, Vss, and VEE, While VCC and Vss provide +5V and ground, respectively, VEE is used for controlling LCD contrast. Also RS is called select register, this register is used for their selection as follows. If RS is a logic zero(0V) allowing the user to send commands to the display throw D0-D7 pins, such as a clear display as shown in figure 2.32. If RS is a logic one(+5V), allowing the user to send data to be displayed on the screen.

R/W(read/write) input allows the user to write information to the LCD or read information from it.

E (Enable) pin is used by the LCD to latch information presented to its data pins.

Also, the remaining 8-bit pins, are used to send data to the LCD or read contents of the LCD's internal register.

2.2.9.2 LCD screen

A 2x16 LCD means it is able to display 32 characters for the entire display. Also, the first and second row have the capacity of 16 characters. Each of character includes (5x8) or (5x11) pixel matrix. But a 5x8 character display which is most ordinarily utilized. The dots matrix as shown in figure 2.33.

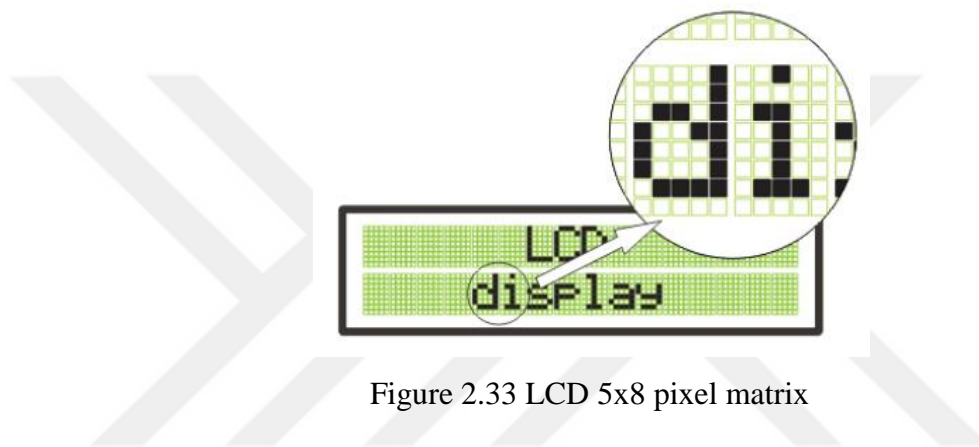


Figure 2.33 LCD 5x8 pixel matrix

2.2.9.3 LCD memory

There are three main blocks in LCD memory:

- Display Data RAM (DDRAM)
- Character Generator RAM (CGRAM)
- Character Generator ROM (CGROM)

2.2.9.3.1 DDRAM (Display Data RAM) memory

DDRAM memory is utilized for putting away characters to be shown. The memory size of DDRAM is equipped for putting away 80 characters. Some memory locations straightforwardly interface with the characters on display. The first line addresses, starts from 00hex to 27hex, also the second line address starts from 40hex to 67hex, as shown in figure 2.34.

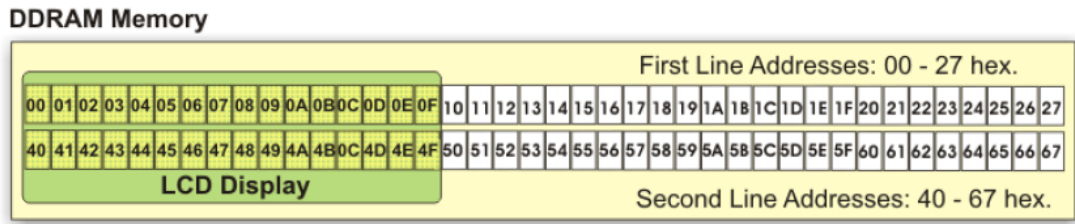


Figure 2.34 DDRAM memory address

2.2.9.3.2 CGROM

CGROM memory contains a standard character map with all characters that can be shown on the screen. Every character is allocated to one memory location:

Lower 4 Bits	Upper 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	1	A	Q	a	q				-	9	3	α	ρ
xxxx0001	(2)		!	1	A	Q	a	q				。	ア	チ	4	ä	q
xxxx0010	(3)		"	2	B	R	b	r				「	イ	ツ	×	ρ	θ
xxxx0011	(4)		#	3	C	S	c	s				」	ウ	テ	ε	ω	
xxxx0100	(5)		\$	4	D	T	d	t				、	エ	ト	ト	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u				・	オ	ナ	1	ε	Ü
xxxx0110	(7)		&	6	F	V	f	v				ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w				ア	キ	ヌ	ラ	α	π

Figure 2.35 CGROM memory map

The addresses of CGROM memory locations match the characters of ASCII. In the event that the system being right now executed encounters a command 'send character P to port' then the binary value 0101 0000 shows up on the port. This value is the ASCII equivalent to the character P, as shown in figure 2.35.

2.2.9.3.3 CGRAM

Aside from standard characters, the client can characterize the images and show it on the LCD. Additionally, the span of images must match with the 5x8 pixels. CGRAM has 64 bytes of size. Memory registers have 8 bits wide, but Only 5 lower bits are

utilized in light of the restricted of (5x8 pixels). Putting away each logic (1) represented a dot on the screen, while 8 locations gathered to together represent one character. It is shown in figure 2.36.

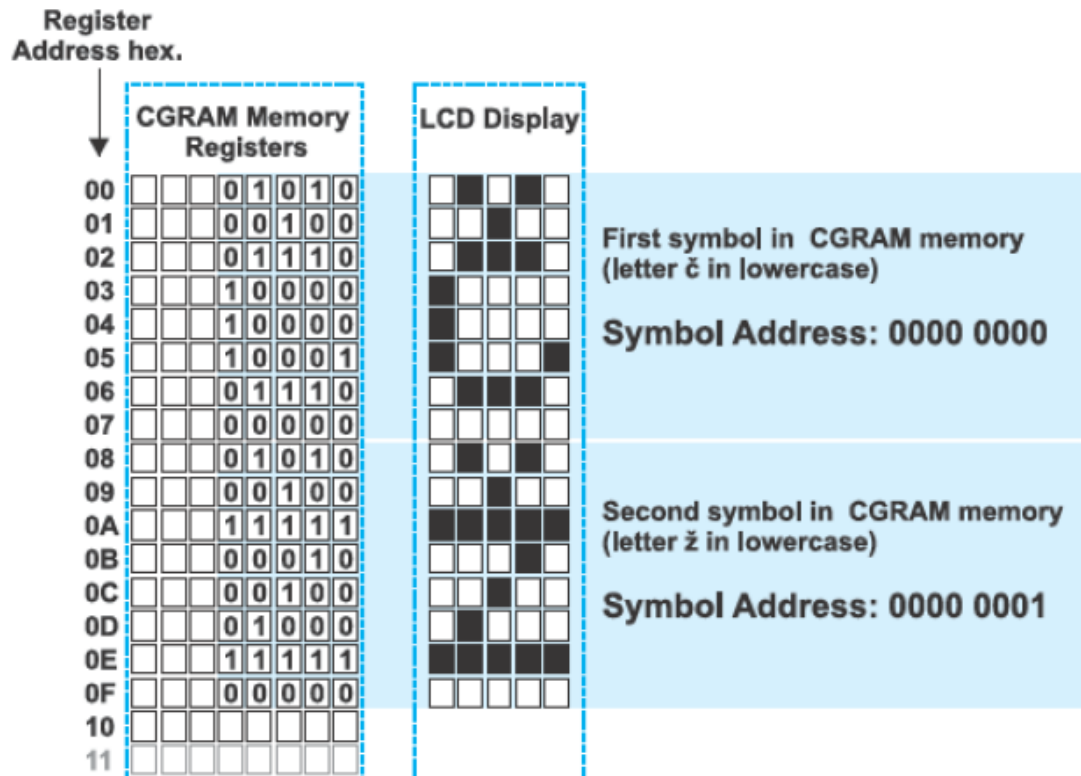


Figure 2.36 CGRAM memory location a dress

2.2.9.4 LCD Basic Commands

- If RS is a logic one, D0 to D7 pins ready for displaying the data.
- If RS is a logic zero, D0 to D7 pins acts as a command for setting the display mode.

The figure 2.37, Shows the most commands is used daily.

Command	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Execution Time
Clear display	0	0	0	0	0	0	0	0	0	1	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	x	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	40uS
Display on/off control	0	0	0	0	0	0	1	D	U	B	40uS
Cursor/Display Shift	0	0	0	0	0	1	D/C	R/L	x	x	40uS
Function set	0	0	0	0	1	DL	N	F	x	x	40uS
Set CGRAM address	0	0	0	1	CGRAM address						40uS
Set DDRAM address	0	0	1	DDRAM address						40uS	
Read "BUSY" flag (BF)	0	1	BF	DDRAM address						-	
Write to CGRAM or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	40uS
Read from CGRAM or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	40uS

I/D 1 = Increment (by 1) 0 = Decrement (by 1)	R/L 1 = Shift right 0 = Shift left
S 1 = Display shift on 0 = Display shift off	DL 1 = 8-bit interface 0 = 4-bit interface
D 1 = Display on 0 = Display off	N 1 = Display in two lines 0 = Display in one line
U 1 = Cursor on 0 = Cursor off	F 1 = Character format 5x10 dots 0 = Character format 5x7 dots
B 1 = Cursor blink on 0 = Cursor blink off	D/C 1 = Display shift 0 = Cursor shift

Figure 2.37 LCD display command setting

2.2.9.5 LCD Connecting

There are two modes for interfacing between microcontroller and LCD. In 4-bit mode, only four higher bits (D4-D7) are used and connected to the microcontroller means the data is sent to the LCD in two stages, four higher bits are sent first (normally through the lines D4-D7), then four lower bits. Initialization enables the LCD to link and interpret received bits correctly. Another mode is 8-bits, in this mode all pins (D0 to D7) are used for connecting with a microcontroller to transmit and receive data.

2.2.10 LED diode

LEDs have two terminals one of them anode and the other is the cathode. Which emits light during applying a suitable voltage across the ends.

The largest current flows by the diode do not exceed 20 mA, so as to prevent this from damaging must connect resistance with it serially, shown in figure 2.38.

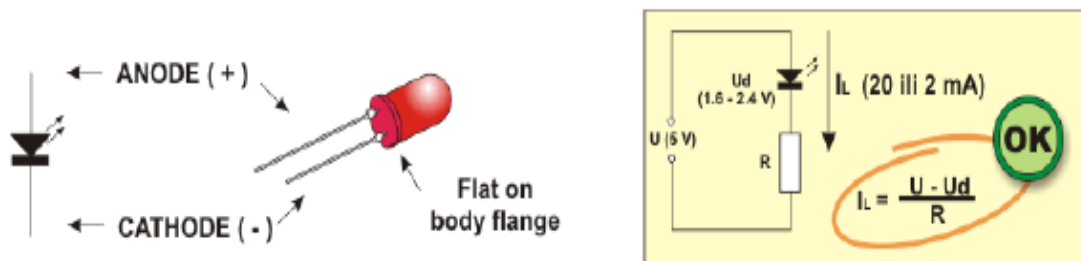


Figure 2.38 Lighting Emitting diode(LED)

2.2.11 MCP2561 Controller Area Network (CAN) Transceiver

The MCP2561 Shown in figure 2.39, is a high-speed CAN, fault-tolerant device that serves as the communication between a CAN protocol controller and the physical bus. The MCP2561 gives differential transmit and get the ability for the CAN protocol controller and is totally perfect with the ISO-11898 standard physical layer, including 5V necessities. It will work at speeds of 1 Mb/s.

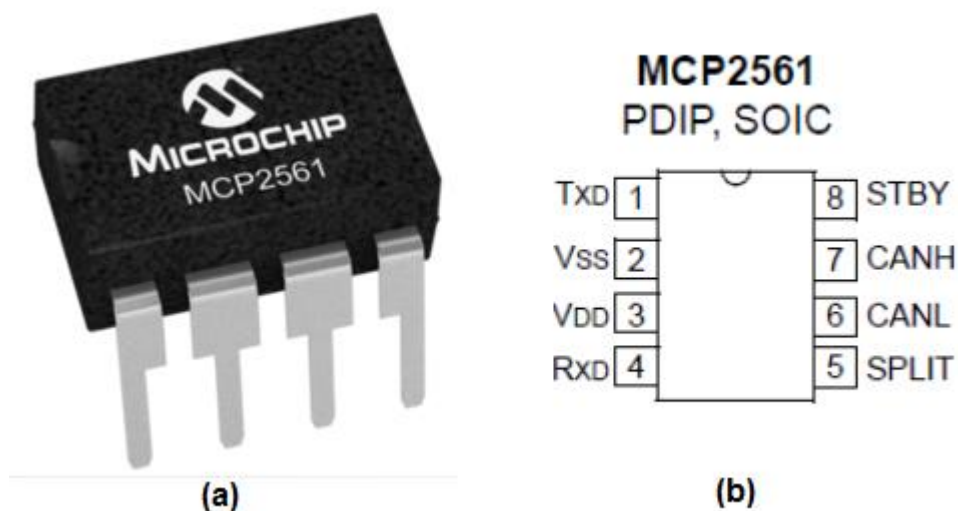


Figure 2.39 (a) MCP2561 chip (b) MCP2561 CAN Transceiver Pin Diagram

Regularly; In CAN system any node need to the device to convert the digital data provide by the CAN controller to a proper information over the twisted-pair wire are called CAN bus as shown in figure 2.40. It likewise gives a support between the CAN controller and the high voltage spikes that can be produced on the CAN Bus [13].

