

DETECTION AND PREVENTION OF FAULTY PINS

# DIPLOMA THESIS

Submitted in partial fulfillment of the requirements for the Degree of  
Engineer in the domain Technology and Telecommunication Systems  
Study program: Telecommunications and Information Technologies

**Thesis advisors:**

Prof.univ.Dr.Ing. Corneliu Burileanu

Microchip: Ruseteanu Nicu

**Student:**

Iuliana Roxana

DINU



Annex 1bis

University "Politehnica" of Bucharest  
Faculty of Electronics, Telecommunications and Information Technology  
Department \* TST

Department Director's Approval \*:

Prof. Dr. Ing. Popovici Eduard-Cristian



**DIPLOMA THESIS**  
of student Dinu I. Iuliana Roxana 441G

1. Thesis title: **Detecting and Preventing a Total Power Failure Caused by a Faulty I/O Short circuit to GND**

2. The student's original contribution will consist of (not including the documentation part):  
*describe in 15..20 rows:*

*It will be designed and produced a testing device, which will verify during the software running if a short appears on any pin of the microcontroller.*

*The final device will contain both the test board and an LCD on which it will be displayed the results of the tests.*

*The principal components will be a 8bit microcontroller PIC18F4550 provided by Microchip Technology S.R.L., an LCD, and LEDs.*

3. The project is based on knowledge mainly from the following 3-4 courses: Microprocessor Architecture, Microcontrollers, Project 2 Application with Microcontrollers.

4. The Intellectual Property upon the project belongs to: *UPB, student Iuliana Roxana Dinu and company Microchip Technology S.R.L.*

5. The research is performed at the following location: *company Microchip Technology S.R.L.*

6. The hardware part of the project stays in the property of: *student Iuliana Roxana Dinu*

7. The thesis project was issued at the date: 1<sup>st</sup> October

Thesis Advisor from ETTI (UPB):

STUDENT:

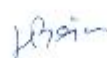
Prof. Cornelia Burileanu, Ph.D.

Iuliana Roxana Dinu



Thesis Advisor from Microchip Technology:

Ing. Horia Boicu





---

## STATEMENT OF ACADEMIC HONESTY

I hereby declare that the thesis "DETECTION AND PREVENTION OF FAULTY PINS", submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of Engineer of Science in the domain Technology and Telecommunication Systems, study program Telecommunications and Information Technologies, is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations and measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations and measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, July 2017

Student: Dinu Iuliana Roxana

  
(Student's signature)



# TABLE OF CONTENTS

Table of Contents .....	7
List of Figures	9
List of Tables	11
List of Acronyms .....	13
CHAPTER 1 Introduction .....	15
1.1 Thesis Motivation .....	15
1.2 Methods Of Detecting Faults .....	16
1.3 Application Description .....	17
CHAPTER 2 Components Description .....	19
2.1 Microcontroller .....	19
2.1.1 Pin Connections .....	21
2.1.2 ADC Module .....	24
2.1.3 UART Module .....	26
2.2 USB Breakout Module .....	30
2.3 LCD Setup .....	32
CHAPTER 3 Software Description .....	35
3.1 MPLAB X Integrated Development Environment .....	35
3.2 XC8 Compiler .....	37
3.3 ICD3 programmer/Debugger .....	37
3.4 Main function .....	38
3.5 Read Potentiometer .....	40
3.6 Detect Fault .....	40
CHAPTER 4 Measurements .....	45
Conclusions	51

Initial Expectations and Final Results .....	51
PERSONAL CONTRIBUTION .....	52
References	53



# LIST OF FIGURES

Figure 1-1.Wire diagnostic circuit (Nisbett, 2012).....	16
Figure 1-2.I/O with A/D function .....	17
Figure 1-3.ICD3 internal pull-ups (Microchip Technology Inc., 2012-1014).....	18
Figure 2-1.Block Diagram of a Microcontroller.....	20
Figure 2-2. PIC18F4550 DIP packaging (Microchip, 2009) .....	20
Figure 2-3.Block diagram for this project.....	21
Figure 2-4.Breadboard circuit with REAL ICE      Figure 2-5.Breadboard circuit with ICD3 ...	22
Figure 2-6.Project Schematics .....	22
Figure 2-7.Home-made PCB .....	23
Figure 2-8.Bottom view PCB .....	23
Figure 2-9.Top view Final PCB.....	23
Figure 2-10.ADC Symbol (George, 2015) .....	24
Figure 2-11.A/D Block Diagram (Verle, 2008).....	24
Figure 2-12.Left / Right Justified (Verle, 2008).....	25
Figure 2-13.I/O port operation (Gottardo, 1999-2013).....	26
Figure 2-14.Transmission Types .....	27
Figure 2-15.EUSART Transmit Block Diagram (Microchip, 2009).....	28
Figure 2-16.EUSART Receive Block Diagram (Microchip, 2009) .....	28

Figure 2-17.MCP2200 USB-to-UART Protocol Converter (MCP2200 USB-to-UART Serial Converter, 2010).....	31
Figure 2-18.Typical usage diagram (Microchip, 2017).....	31
Figure 2-19.RTS/CTS Connections Example (Microchip, 2017) .....	32
Figure 2-20.Simplified schematic for LCD and PIC18F4550.....	33
Figure 3-1.Screen shot with MPLAB X IDE.....	36
Figure 3-2. Breadboard simulation circuit.....	37
Figure 3-3.Correct connection to ICD3 and a simplified internal circuit (Microchip, 2012-2014) .....	38
Figure 3-4.ICD3 and RJ-11 adaptor .....	38
Figure 3-5.Block Diagram for Interrupt .....	39
Figure 3-6.Main Block Diagram.....	40
Figure 3-7.Demo Code Block Diagram.....	40
Figure 3-8.Detection Block Diagram .....	41
Figure 3-9.Fault Values Displayed on LCD .....	43
Figure 4-1.Functional mode, detecting time .....