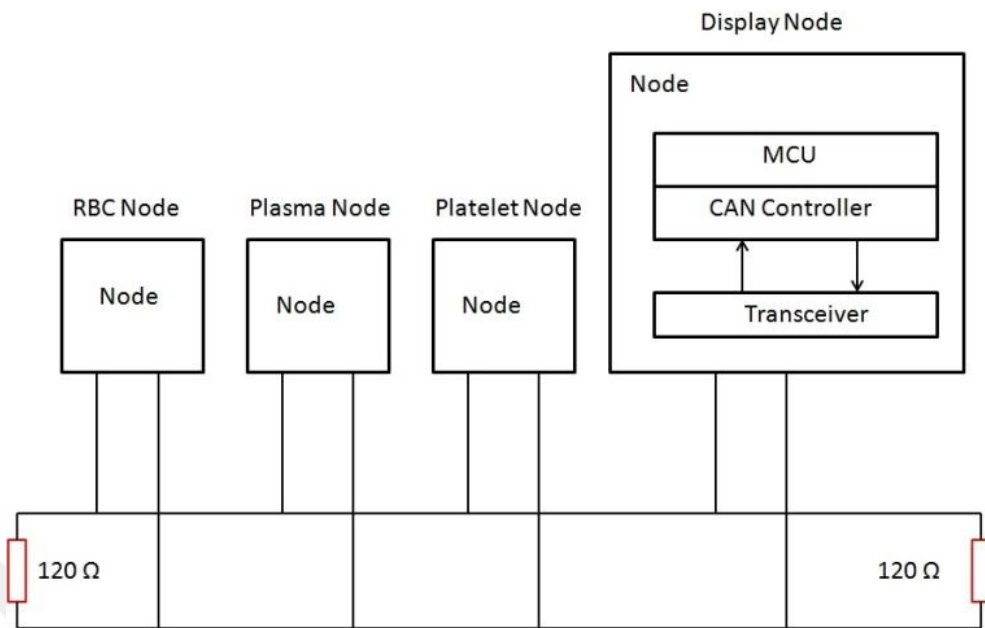


Figure 2.40 Four-Node CAN Bus Block Diagram

2.2.12 LM7805C Voltage Regulator

Voltage Regulator (7805) is a voltage regulator integrated circuit (IC) as shown in Figure 2.41. The voltage source in a circuit may have variations and would not give the constant output voltage. The voltage regulator means the output voltage at a stable value. 7805 provides +5V regulated power supply. Used 7805 prototypes of the regulator to feed the microcontroller.

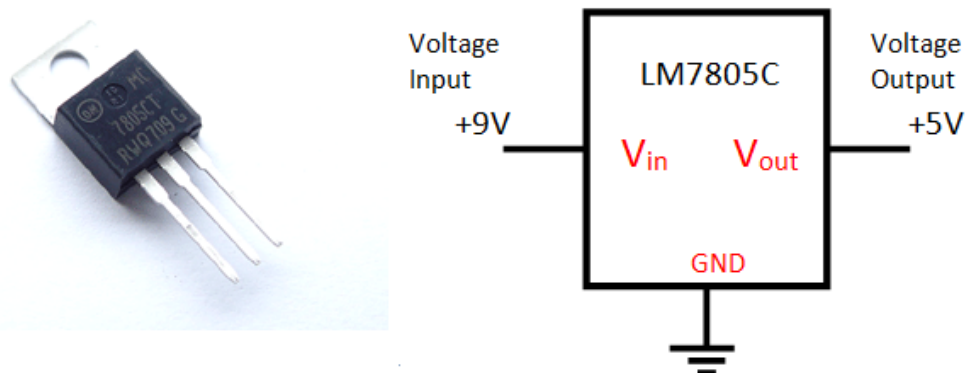


Figure 2.41 Voltage Regulator Circuit

2.2.13 ULN2003APG Seven channel Darlington Sink Driver

The ULN2003APG is integrated circuit(IC) consist of 7 NPN transistors, Also the output voltage and current capacity about 50V,500mA respectively [14][12]. As shown in figure 2.42.

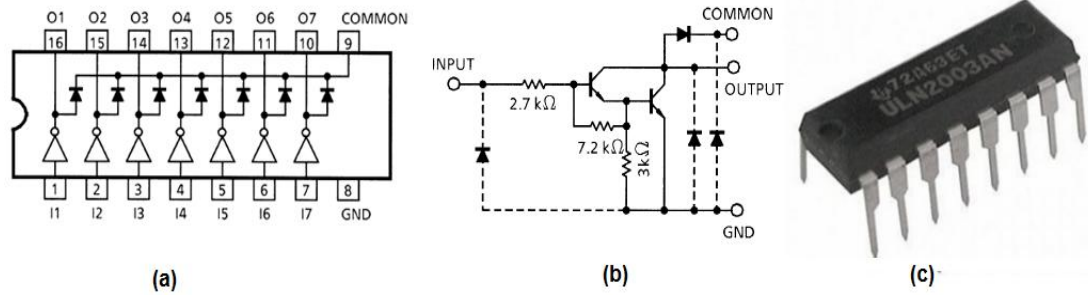


Figure 2.42 (a) 7-Ch Darlington Sink Driver Pin Diagram (b) 1-ch sink driver circuit (c) 7-ch Sink Driver chip

2.2.14 Buzzer

A bell or beeper is a sound flagging device, which might be mechanical, electromechanical, or piezoelectric. In this project this device is used for the purpose of alarm, this buzzer will activate when the override temperature of each refrigerator is occurring, figure 2.43 displays the 5v buzzer.



Figure 2.43 (5V) buzzer

2.2.15 Related work

The relevant research reports in the literature with regards to the field of Controller Area Network (CAN) system, Microcontrollers, and their conclusions are briefly cited below:

Steve Corrigan (2008) presented a brief introduction to the CAN operating principles, the execution of a fundamental CAN bus using Texas Instrument's CAN transceivers and DSPs. Some of the properties of CAN, especially relating to the electrical layer and features of transceiver products, are then discussed at a tutorial level [10].

Presi, T.P.(2013) Proposed and implemented by a CAN protocol using PIC for vehicle monitoring system. The main feature of the system includes monitoring of various vehicle parameters such as Temperature, presence of CO level in the exhaust, Battery Voltage and Light due to spark or fire. The software part is done in MPLab IDE using Embedded C. Schematic is prepared using OrCAD[15].

Sekhar, N.C, Reddy, T.S. ; Bhavani, G. et al (2014) proposed to implement low-cost MEMS (MicroElectro MechanicalSystems) based real-time temperature measuring, monitoring and control system which uses monolithic temperature sensors that provides a pulse width modulated digital output signal corresponding to its input temperature. Two different platforms are chosen to implement the MEMS based temperature control system. First one is using real time controlling software NI LabVIEW and another is using the PIC Microcontroller. As the implemented system uses advanced technologies [16].

Badri, M.A, Halim, A.K.,(2008) presented a design of moving message LCD display system (MMDS) via short message service (SMS) entry using Rabbit 2000 microcontroller. The objectives of the project are to design and integrate the hardware and software that interface the RCM2000 microcontroller, GSM module and LCD module in order to create MMDS [17].

Muhammad Ali Mazidi and Janice Gillispe Mazidi (2000) discussed the overview of 8051 microcontrollers. Microcontrollers and microprocessors are widely used in embedded system products. An embedded product uses a microcontroller to do one

task and one task only. In addition to the description of criteria for choosing a microcontroller, the interfacing with the real world devices such as LCDs, ADCs, sensors and a keyboard is described in detail [18].

S.J.Perez, M.A.Calva, R.Castañeda(2014) proposed a Microcontroller-based data logging system to record temperature and relative humidity for acoustic measurement applications. The system is simple to use, requires no additional hardware and allows the selection of the amount of data and the time intervals between them. The collected data can easily be exported to a PC computer via a serial port [19].

2.2.16 Scope of the work

The aim of the present work is to design and implement the circuit to monitor the temperature of 24 refrigerators inside blood bank center. The developed system includes four nodes, which are named as RBC, plasma, platelet, and display nodes. One node (called DISPLAY node) requests the temperature periodically and displays it on an LCD. This process is then repeated continuously. The second, third and fourth node reads the temperature of all refrigerators from an external semiconductor temperature sensor, this will be a linear active sensor that converts temperature to an analog voltage. The temperature sensor of choice is the LM35DZ, which provides us a10mV per degree Centigrade output with no calibration necessary. By using this sensor to read the output signal into PORTA, pin RA1of of the microcontroller. Also, each node is far from together in a different place. In addition, each node sends alarm address of each refrigerator to the LCD display when override temperature is occurring. PIC18F458 are used to implement the hardware architecture of each node, for programming of Microcontroller, MPLAB and C code is used. Also, PIC18F458 build in CAN module for communication between nodes [20].

CHAPTER 3

SOFTWARE AND CODE

3.1 Introduction

This chapter mentions about MPLAB IDE, using software and creating projects, also mentions about the C language, especially C library for PIC18F458, also mentions about PICKit 3 programmer tool for writing C Code to EEPROM of the PIC via MPLAB IDE.

3.2 MPLAB IDE

MPLAB IDE is a free integrated development environment for the development of embedded applications on PIC and dsPIC microcontrollers, and is produced by Microchip Technology [21]. MPLAB is intended to work with MPLAB-certified devices such as the MPLAB ICD 3 and MPLAB REAL ICE, for programming and debugging PIC Microcontrollers utilizing a personal PC. PICKit programmers are likewise upheld by MPLAB [22], figure 3.1 displays the screenshot of MPLAB IDE software.

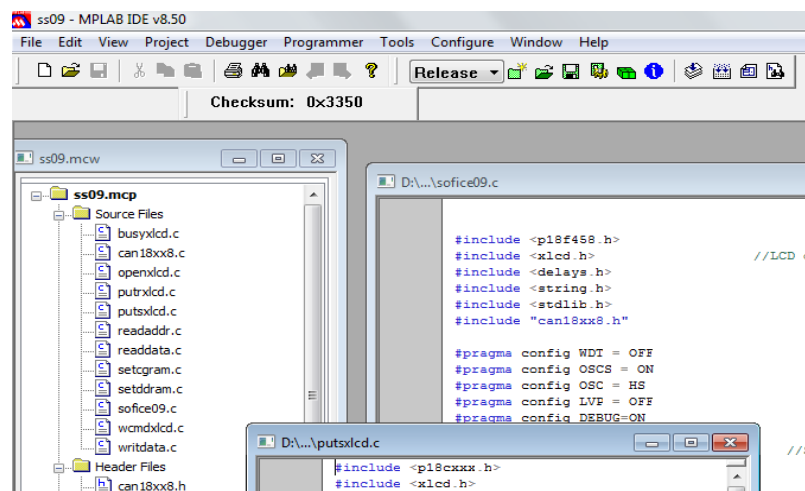


Figure 3.1 MPLAB IDE

3.3 PICKit 3

Microchip technology created a group of programmers, among of them is PICKit as shown in 3.2. They are utilized to program and debug Microcontrollers, additionally program EEPROM. Some models likewise feature logic analyzer and serial communications (UART) instrument, figure 3.3 displays the pin diagram of PICKit 3. The general population who create open-source programming for the PICKit utilize a mailing list for collaboration [23].



Figure 3.2 PICKit 3 package

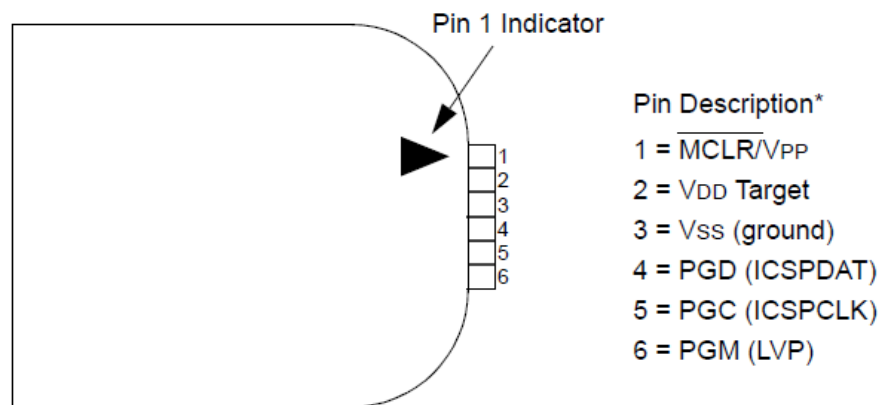


Figure 3.3 PICKit 3 Programmer Connector pinout

3.4 programming steps in Microcontroller

Generally, Programming of PIC Microcontroller includes the following stages

1. Write the code
2. Compile the code
3. Upload the code into a Microcontroller, as shown in figure 3.4.

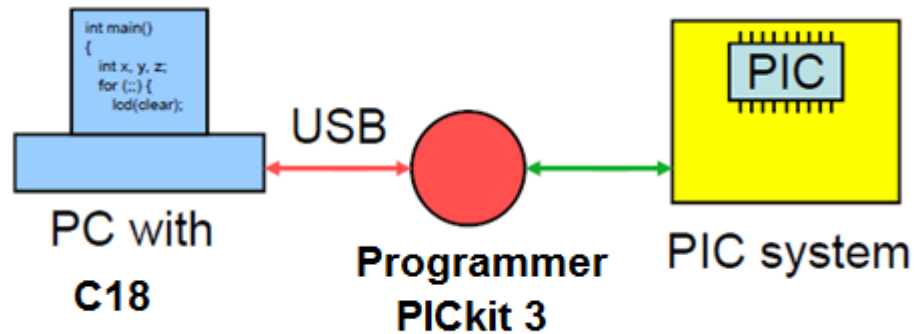


Figure 3.4 PIC Microcontroller programming

3.5 C programming versus Assembly

- If compare C language to another language, it is easy for transporting when this language transferred from one processor to next one. Also, no need more for changing.
- C programming in high-level language rather than assembler, also developing of C language more quickly than assembler.
- Typically, a program which takes a couple of weeks in assembler agent Can be composed in C in a couple days.

Also, C is not as efficient as the assembly because the assembly programmer can usually do better than the compiler, no matter what the optimal level – C will use more memory.

3.6 PIC compilers

A compiler converts a high-level language program to machine instructions for the target processor. Likewise, a compiler that keeps running on a processor (for the most part a PC), that is not quite the same as the objective processor. Additionally,

the C18 compiler is a free program for understudies utilized for programming the PIC as a part of C Language [24].

3.7 Downloading MPLAB C18 C Compiler

After installing MPLAB IDE from Microchip website on the PC, once in Microchip's site, choose to download this file: MPLAB C for PIC18 v3.46

- Execute file. Click to accept the license agreement. Next
- This window in figure 3.5 shows the path where the compiler will be installed and then next.

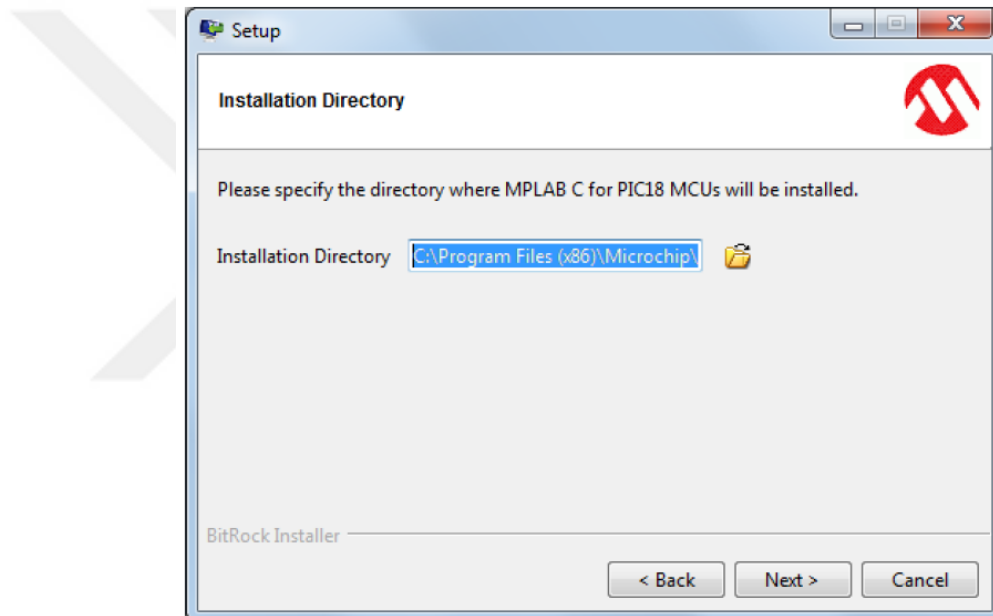


Figure 3.5 choosing C programming download location

- Continue installation. When the last window appears, click "Finish" as shown in figure 3.6.

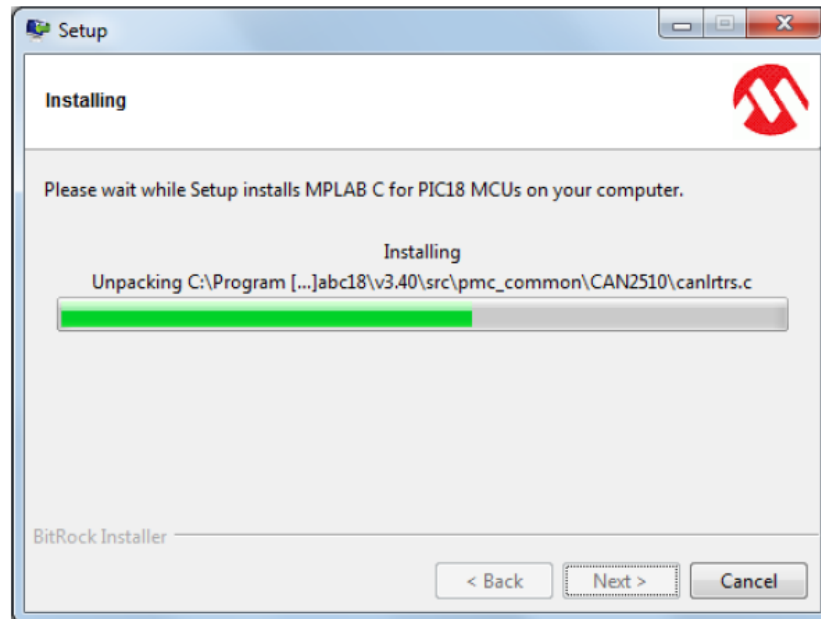


Figure 3.6 C downloading guide

The installation program generates four applications

1. mcc18.exe (compiler).
2. mplink.exe (linker).
3. mplib.exe (library management).
4. MPASMWIN.EXE (assembler).

These four applications must be included by the user in any C18 project.

3.8 MPLAB IDE Create Project wizard choosing PIC18F458 with C language

- Projects are groups of files associated with language tools.
 - A project consists of source files, header files, object files, library files and a linker script.
 - At minimum, one header file is required to identify the register names of the objective microcontroller.
 - At least one header file is required to identify the register names of the target microcontroller.
 - The project's output files consist of executable code to be loaded into the objective microcontroller.
1. Write source text(.c) into this new file.

2. Open File Go to save this file. Browse by specific location as you want, as shown in figure 3.7.
3. Click Save.

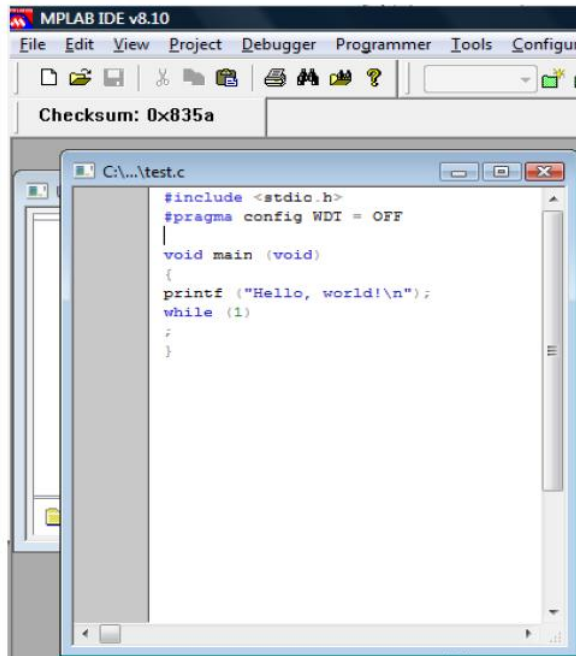


Figure 3.7 MPLAB IDE with the main program

For creating project wizard must do the following:

4. To create a new project, go to the project then Project Wizard, displays in figure 3.8.

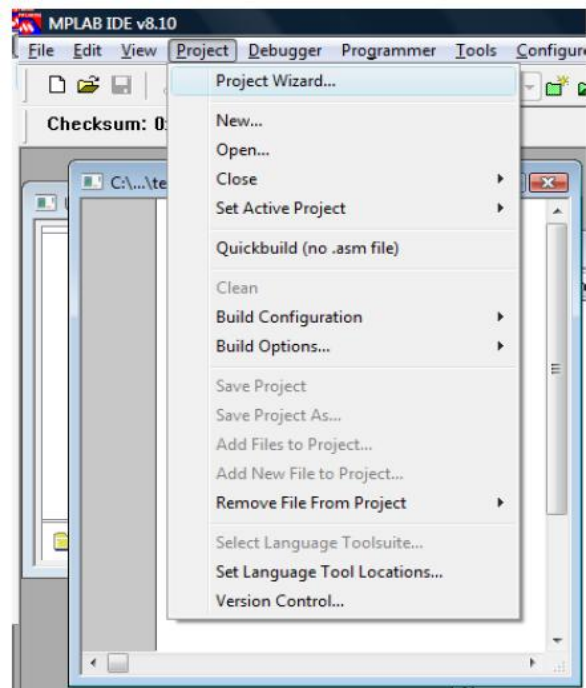


Figure 3.8 MPLAB IDE Creating C project

5. Choose device (specific PIC), In this project Choose PIC18F458 as shown in figure 3.9.

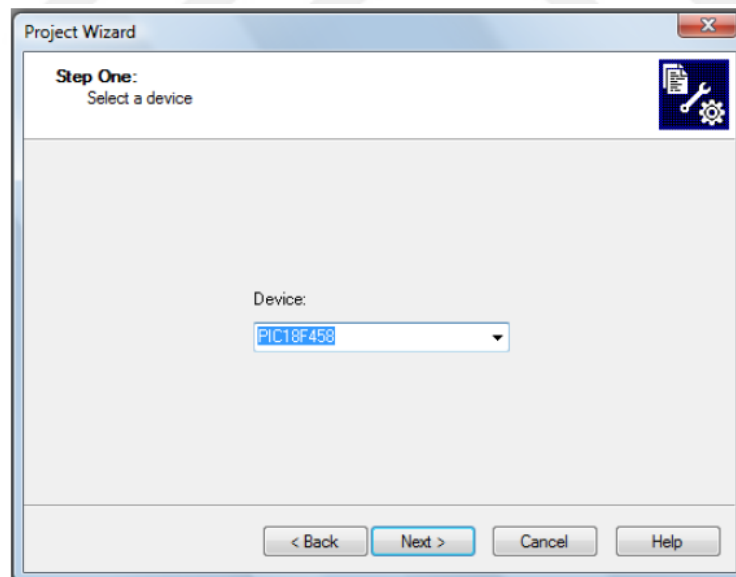


Figure 3.9 MPLAB IDE choosing specific PIC

6. Select the language toolsuite. “Microchip C18 Toolsuite” as the “Active Toolsuite”.

7. As shown in figure 3.10, Click on each language tool in the toolsuite (under “Toolsuite Contents”) and check or set up its associated executable location.

MPASM Assembler should point to the assembler executable, MPASMWIN.exe, under “Location”. If it does not, enter or browse to the executable location, which is by default: C:\mcc18\mpasm\MPASMWIN.exe.

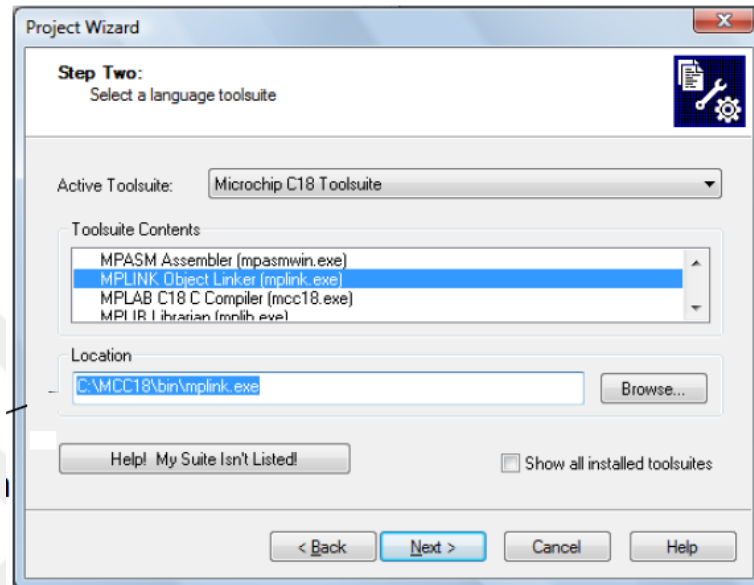


Figure 3.10 Choosing executable location

3.9 MPLAB C18 C COMPILER LIBRARIES

For my project used MPLAB C18 C language for configuring the following

1. Delay Function.
2. PIC18F458 external LCD function.
3. PIC18F458 ADC.
4. PIC18F458 CAN system.

3.9.1 XLCD DELAY FUNCTIONS

The XLCD libraries additionally require that the accompanying functions characterized by the user to give the proper defers:

Table 3.1 LCD delay function

Function	Behavior
DelayFor18TCY	Delay for 18 cycles.
DelayPORXLCD	Delay for 15 ms.
DelayXLCD	Delay for 5 ms.

3.9.2 External LCD functions

1. BusyXLCD

Used for is the LCD is busy?

Table 3.2 LCD Busy function

Discription name	function
Include	xlcd.h (header file)
Prototype	unsigned char BusyXLCD(void);//function declaration
File Name	busyxlcd.c
Code Example	while(BusyXLCD());

2. OpenXLCD

This function is used to Configure the PIC port that connected to the LCD and initializes it.

Table 3.3 Open LCD function

Discription name	function
Include	xlcd.h (header file)
Prototype	void OpenXLCD(unsigned char <i>lcdtype</i>); lcdtype example: FOUR_BIT for (4-bit Data Interface mode) EIGHT_BIT for (8-bit Data Interface mode) OR LINE_5X7 for (5x7 characters, single line display) LINE_5X10 for (5x10 characters display) LINES_5X7 for (5x7 characters, multiple line display)
File Name	openxlcd.c
Code Example	OpenXLCD(EIGHT_BIT & LINES_5X7);

3. putsXLCD & putrsXLCD

This function is used to writing a string to the LCD controller.

Table 3.4 put string to the LCD

Discription name	function
Include	xlcd.h (header file)
Prototype	void putsXLCD(char * <i>buffer</i>); void putsXLCD(const rom char * <i>buffer</i>);
File Name	putsxlcd.c putrxlcd.c
Code Example	char mybuff [20]; putsXLCD("Hello World"); putsXLCD(mybuff);

4. ReadAddrXLCD

This function is used to read the address of (CGRAM or DDRAM) depended the previous function. Also, before performing this function, the controller should not busy (BusyXLCD must be used).

Table 3.5 function reads the address byte from the Hitachi HD44780 LCD controller

Discription name	function
Include	xlcd.h (header file)
Prototype	unsigned char ReadAddrXLCD(void);
File Name	readaddr.c
Code Example	char addr; while (BusyXLCD()); addr = ReadAddrXLCD();

5. Read a data byte from the LCD controller

This function is used to read the data LCD controller. Before performing this function, the controller should not busy (BusyXLCD must be used).

Table 3.6 Reading Data from the LCD

Discription name	function
Include	xlcd.h (header file)
Prototype	char ReadDataXLCD(void);
File Name	readdata.c
Code Example	char data; while (BusyXLCD()); data = ReadAddrXLCD();

6. Set the character generator address

This function sets the character generator address of the LCD controller. Before performing this function, the controller should not busy (BusyXLCD must be used).

Table 3.7 Character Generator Address function

Discription name	function
Include	xlcd.h (header file)
Prototype	void SetCGRamAddr(unsigned char addr); addr is the character generator address.
File Name	setcgram.c
Code Example	char cgaddr = 0x1F; while(BusyXLCD()); SetCGRamAddr(cgaddr);

7. Set the display data address

This function sets the display data address of the LCD controller. Before performing this function, the controller should not busy (BusyXLCD must be used).

Table 3.8 Display Data Address Function

Discription name	function
Include	xlcd.h (header file)
Prototype	void SetDDRamAddr(unsigned char addr); addr is Display data address.
File Name	setddram.c
Code Example	char ddaddr = 0x10; while(BusyXLCD()); SetDDRamAddr(ddaddr);

8. Write a command to the LCD controller

Table 3.9 Writing Command to the LCD

Discription name	function
Include	xlcd.h (header file)
Prototype	void WriteCmdXLCD(unsigned char cmd); cmd is the following DOFF Turn display off CURSOR_OFF Enable display with no cursor BLINK_ON Enable display with blinking cursor BLINK_OFF Enable display with unblinking cursor SHIFT_CUR_LEFT Cursor shifts to the left SHIFT_CUR_RIGHT Cursor shifts to the right SHIFT_DISP_LEFT Display shifts to the left SHIFT_DISP_RIGHT Display shifts to the right
File Name	setddram.c
Code Example	while(BusyXLCD()); WriteCmdXLCD(EIGHT_BIT & LINES_5X7); WriteCmdXLCD(BLINK_ON); WriteCmdXLCD(SHIFT_DISP_LEFT);

9. Writes a byte to the Hitachi HD44780 LCD controller

This function is used to write a data byte of (CGRAM or DDRAM) depended on previous function. Also, before performing this function, the controller should not busy (BusyXLCD must be used).

Table 3.9 Writing Command to the LCD

Discription name	function
Include	xlcd.h (header file)
Prototype	void WriteDataXLCD(char data);
File Name	writdata.c

4.9.3 PIC18F458 ADC C configuration

There are five main function use for converting ADC:

1. Is the A/D converter at present performing a conversion (BusyADC)?

This function shows if the A/D peripheral is in the process of converting a value.

Table 3.10 ADC Busy function

Discription name	function
Include	adc.h (header file)
Prototype	char BusyADC(void);
File Name	adcbusy.c
Code Example	while(BusyADC());

1. Disable the A/D converter (CloseADC)

This function disables the A/D converter and A/D interrupt technique.

Table 3.11 Close the ADC function

Discription name	function
Include	adc.h (header file)
Prototype	void CloseADC(void);
File Name	adcclose.c

2. Starts the A/D conversion procedure (ConvertADC)

This function starts an A/D conversion. Before performing this function, The BusyADC () function may be used.

Table 3.12 Start Conversion Function of ADC

Discription name	function
Include	adc.h (header file)
Prototype	void ConvertADC(void);
File Name	adconv.c

3. Open ADC for PIC18F458

Table 3.13 General Configuration of ADC function

Discription name	function
Include	adc.h (header file)
Prototype	void OpenADC(unsigned char config , unsigned char config2);
Config	A/D clock source: ADC_FOSC_2 Fosc / 2 ADC_FOSC_4 Fosc / 4 ADC_FOSC_8 Fosc / 8 ADC_FOSC_16 Fosc / 16 ADC_FOSC_32 Fosc / 32 ADC_FOSC_64 Fosc / 64 ADC_FOSC_RC Internal RC Oscillator A/D result justification: ADC_RIGHT_JUST Result in Least Significant bits ADC_LEFT_JUST Result in Most Significant bits A/D voltage reference source: ADC_8ANA_0REF VREF+=VDD, VREF-=VSS, All analog channels ADC_7ANA_1REF AN3=VREF+, All analog channels except AN3 ADC_6ANA_2REF AN3=VREF+, AN2=VREF ADC_6ANA_0REF VREF+=VDD, VREF-=VSS

4. Read the result of an A/D conversion(ReadADC)

Using this function dedicated to reading the result of 10 bit of analog to digital conversion. In light of the configuration of the A/D converter (e.g., using the OpenADC () function), the result will be contained in the Least Significant or Most Significant bits of the 10-bit result.

Table 3.14 Reading Data function of ADC

Discription name	function
Include	adc.h (header file)
Prototype	int ReadADC(void);
File Name	adcread.c

5. This function is used for selecting channels to be as analog input.

Table 3.15 Channel Selection of ADC Function

Discription name	function
Include	adc.h (header file)
Prototype	void SetChanADC (unsigned char <i>channel</i>);
channel	<p><i>channel</i> One of the following values (defined in adc.h): ADC_CH0 Channel 0 ADC_CH1 Channel 1 ADC_CH2 Channel 2 ADC_CH3 Channel 3 ADC_CH4 Channel 4 ADC_CH5 Channel 5 ADC_CH6 Channel 6 ADC_CH7 Channel 7 ADC_CH8 Channel 8 ADC_CH9 Channel 9 ADC_CH10 Channel 10 ADC_CH11 Channel 11</p>
File Name	adcsetch.c
Code Example	SetChanADC (ADC_CH0);

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 Introduction

This system is made up of four CAN nodes. One node (called DISPLAY node) requests the temperature periodically and displays it on an LCD. This process is then repeated continuously. The second, third and fourth node reads the temperature of 24 refrigerators from an external semiconductor temperature sensor. This will be a linear active sensor that converts temperature to an analog voltage. Also, each node is far from together in a different place inside Blood Bank as shown in figure 4.1. The chosen Microcontroller will be the cost Efficient is seeing that the controller consists of an already included CAN module [25].

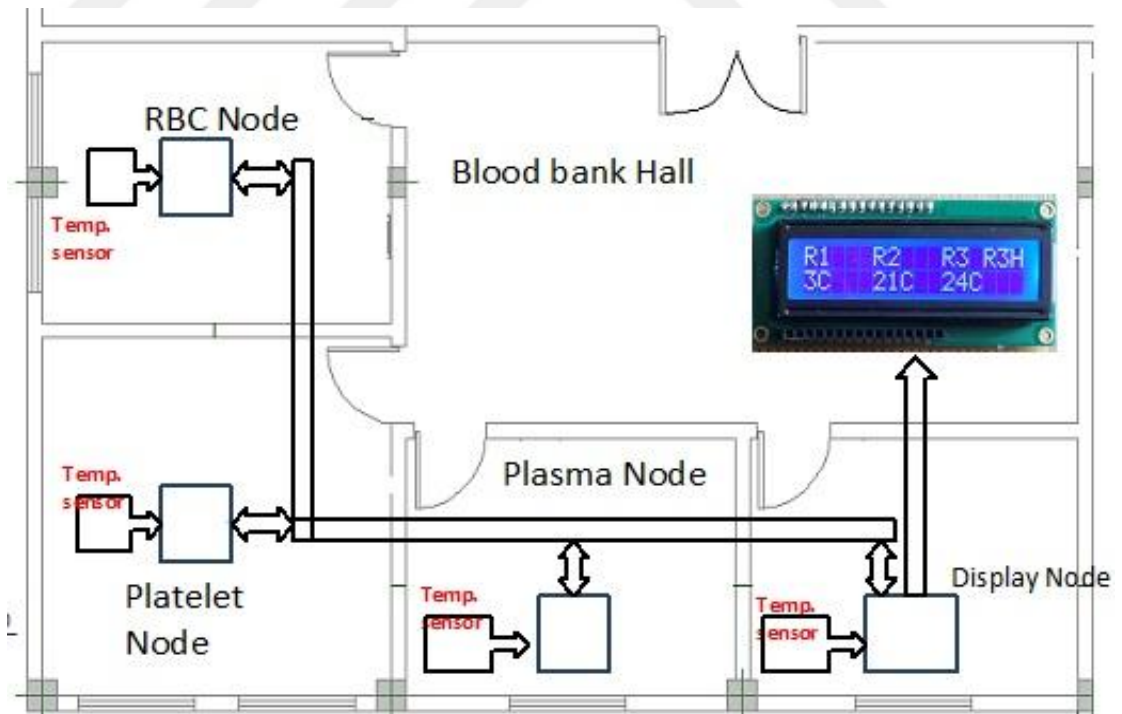


Figure 4.1 Blood Bank Center designed layout

4.2 System Design

The system is designed such that each node will send and receive messages across a network. Given I chose ECS Inc's 4 MHz Oscillator (ECS) our nominal Baud Rate will be 25Kb/s. Nodes will be networked together such that each node will perform some action given the voltage output of a temperature sensor. The temperature sensor of choice is the LM35DZ which provides us a10mV per degree Centigrade output with no calibration necessary. By using this sensor to read the output signal into PORTA, pin RA1of the one of the PIC18F458 [24]. Controller's Analog to Digital Converter. A final PIC18F458 microcontroller will be used to display the temperature in C°.

4.2.1 BIT Timing Parameters

The CAN Module Bit Timing is made up of non-overlapping segments. Each of these segments is made up of integer units called Time Quanta (TQ). The Nominal Bit Rate (NBR) is defined as the number of bits per second transmitted by an ideal transmitter with no resynchronization [26].

$$NBR = 1/T_{BIT} \quad (4.1)$$

Where ; T_{BIT} = Nominal Bit Time

NBR = Nominal Bit Rate

$$T_{BIT} = T_Q \times (\text{Sync_Seg} + \text{Prop_Seg} + \text{Phase_Seg1} + \text{Phase_Seg2}) \quad (4.2)$$

Where; Sync_Seg= Synchronization Segment

Prop_Seg = Propagation Time Segment

Phase_Seg1= Phase Buffer Segment 1

Phase_Seg2=Phase Buffer Segment

$$T_Q = 2x(BRP + 1)xT_{osc} \quad (4.3)$$

Where; BRP=Baud Rate Prescaler

$$F_{osc} = 4 \text{ MHz}$$

$$T_{osc} = \frac{1}{4} MHz \rightarrow = 0.25 \mu s$$

$$NBR = 25 Kbit/s$$

$$T_{BIT} = \frac{1}{25 Kbit/s} \rightarrow = 40 \mu s$$

IF

$$BRP = 4$$

$$T_Q = 2x(BRP + 1)xT_{osc} = 2x(4 + 1)x0.25\mu s$$

$$T_Q = 2x(4 + 1)x0.25\mu s \rightarrow = 2.5\mu s$$

SO on equation (4.2)

$$40\mu s = 2.5\mu s \times (\text{Sync Seg} + \text{Prop Seg} + \text{Phase Seg1} + \text{Phase Seg2})$$

$$40\mu s / 2.5\mu s = (\text{Sync Seg} + \text{Prop Seg} + \text{Phase Seg1} + \text{Phase Seg2})$$

$$16 = (\text{Sync Seg} + \text{Prop Seg} + \text{Phase Seg1} + \text{Phase Seg2})$$

Nodes must have same Nominal Bit Rate (number of bits per second)

- The Nominal Bit Time is defined as: $T_{BIT} = 1/\text{Nominal Bit Rate}$
- Synchronization Jump Width (SJW): 1-4
- Baud Rate Prescaler (BRP): 1-64
- Propagation Time Segment (Prop_Seg) 1 - 8
- Phase Buffer Segment 1 (Phase_Seg1) 1- 8
- Phase Buffer Segment 2 (Phase_Seg2) 1-8

$$\text{Prop Seg} + \text{Phase Seg 1} \geq \text{Phase Seg 2}$$

$$\text{Phase Seg 2} \geq \text{Sync Jump Width}$$

These timing bits of our project were found to be:

- SJW = 1;

- BRP = 4;
- Phase_Seg1 = 4;
- Phase_Seg2 = 3;
- Prop_Seg = 3;

$$\text{Prop_Seg} + \text{Phase_Seg 1} \geq \text{Phase_Seg 2}$$

$$3 + 4 \geq 3 \quad (\text{True})$$

$$\text{Phase Seg 2} \geq \text{Sync Jump Width}$$

$$3 \geq 1 \quad (\text{True})$$

4.2.2 The Implemented Algorithms

This project is a four node CAN bus project, which are Display, Plasma, platelet, and RBC node as shown in figure 4.1.

4.2.2.1 Display node

The display node is consisting of a PIC18f458, a 4 MHz Oscillator, a transceiver MCP2561, the input of 7-channel Darlington Sink Driver ULN2003APG is connected to the PORTE of PIC to turn on and off Buzzer and LED, also an LCD is connected to the PIC via port D. All the components interface with each other and work based on a specified algorithm. Using C-Language that is compiled on MPLAB v.92 we have programmed the PIC. The PIC18f458 has one build in CAN module and CAN controller. This facilitates the CAN project because there is an internal interface between the CAN module and the CAN controller. Figure 4.2 shows the circuit diagram of the Node.

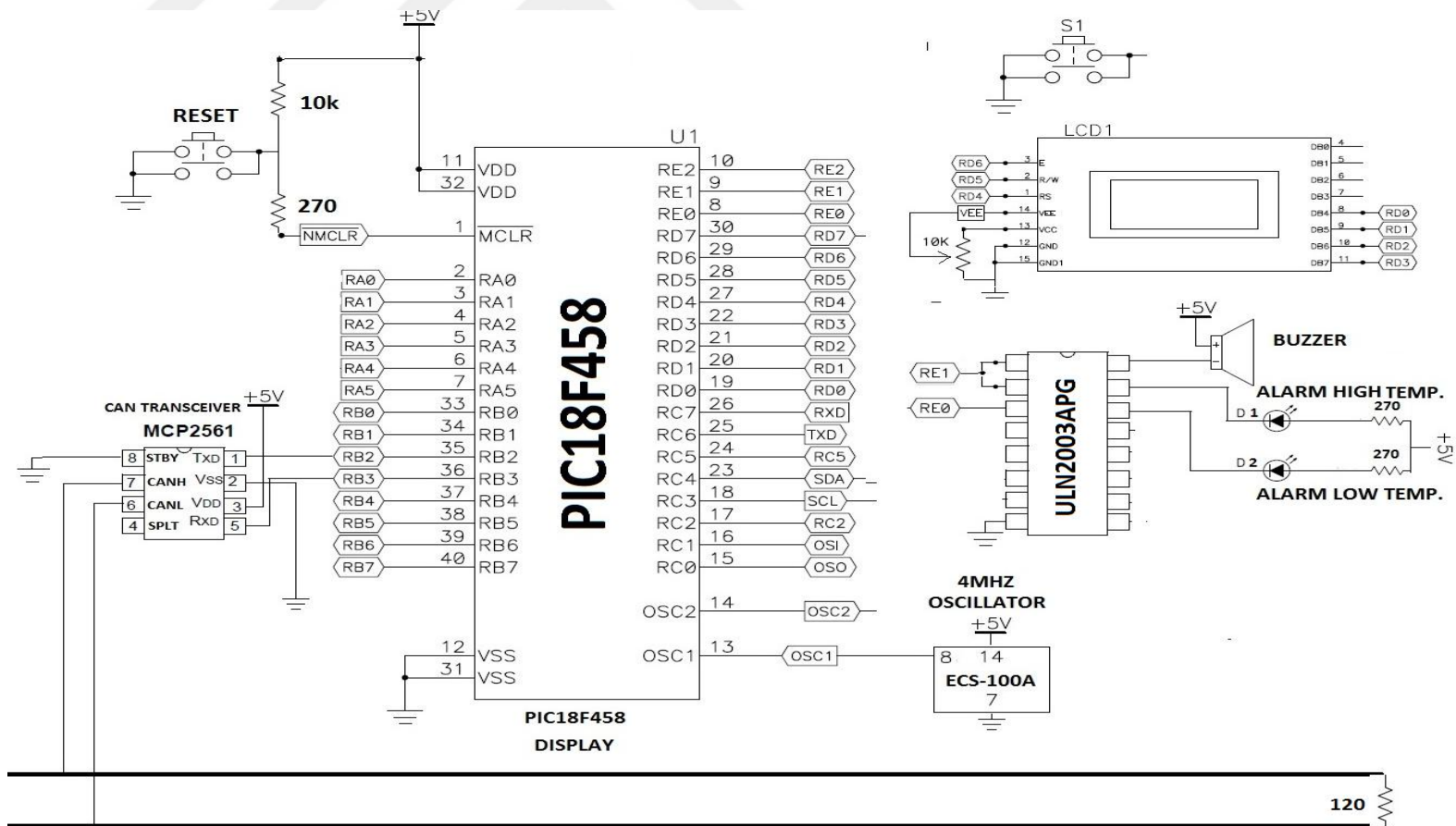


Figure 4.2 Display Node Schematic Diagram

4.2.2.1.1 Configuration CAN Module

The first thing must be done is initializing the CAN module. Using the specified CAN initializing parameter and written code functions in C, the CAN module is initialized. Then it should be put the CAN module in the CONFIG mode and must be set some parameters in the masks and filters. We assign the filter B2-F3 for this node and set the band pass as 6 for RBC, 7 for platelet and 8 for plasma node. That means this CAN module passes any messages with the identification of 6 during setting filter to 6 and so on. Then the Module is put back to NORMAL mode, which means that it is ready to work.

4.2.2.1.2 Implementing the Node feature

Initializing the LCD so as to be ready to get the data and show it. After initializing the LCD, the node should send a message to the bus that contains character “T”, as a symbol of temperature, with a specified identifier, for example, 65 in our case. Reading the messages on the bus, this module passes all messages that have 6 during filter setting as 6 identifiers. The messages with ID=6 contain the data of temperature that has been sent by another node, which is RBC (Red Blood Cell) node in our case and so on. This node sends the character “T” to the bus, reads the temperature data on the bus every second, and shows the Temperature in a suitable position on the LCD continuously which is the updated temperature from the Temperature Node.

All the necessary procedures to achieve the display node algorithm is shown in figure 4.3.

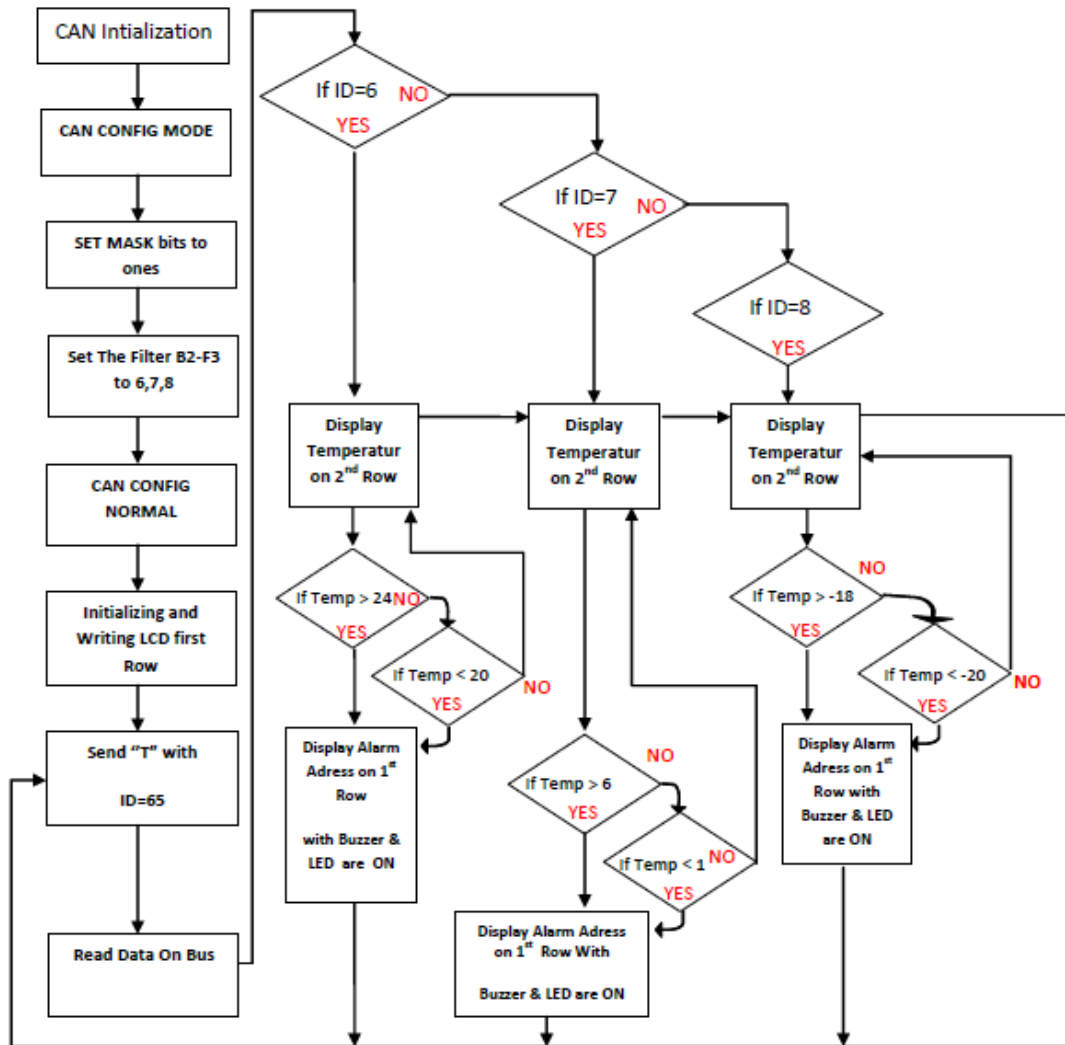


Figure 4.3 Display Node Schematic Diagram

4.2.2.2 Red Blood Cell (RBC), Platelet and Plasma Node

This node has the following components: a PIC18f458, an LM35 temperature sensor, a 4 MHz Oscillator, and an MCP2561 transceiver. The temperature sensor is connected to the (ADC) which is a module inside the PIC through Port A (RA1). The structure of the node has illustrated the figure 4.4.

These three nodes have the following components: a PIC18f458, an LM35 temperature sensor, a 4 MHz Oscillator, and an MCP2561 transceiver. The temperature sensor is ADC, which is a module inside the PIC through Port A (RA1). The structure of all three nodes has the same components, but in software are different, especially in the identification of nodes and setting of collecting temperature.

contains the character “T”. If there is a character “T” in the message, the RBC node reads the output of the ADC and builds a CAN packet message that contains the actual temperature data and sends it to the bus with the identifier of 6. Similarly, the platelet and plasma nodes build the messages containing the temperature information and send them to the CAN bus with difference identifier. Means the platelet sends the ID = 7, also, plasma sends ID=8. The reading bus and sending temperature to the bus is a loop procedure that occurs every second. Figure 4.5 shows the flowchart that explains the algorithm of the RBC, platelet, and plasma node.

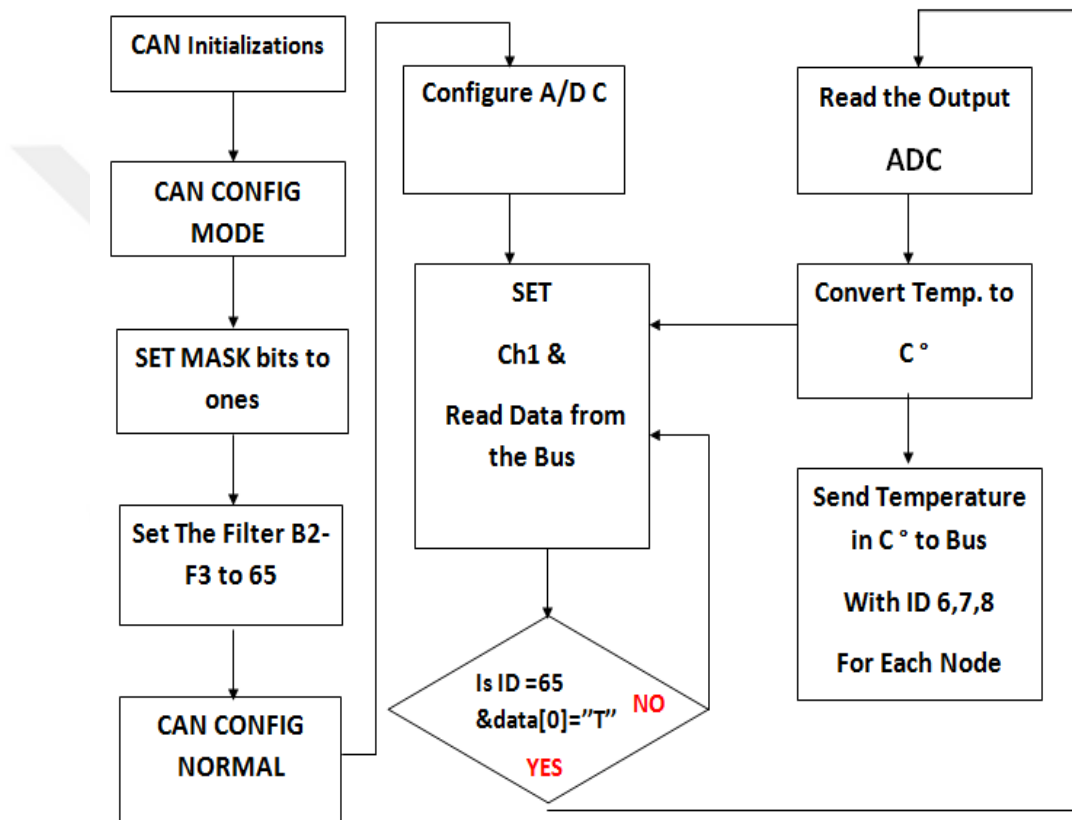


Figure 4.5 Flow Chart for CAN Based RBC, Platelet, and Plasma Node

4.2.2.3 Hardware structure

As it is shown in figure 4. all the four nodes have been connected to the CAN bus. The interfaces between all the nodes and the CAN bus are done successfully. In my project, practically each node has been connected through one temperature sensor represented as 8 refrigerators, nevertheless PIC18F458 has 8 analog channels, it's capable of connecting 8 refrigerators until send data to the Bus of each of them respectively. Actually my project consists of four parts of the boards (nodes),

Display, RBC, plasma, and platelet boards, each of them located in different places in inside blood bank center but I simulated in one board just for easy testing and transporting purpose. Also, this simulated board is more rugged than four parts of the board. As shown in Figure 4.7. The all four nodes simulated Wiring diagram as shown in figure4.6.

As long as we power up the system, it continuously monitors the temperature and shows it on the LCD. This temperature is compared with the value stored by the user and if the temperature goes beyond the preset temperature, then buzzer and a red LED will switch on. If the temperature goes below to preset value, then white LED will switch on. In these cases, the address of each refrigerator will send to the LCD display. Figure - shows the temperature of three refrigerators connected with three nodes and the alarm address (R3H) indicated the high temperature of the refrigerator of plasma node.

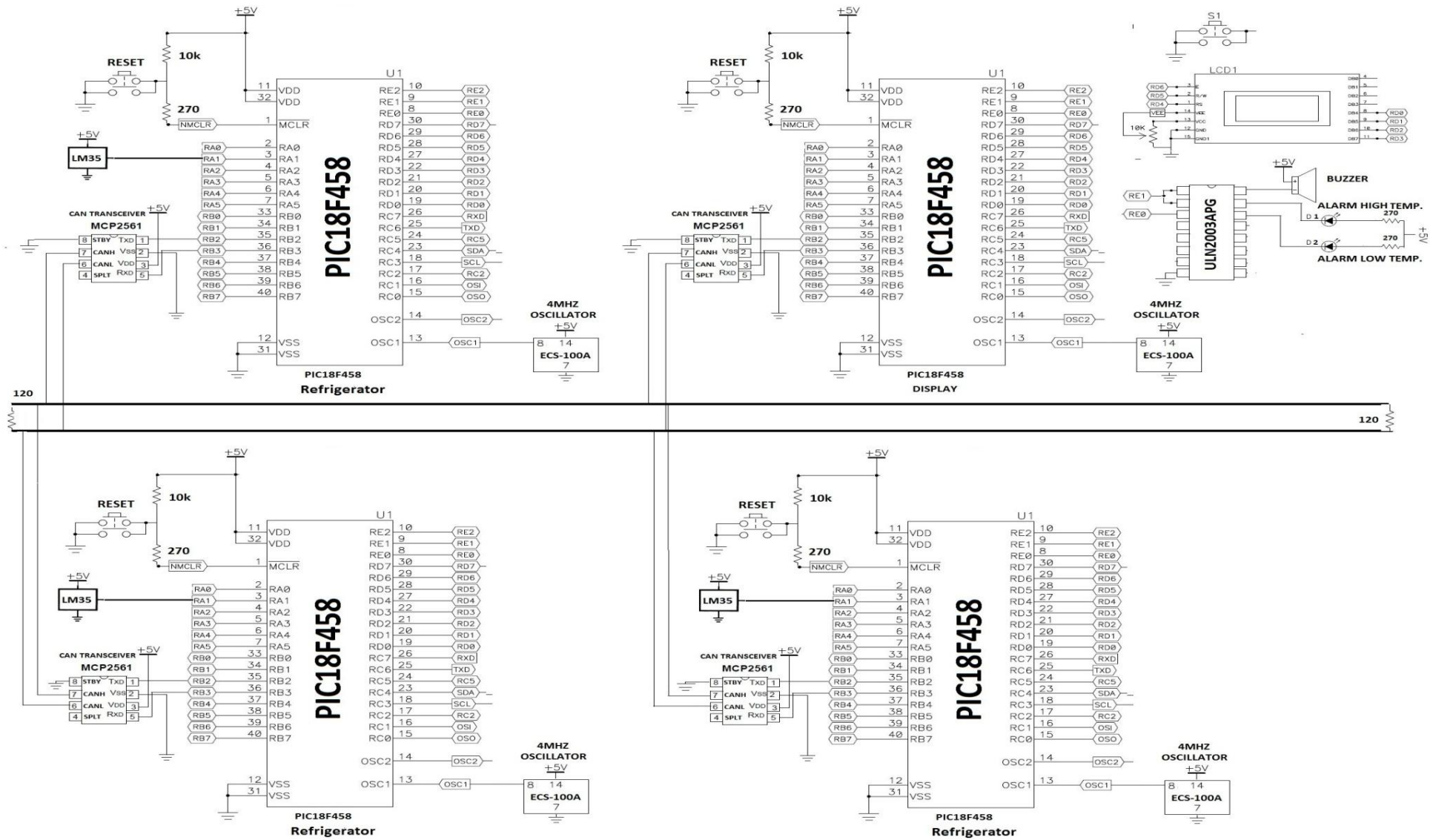


Figure 4.6 Four node circuit diagram of the four node monitoring system based CAN system

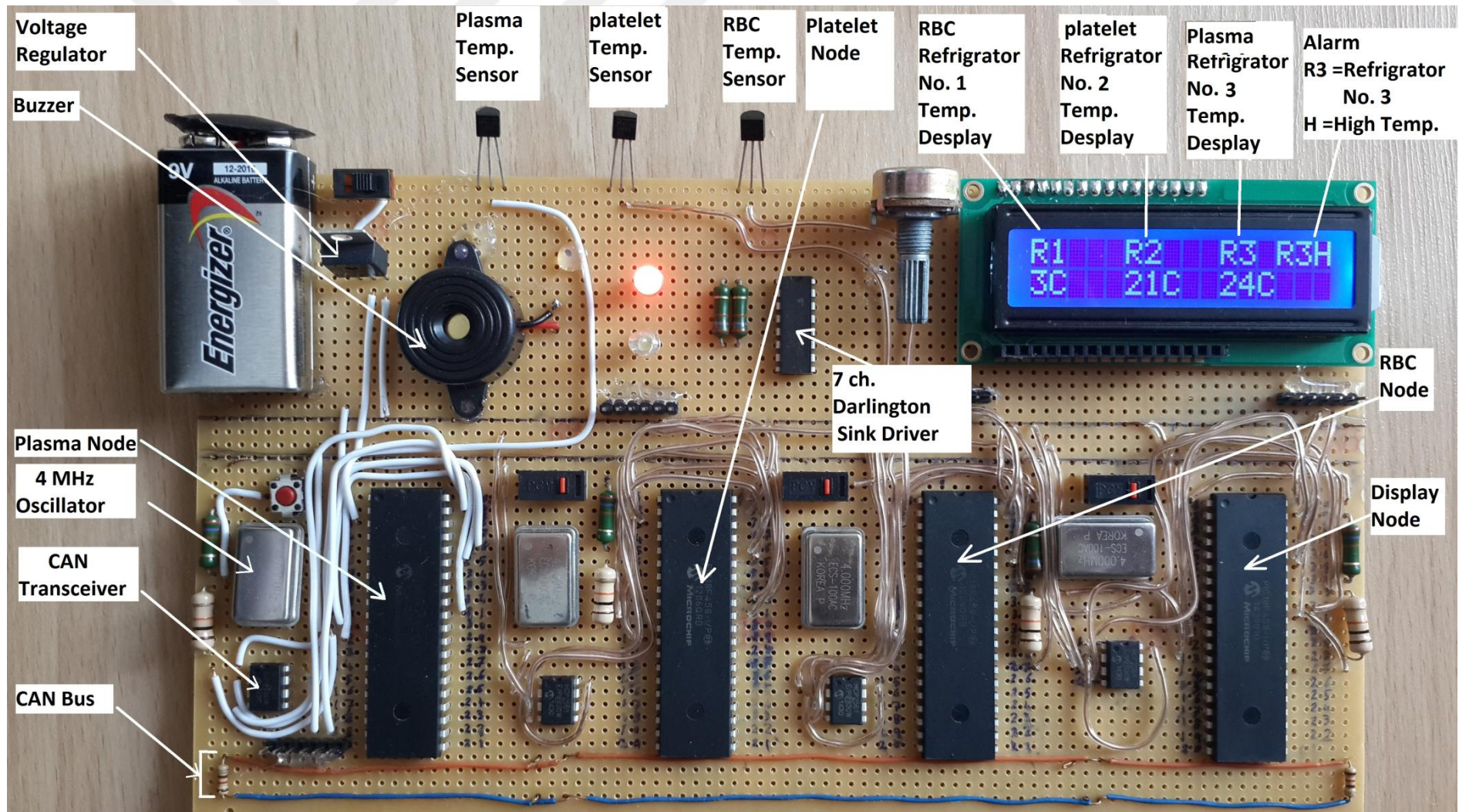


Figure 4.7 The Actual hardware structure of the project system.

CHAPTER 5

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the LCD display and laboratory instruments have been used to carry out measurements and test the result of all nodes. Also, the result of monitoring of temperature obtained with the actual temperature is compared.

4.2

1. Figure 5.1 shows the result of the designed system after collecting and running the system. The LCD node that sends the data to the bus, collecting data on the bus and then shows the three nodes Temperature in Celsius in the 2nd row of the LCD. While showing the name of each Refrigerator and the alarm address on the first row of the LCD. The temperature shown on the LCD changes based on the data, which is generated and sent to the bus by the RBC, platelet and plasma node. The temperature sensor continuously reads the temperature and the (RBC, platelet, and plasma) nodes generate the CAN packet data and put it on the bus every second. Therefore, a temperature that is shown on the LCD gets updated every second.

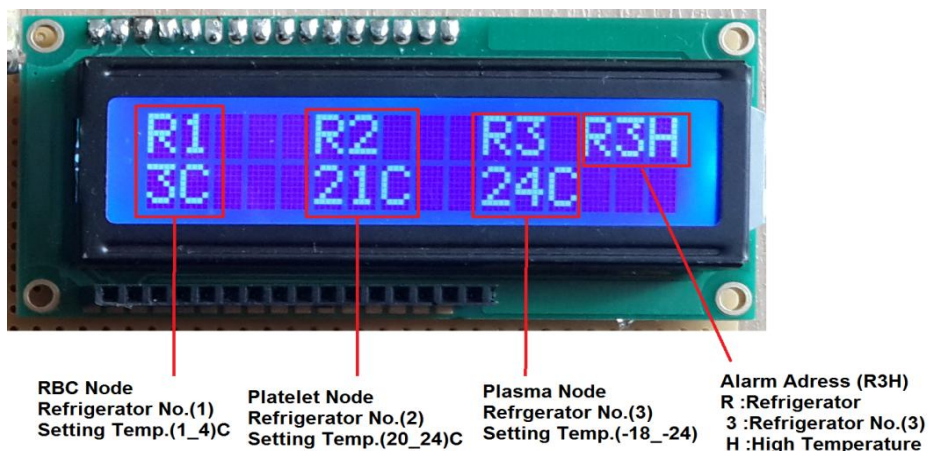


Figure 5.1 displays three different temperatures in three different nodes and alarm address

Table 5.1 Practical Result of Implemented Board for monitoring Three Different temperatures in Three Nodes and Reaction of LED s and Buzzer

Node sensor	Temp.Setting (°C)		Override Temp.	Red LED	Blue LED	Alarm address
	High	Low				
RBC	High	24	> 24	ON	OFF	R1H (Refrigerator 1 high temp.)
	Low	20	< 20	OFF	ON	R1L (Refrigerator 1 low temp.)
Platelet	High	4	> 4	ON	OFF	R2H (Refrigerator 2 high temp.)
	Low	1	< 1	OFF	ON	R2L (Refrigerator 2 low temp.)
Plasma	High	-24	> -24	ON	OFF	R3H (Refrigerator 3 high temp.)
	Low	-18	< -18	OFF	ON	R3L (Refrigerator 3 low temp.)

1. Table - shows the result of this project as follows;

The alarm address on the LCD, Buzzer, and LED works properly according to the designed algorithm.

(a) RBC node:

This refrigerator set's the temperature from minimum 20 °C and maximum 24 °C ,when the temperature decreased below 20 °C the buzzer turned on, white LED turned on (indicated low-temperature alarm), Also the alarm address (R1L) appeared in the first row of the LCD display(indicated low-temperature of refrigerator number one), when the temperature increased beyond the 24 °C the buzzer and Red LED turned on (indicated high-temperature alarm), also the alarm address R1H will appear in the first row (indicated the high-temperature of Refrigerator number one). When the temperature remained between 20 °C to 24 °C (normal state), All alarm didn't appear.

(b) Platelet Node:

This refrigerator set's the temperature from 1 °C to 6 °C in the same way of RBC node, when the temperature is overriding the buzzer and LED turned on, Also the alarm address R2L, R2H indicated to low, high temperature, respectively, this alarm appeared on the first row on the display, When the temperature remained between 1 °C to 6 °C (normal state), all alarm didn't appear, the result shown in table 5.1.

(c) Plasma Node

This refrigerator set's the temperature from -18 °C to -24 °C in the same way of RBC and platelet node, when the temperature overrides the buzzer and LED turned on, Also the alarm address R3L, R3H indicated to low, high temperature, respectively, this alarm appeared on the first row on the display, When the temperature remained between -18 °C to -24 °C (normal state), all alarm didn't appear, the result is shown in table 5.1 , Also figure 5.1 shows the temperature of this refrigerator is 24 °C in the same time alarm address showed R3H indicated high temperature of refrigerator number 3.

2. We can see the data flow on the bus using an oscilloscope as it is Illustrated in figure 5.2. Also, We can see the data as a pulse train on both CAN high and CAN low. The upper part which is in yellow is CAN high while the lower part which is in blue color is the data on CAN low. Notice that there is the same data on CAN high and low part, except they are negative to each other.



Figure 5.2 The Actual hardware structure of the project system

3. Although the system works perfectly, there are some noises appears on the pulses. The reason of those noises is the imperfection of the wires and components that are used in the system, for example, the bus is a normal twisted wire instead of a shielded twisted wire bus. Also, most of the components are cheaper versions that I bought to implement the CAN bus system, if I use the most advanced system which is more expensive, we will see less noise on the pulse trains.



CHAPTER 6

CONCLUSION AND SUGGESTIONS

6.1 Conclusion

The four nodes CAN bus system designed and implemented successfully and the data exchanges between all the four nodes. The PIC18f458 is used in each of the four nodes. The PIC18f458 that used in display node interfaced with an LM35 temperature sensor and the CAN bus. The node is read the temperature from the sensor and build a CAN package data and then put it on the can bus as well as the node read the data on the bus successfully. Another node has been designed in this project that connected to an LCD and has sent the data to show on the LCD successfully. The LCD (display) node could send and read the data on the bus perfectly, also it turned on and off the LED and Buzzer successfully in case the temperature is greater or less than the limited range. We conclude that one of the most important and sensitive parts in CAN system at the baud rate and timing parameters. It's necessary to follow the CAN data sheet and all the component data sheets, which has been used in the design, to build the correct timing parameters and consequently get the accurate interface between the PICs that have been used in the CAN bus system.

6.2 Suggestions for Future Work

Monitoring of the temperature related to many factors for people's life for example; industry, medicine, military, agricultural, automotive, for all these uses the temperatures for a specific purpose, Also the future work of this work can extend to design:

1. This system can be developed for central monitoring station in the Intensive Care Unit (ICU) department inside the hospital, In this case, must increase some analog to digital sensors like pressure sensor for measuring the blood pressure, special

oxygen sensor for measuring O₂ inside the human's blood, Electrocardiogram sensor (ECG or EKG) for sensing the heart pulses and so on.

2. In future, the circuit can be enhanced by connecting a GSM or a wireless module for increasing the screen monitor in many places.

3. Using the monitoring and controlled the temperature in industry factories like an asphalt factory for collecting of temperature of the mixer, tar, dryer system, Filter system, Weighbridges system, heating fuel and controlling the variety of electrical valves and motors. Also, Wireless controller area network (WCAN) can be used for communication between sensors and actuators.



REFERENCES

- [1] Psychology. (n.d.). In Wikipedia. Retrieved October (2009). from <http://en.wikipedia.org/wiki/psychology>.
- [2] J Anaesth (2014). In NCBI website. Retrieved from <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4260297>
- [3] Mishra, M. (2013). Department of Electronics & Communication Engineering (Doctoral dissertation, National Institute of Technology, Rourkela).
- [4] Microcontroller. (n.d.). In Wikipedia. Retrieved from <https://en.wikipedia.org/wiki/Microcontroller#Interrupts>.
- [5] Microcontroller. (n.d.). Retrieved from <http://internetofthingsagenda.techtarget.com/definition/microcontroller>.
- [6] Microchip Technology. (n.d.). Retrieved from URL: http://www.worldlibrary.org/articles/microchip_technology website of World Public Library.
- [7] Mazidi, M. A., Mazidi, J. G., & McKinlay, R. D. (2000). The 8051 microcontroller and embedded systems. New Delhi.
- [8] Verle, M. (2009). PIC Microcontrollers Programming in C: A Complete Guide to Pic Microcontrollers. MikroElektronika.
- [9] Sheet, N. D. (2000). LM35CZ: Precision Centigrade Temperature Sensor. National Semiconductor Corporation, USA (November 2000).
- [10] Steve Corrigan (2008). Introduction to the Controller Area Network, Published by Texas Instruments Application Report, SLOA101A.
- [11] Singh, M. N. (2006). Designing a Microcontroller Based Temperature Data Logger.
- [12] Singh, M. N. (2006). Designing a Microcontroller Based Temperature Data Logger.
- [13] Microchip technolog Inc. (2013). High-Speed CAN Transceiver, MCP2561/2, Device Document DS25167B.
- [14] Kundariya, R. C., Agravat, M. N., Vekariya, D. M., Bhatti, C. C., & Bambhroliya, C. B. (2016). Electronic Grocery Machine. *International Journal for Innovative Research in Science and Technology*, **2(9)**, 140-145.
- [15] Presi, T. P. (2013). Design and development Of PIC microcontroller based vehicle monitoring system using Controller Area Network (CAN) protocol. In Information Communication and Embedded Systems (ICICES), 2013 *International Conference on (pp. 1070-1076). IEEE.*

- [16] Sekhar, N. C., Reddy, T. S., & Bhavani, G. (2014). Implementation of low-cost MEMS based temperature measurement and control system using Lab VIEW and microcontroller. In Power, Control and Embedded Systems (ICPCES), 2014 *International Conference on* (pp. 1-4). *IEEE*.
- [17] Badri, M. A., & Halim, A. K. (2008, December). Design of moving message LCD display system (MMDS) via Short Message Service (SMS) entry using Rabbit 2000 microcontroller. In RF and Microwave Conference, 2008. RFM 2008. *IEEE International* (pp. 81-85). *IEEE*.
- [18] Mazidi, M. A., Mazidi, J. G., & McKinlay, R. D. (2000). The 8051 microcontroller and embedded systems. New Delhi.
- [19] S.J.Perez, M.A.Calva, R.Castañeda(2014).Microcontroller Based Automatic Multichannel Temperature Monitoring System *IJAREEIE*.
- [20] Kaiwan S,Ergun E.,Majed N., et al. (2015). Development of Monitoring Blood bank Center Based PIC Microcontroller Using CAN communication. *International .Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering* **Vol:9**, No:12.
- [21] MPLAB. (n.d.). In Wikipedia. Retrieved from <https://en.wikipedia.org/wiki/MPLAB>.
- [22] PICkit. (n.d.). In Wikipedia. Retrieved from <http://en.wikipedia.org/wiki/PICkit>.
- [23] PICkit. (n.d.). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/PICkit#cite_note-1.
- [24] Farahmand (2012) Programming the PIC18 Using C- Coding.
- [25] Kaiwan S,Ergun E.,Majed N., et al. (2015). Development of Monitoring Blood bank Center Based PIC Microcontroller Using CAN communication. *International .Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering* **Vol:9**, No:12.
- [26] Richards, P. (2001). Understanding Microchip's CAN Module Bit Timing. Application Note AN754, Microchip Technology Inc.



APPENDIX - A

Display Node code in C language of PIC18F458 for receiving CAN message from the bus and send it to the display. Also includes the temperature setting of each refrigerator :

```

#include <p18f458.h>

#include <xlcd.h> //LCD display, modified header to set pins

#include <delays.h>

#include <string.h>

#include <stdlib.h>

#include "can18xx8.h"

#pragma config WDT = OFF

#pragma config OSCS = ON

#pragma config OSC = HS

#pragma config LVP = OFF

#pragma config DEBUG=ON

//int count; //Stores the number of hits of the Pushbutton

char Celc[10] = "C ";

char Odata[10]; //Store old value of the pushbutton

unsigned long id, mask;

unsigned char temperature, data[8];

//time delays for LCD

void DelayFor18TCY(void);

void DelayPORXLCD(void);

void DelayXLCD(void);

void Delay1(void);

//write to LCD function

void writeLCD (char buffer[]);

void Disp_sec(void);

//time delay functions

void DelayPORXLCD(void) //provides at least 15ms delay

{

    Delay1KTCYx (15); //delay of 15*1000 microseconds

```

```

}

void DelayXLCD(void)           //provides at least 5ms delay
{
    Delay1KTCYx (5);           //delay of 5*1000 microseconds
}

void Delay1(void)             //provides at least 000ms delay
{
    Delay1KTCYx (1000); //delay of 1000*1000 microseconds= 1 S
}

//write function for LCD
void writeLCD (char buffer[])
{
    TRISDbits.TRISD7 = 0;           //VEE = 1, turn on lcd
    LATDbits.LATD7 = 1;
    DelayPORXLCD();
    DelayPORXLCD();
    while(BusyXLCD());           // Wait if LCD busy
    WriteCmdXLCD(0x01);           // Clear display
    DelayPORXLCD();               //time for LCD to be ready
    putsXLCD (" ");
    putsXLCD( buffer );           //prints the text
    putsXLCD (" ");
    DelayPORXLCD();               //time for LCD to be ready
}

void Disp_sec(void)
{ if(id == 6)
{
    SetCGRamAddr( 0x40 );
    SetDDRamAddr( 0x40 );
    WriteCmdXLCD(0xc0);
}
}

```

```

        putsXLCD(Odata);

        putsXLCD(Celc);
    }
else if(id == 7)
{
SetCGRamAddr( 0x40 );

        SetDDRamAddr( 0x40 );
WriteCmdXLCD(0xc5);

        putsXLCD(Odata);

        putsXLCD(Celc);
}

else if(id == 5)
{
SetCGRamAddr( 0x40 );

        SetDDRamAddr( 0x40 );
WriteCmdXLCD(0xcA);

        putsXLCD(Odata);

        putsXLCD(Celc);
}
}

//time delay functions for the LCD

void DelayFor18TCY(void) //provides a 18 Tcy delay
{
    Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(), Nop(),Nop(),Nop();

    Nop(),Nop(),Nop(), Nop();
}

void DelayFor10TCY(void) //provides a 10 Tcy delay
{
    Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop();
}

```

```

}

//-----

void main()

{

int i;

unsigned char temperature, data[8];

unsigned short init_flag, dt;

BYTE SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg;

unsigned char len;

enum CAN_RX_MSG_FLAGS read_flag;

enum CAN_TX_MSG_FLAGS send_flag;

unsigned int j;

TRISE = 0x00;      //Set PORTE as an Output

PORTD = 0x00;      //Clear PORTD

TRISD = 0x00;      //Set PORTD as an Output

PORTE = 0x00;      //Clear PORTE

TRISB = 0x08;      // (0000 1000)RB2 is output, RB3 is input

// CAN BUS Parameters

//SJW = 1;

//BRP = 4;

//Phase_Seg1 = 4;

//Phase_Seg2 = 3;

//Prop_Seg = 3;

SJW = 1;

BRP = 4;

Phase_Seg1 = 4;

Phase_Seg2 = 3;

Prop_Seg = 3;

init_flag = CAN_CONFIG_SAMPLE_THRICE &

                CAN_CONFIG_PHSEG2_PRG_ON &

```

```

        CAN_CONFIG_STD_MSG &

        CAN_CONFIG_DBL_BUFFER_ON &

        CAN_CONFIG_ALL_VALID_MSG &

        CAN_CONFIG_LINE_FILTER_OFF;

send_flag = CAN_TX_PRIORITY_0 &

        CAN_TX_XTD_FRAME &

        CAN_TX_NO_RTR_FRAME;

read_flag = 0;

// Initialize CAN module

CANInitialize(SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg, init_flag);

// Set CAN CONFIG mode

//CANSetOperationMode(CAN_OP_MODE_CONFIG);

//mask = -1;

mask = 0xFFFFFFFF;

// Set all MASK1 bits to 1's

CANSetMask(CAN_MASK_B1, mask, CAN_CONFIG_STD_MSG);

// Set all MASK2 bits to 1's

CANSetMask(CAN_MASK_B2, mask, CAN_CONFIG_STD_MSG);

// Set id of filter B2_F3 to 6

CANSetFilter(CAN_FILTER_B2_F3, 7, CAN_CONFIG_STD_MSG);

// Set CAN module to NORMAL mode

//CANSetOperationMode(CAN_MODE_NORMAL, 0xFF);

CANSetOperationMode(CAN_OP_MODE_NORMAL);

// Configure LCD

// initialization for the LCD starts here

LATDbits.LATD7 = 1;

DelayPORXLCD();

DelayPORXLCD();

OpenXLCD( FOUR_BIT & LINES_5X7 );           //initialize LCD 4 bit mode

while(BusyXLCD());

```

```

WriteCmdXLCD(0x80);      //display on the LCD (2x16 character)from first Line & first colum
Delay1();

putsXLCD (" University of ");

while(BusyXLCD());

WriteCmdXLCD(0xC3);      //display on the LCD (2x16 character)start from second Line & 4 colum
putsXLCD ("Gaziantep");

while(BusyXLCD());

Delay1();

Delay1();

WriteCmdXLCD(DON&CURSOR_OFF&BLINK_OFF);

Delay1(),Delay1(),Delay1(),Delay1(),Delay1(),Delay1(),Delay1(),Delay1(),Delay1(),Delay1();

for(l=0;l<15;l++)      //counter loop 15 times
{
WriteCmdXLCD(0x1c);    //shift display to right side
Delay1KTCYx (400);
}

while(BusyXLCD());

WriteCmdXLCD(0x01);

Delay1();

// Program loop. Read the temperature from Node:plasma ,platelet,RBC and display
// on the LCD continuously

for(;;) // Endless loop
{

data[0] = 'T'; // Data to be sent

id = 65; // Identifier

CANSendMessage(id, data, 1, send_flag); // send 'T'

// Get temperature from node:plasma ,platelet,RBC

dt = 0;

```



```

while(!dt)

dt=CANReceiveMessage(&id, data, &len, &read_flag);

if(id == 6) // id 6 is RBC node id

{

temperature = data[0];

itoa(temperature,Odata); // convert integer data to charecter

if(temperature > 4)

{

while(BusyXLCD());

WriteCmdXLCD(0x8D);

putsXLCD("R1H"); // R :Refrigerator , 1:Refr. number ,H: High temperature

PORTEbits.RE1 = 1; // Turn on portE1(red LED & Buzzer)

}

else {

while(BusyXLCD());

WriteCmdXLCD(0x80); //display on the LCD (2x16 character)from first Line & first colum

putsXLCD("R1 ");

while(BusyXLCD());

WriteCmdXLCD(0x8A); //display on the LCD (2x16 character)start from second Line & 10 colum

putsXLCD("R3 ");

PORTEbits.RE1 = 0; // Turn off portE1(red LED & Buzzer)

}

Disp_sec(); //Display the value of count on the second line of the LCD

}

Delay1KTCYx (1000);

if(id == 7) // id 7 is plstelet node id

{

temperature = data[0];

itoa(temperature,Odata);

if(temperature > 24)

```

```

{
while(BusyXLCD());

WriteCmdXLCD(0x8D); //display on the LCD (2x16 character)from first Line & colum 13

putsXLCD("R2H");

PORTEbits.RE1 = 1; // Turn on portE1(red LED & Buzzer)

}

else {

while(BusyXLCD());

WriteCmdXLCD(0x85);

putsXLCD("R2 ");

while(BusyXLCD());

WriteCmdXLCD(0x8A);

putsXLCD("R3 ");

PORTEbits.RE1 = 0; // Turn off portE1(red LED & Buzzer)

}

Disp_sec();//Display the value of count on the second line of the LCD

}

Delay1KTCYx (1000);

if(id == 5) // id 5 is plasma node id

{

temperature = data[0];

itoa(temperature,Odata); // convert integer data to charecter data

if(temperature > -24)

{

while(BusyXLCD());

WriteCmdXLCD(0x8D); //display on the LCD (2x16 character)from first Line & colum 13

putsXLCD("R3H");

PORTEbits.RE1 = 1; // Turn on portE1(red LED & Buzzer)

}

else {

```

```
while(BusyXLCD());  
WriteCmdXLCD(0x8A);  
putsXLCD("R3 ");  
PORTEbits.RE1 = 0; // Turn on portE1(red LED & Buzzer)  
}  
Disp_sec();  
}  
}  
}
```



APPENDIX-B

RBC,Platelet,Plasma Node code in C language of PIC18F458 for receiving data from Temperature sensor and sending to the bus in CAN message , The same program for each three nod jut in different ID, for example RBC ID is 7, Platelet ID is 8, Also Plasma ID is 5:

```
#include <p18f458.h>

#include <xlcd.h> //LCD display, modified header to set pins

#include <delays.h>

#include <string.h>

#include <stdlib.h>

#include <adc.h>

#include "can18xx8.h"

#pragma config WDT = OFF

#pragma config OSCS = ON

#pragma config OSC = HS

#pragma config LVP = OFF

#pragma config DEBUG=ON

char Odata[10];    //Store old value of the pushbutton

char Ndata[10];    //Store new value of the Pushbutton

char result_disp [1];

double output = 0;

double temp;

//time delays for LCD

    void DelayFor18TCY(void);

    void DelayPORXLCD(void);

    void DelayXLCD(void);

    void Delay1(void);

//write to LCD function

    void writeLCD (char buffer[]);

    void Disp_sec(void);

//time delay functions
```

```

void DelayPORXLCD(void)           //provides at least 15ms delay
{
    Delay1KTCYx (15);           //delay of 15*1000 microseconds
}

void DelayXLCD(void)             //provides at least 5ms delay
{
    Delay1KTCYx (5);           //delay of 5*1000 microseconds
}

void Delay1(void)                //provides at least 000ms delay
{
    Delay1KTCYx (1000);        //delay of 1000*1000 microseconds= 1 S
}

//-----
//time delay functions for the LCD

void DelayFor18TCY(void) //provides a 18 Tcy delay
{
    Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop();
    Nop(),Nop(),Nop(),Nop(),Nop(),Nop();
}

void DelayFor10TCY(void) //provides a 10 Tcy delay
{
    Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop(),Nop();
}

//-----

void main()
{
    unsigned char temperature, data[8];
    unsigned short init_flag, dt;
    BYTE SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg;
    unsigned long id, mask;

```

```

unsigned char len;

enum CAN_RX_MSG_FLAGS read_flag;

enum CAN_TX_MSG_FLAGS send_flag;

//unsigned int j;

TRISE = 0x00;    //Set PORTE as an Output
PORTD = 0x00;    //Clear PORTD
TRISD = 0x00;    //Set PORTD as an Output
PORTE = 0x00;    //Clear PORTD

    //configure ADC

    OpenADC( ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_1ANA_0REF,
            ADC_CH1 & ADC_INT_OFF );

    Delay10TCYx( 4 ); // Delay for 40TCY
TRISB = 0x08; // RB2 is output, RB3 is input
// CAN BUS Parameters

SJW = 1;
BRP = 4;
Phase_Seg1 = 4;
Phase_Seg2 = 3;
Prop_Seg = 3;

init_flag = CAN_CONFIG_SAMPLE_THRICE &
            CAN_CONFIG_PHSEG2_PRG_ON &
            CAN_CONFIG_STD_MSG &
            CAN_CONFIG_DBL_BUFFER_ON &
            CAN_CONFIG_ALL_VALID_MSG &
            CAN_CONFIG_LINE_FILTER_OFF;

send_flag = CAN_TX_PRIORITY_0 &
            CAN_TX_STD_FRAME &
            CAN_TX_NO_RTR_FRAME;

read_flag = 0;

// Initialize CAN module

```

```

CANInitialize(SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg, init_flag);

// Set CAN CONFIG mode

CANSetOperationMode(CAN_OP_MODE_CONFIG);

//mask = -1;

mask = 0xFFFFFFFF;

CANSetMask(CAN_MASK_B1, mask, CAN_CONFIG_STD_MSG); // Set all MASK1 bits to 1's
CANSetMask(CAN_MASK_B2, mask, CAN_CONFIG_STD_MSG); // Set all MASK2 bits to 1's
CANSetFilter(CAN_FILTER_B2_F3,65,CAN_CONFIG_STD_MSG); // Set id of filter B2_F3 to 65
CANSetOperationMode(CAN_OP_MODE_NORMAL); // Set CAN module to NORMAL mode

// Program loop. Read the temperature from Node:RBC,Platelet,plsma
// on the LCD continuously
for(;;) // Endless loop
{
    dt=0;
    while(!dt)
    dt=CANReceiveMessage(&id, data, &len, &read_flag);

    if(id==65 && data[0]=="T")
    {
        ConvertADC(); // Start conversion

        while( BusyADC() ); // Wait for completion

        output = ReadADC(); // Read result

        temp=(output*500)/1023; //FORMULA TO change to TEMP

        data[0]= temp;

        id = 7; // Identifier for RBC ID=7,Platelet ID=8,Plasma ID=5

        CANSendMessage(id, data, 1, send_flag); // send 'T'

        PORTE = 0xFF;
    }
}

```

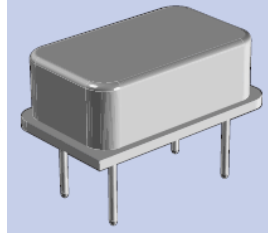



APPENDIX - C

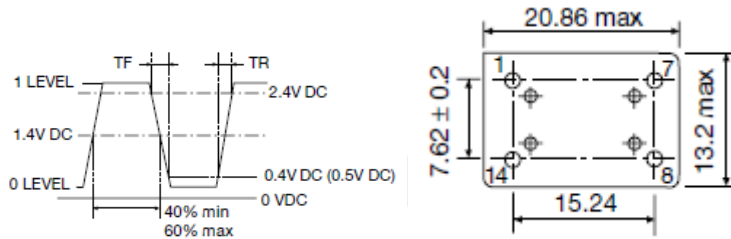
ECS-100 Oscillator :

FEATURES

- 10 TTL output load
- Low cost
- Wide frequency range
- Industry standard footprint
- Resistance weld package
- 3.3V operation (optional)



PIN CONNECTIONS	
#1	NC
#7	CASE GND
#8	OUTPUT
#14	+5 V DC



PARAMETERS	FREQUENCY RANGE	CONDITIONS	MINIMUM	TYPICAL	MAXIMUM	UNITS
FREQUENCY RANGE (f_0)	1.000 ~ 150.000		1.000		150.000	MHz
OPERATING TEMP. RANGE (T_{OPR})	1.000 ~ 150.000		0		+70	°C
STORAGE TEMP. RANGE (T_{STG})	1.000 ~ 150.000		-55		+125	°C
FREQUENCY STABILITY	1.000 ~ 150.000	All conditions*	-100		+100	PPM
INPUT CURRENT (I_{OO})	1.000 ~ 7.999	max. load			15	mA
	8.000 ~ 23.999	max. load			30	mA
	24.000 ~ 69.999	max. load			70	mA
	70.000 ~ 150.000	max. load			80	mA
OUTPUT SYMMETRY	1.000 ~ 7.999	1.4V level	45	50 ±3	55	%
	8.000 ~ 150.000	1.4V level	40	50 ±3	60	%
RISE TIME (T_R)	1.000 ~ 24.999	0.4V ~ 2.4V			10	nS
	25.000 ~ 69.999	0.5V ~ 2.4V			5	nS
	70.000 ~ 150.000	0.5V ~ 2.4V			4	nS
FALL TIME (T_F)	1.000 ~ 24.999	2.4V ~ 0.4V			10	nS
	25.000 ~ 69.999	2.4V ~ 0.5V			5	nS
	70.000 ~ 150.000	2.4V ~ 0.5V			4	nS
OUTPUT VOLTAGE	1.000 ~ 24.999	$I_{OL} = 20$ mA			0.4	V
	25.000 ~ 150.000	$I_{OL} = 20$ mA			0.5	V
	70.000 ~ 150.000	$I_{OH} = 1$ mA	2.4			V
OUTPUT CURRENT	1.000 ~ 150.000	$V_{OL} = 0.5$ V			20	mA
	1.000 ~ 150.000	$V_{OH} = 2.4$ V			1.0	mA
OUTPUT LOAD	1.000 ~ 150.000				10	TTL
START-UP TIME (T_s)	1.000 ~ 3.499				20	mS
	3.500 ~ 3.999				35	mS
	4.000 ~ 5.999				30	mS
	6.000 ~ 19.999				20	mS
	20.000 ~ 150.000				15	mS
SUPPLY VOLTAGE	1.000 ~ 150.000	+5.0 ±0.25			-	V _{DC}



APPENDIX - D

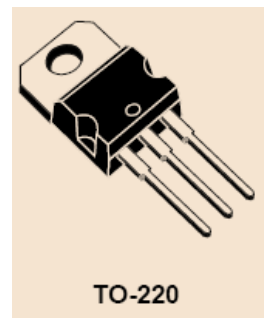
POSITIVE VOLTAGE REGULATORS

- output current to 1.5a output voltages of 5; 5.2; 6; 8; 8.5; 9; 10; 12; 15; 18; 24v
- thermal overload protection
- short circuit protection
- output transition SOA protection

Description

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-220FM, TO-3 and D2PAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

Electrical Characteristics Of L7805 (refer to the test circuits, $T_J = -55$ to 150°C , $V_I = 10\text{V}$, $I_O = 500\text{ mA}$, $C_I = 0.33\ \mu\text{F}$, $C_O = 0.1\ \mu\text{F}$ unless otherwise specified).





APPENDIX-D

PIC18F458 :

High-Performance RISC CPU:

- Linear program memory addressing up to 2 Mbytes
- Linear data memory addressing to 4 Kbytes
- Up to 10 MIPS operation
- DC – 40 MHz clock input
- 4 MHz-10 MHz oscillator/clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier

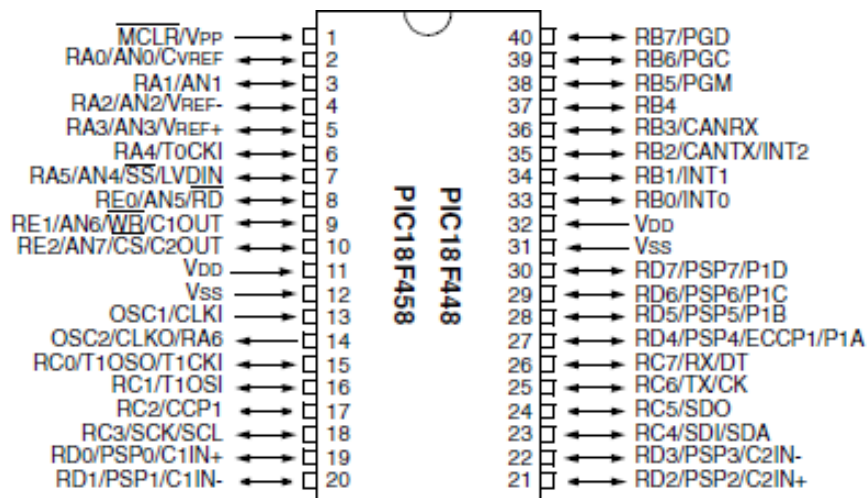
Advanced Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter module (A/D) with:

- Conversion available during Sleep
- Up to 8 channels available
- Analog Comparator module:
- Programmable input and output multiplexing
- Comparator Voltage Reference module
- Programmable Low-Voltage Detection (LVD) module:
- Supports interrupt-on-Low-Voltage Detection
- Programmable Brown-out Reset (BOR)

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	Comparators	CCP/ ECCP (PWM)	MSSP		USART	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I ² C™		
PIC18F248	16K	8192	768	256	22	5	—	1/0	Y	Y	Y	1/3
PIC18F258	32K	16384	1536	256	22	5	—	1/0	Y	Y	Y	1/3
PIC18F448	16K	8192	768	256	33	8	2	1/1	Y	Y	Y	1/3
PIC18F458	32K	16384	1536	256	33	8	2	1/1	Y	Y	Y	1/3

PIC18F458 specification



PIC18F458 pin

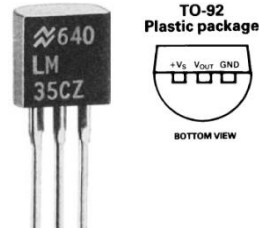


APPENDIX-E

LM35 Temperature sensor

Features

- Output proportional to °C
- Wide temperature range -40°C to +110°C (CZ version)
- Accurate 1/4°C at room temperature typical
- Linear output 0.2°C typical
- Low current drain (60µA typical)
- Low self heating (0.08°C typical)
- Output impedance 0.1Ω at 1mA
- Standard T092 package.



LM35 temperature sensor

Absolute maximum ratings (Note 10)

Supply voltage _____ +35V to -0.2V
 Output voltage _____ +6V to -1.0V
 Output current _____ 10mA
 Storage temperature, TO-92 package _-60°C to +150°C
 Lead temperature (soldering, 10 seconds) ____260°C
 Specified operating temperature range

Electrical characteristics (Note 1) (Note 6)

Parameter	Conditions	LM35CZ, LM35DZ			Units (Max.)	
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)		
Accuracy, LM35, LM35C (Note 7)	$T_A = +25^\circ\text{C}$	±0.4	±1.0		°C	
	$T_A = -10^\circ\text{C}$	±0.5			°C	
	$T_A = T_{MAX}$	±0.8			°C	
	$T_A = T_{MIN}$	±0.8			°C	
Accuracy, LM35D (Note 7)	$T_A = +25^\circ\text{C}$	±0.6	±1.5		°C	
	$T_A = T_{MAX}$	±0.9			°C	
	$T_A = T_{MIN}$	±0.9			°C	
Non linearity (Note 8)	$T_{MIN} \leq T_A \leq T_{MAX}$	±0.2		±0.5	°C	
Sensor gain (Average slope)	$T_{MIN} \leq T_A \leq T_{MAX}$	+10.0		+9.8 +10.2	mV/°C	
Load regulation (Note 3) $0 \leq I_L \leq 1\text{mA}$	$T_A = +25^\circ\text{C}$	±0.4	±2.0		mV/mA	
	$T_{MIN} \leq T_A \leq T_{MAX}$	±0.5			mV/mA	
Line regulation (Note 3)	$T_A = +25^\circ\text{C}$	±0.01	±0.1		mV/V	
	$4\text{V} \leq V_S \leq \pm 30\text{V}$	±0.02			mV/V	
Quiescent current (Note 9)	$V_S = +5\text{V}, +25^\circ\text{C}$	56	80		µA	
	$V_S = +5\text{V}$	91	82		138	µA
	$V_S = +30\text{V}, +25^\circ\text{C}$	56.2			141	µA
	$V_S = +30\text{V}$	91.5			µA	
Change of quiescent current (Note 3)	$4\text{V} \leq V_S \leq 30\text{V}, +25^\circ\text{C}$	0.2	2.0		µA	
	$4\text{V} \leq V_S \leq 30\text{V}$	0.5			µA	
Temperature coefficient of quiescent current		+0.39		+0.7	µA/°C	
Minimum temperature for rated accuracy	In circuit of Figure 1, $I_L = 0$	+1.5		+2.0	°C	
Long term stability	$T_1 - T_{MAX}$, for 1000 hours	+0.08			°C	



APPENDIX-F

MCP2561 CAN Transceiver

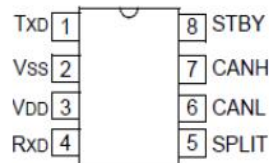
Features:

- Supports 1 Mb/s Operation
- Implements ISO-11898-5 Standard Physical Layer Requirements
- Very Low Standby Current (5 μ A, typical)
- V_{IO} Supply Pin to Interface Directly to CAN Controllers and Microcontrollers with 1.8V to 5.5V I/O
- SPLIT Output Pin to Stabilize Common Mode in Biased Split Termination Schemes
- CAN Bus Pins are Disconnected when Device is Unpowered
- An Unpowered Node or Brown-Out Event will Not Load the CAN Bus
- Detection of Ground Fault:
 - Permanent Dominant Detection on TXD
 - Permanent Dominant Detection on Bus Protection on VDD and V_{IO} Pin
- Power-on Reset and Voltage Brown-Out
- Protection Against Damage Due to Short-Circuit Conditions (Positive or Negative Battery Voltage)
- Protection Against High-Voltage Transients in Automotive Environments
- Automatic Thermal Shutdown Protection
- Suitable for 12V and 24V Systems
- Meets or exceeds stringent automotive design requirements including “Hardware Requirements for LIN, CAN and FlexRay Interfaces in Automotive Applications”, Version 1.3, May 2012
- High-Noise Immunity Due to Differential Bus Implementation
- High ESD Protection on CANH and CANL, Meets IEC61000-4-2 greater \pm 8 kV
- Available in PDIP-8L, SOIC-8L and 3x3 DFN-8L
- Temperature ranges:
 - Extended (E): -40°C to +125°C
 - High (H): -40°C to +150°C



(a)

MCP2561
PDIP, SOIC



(b)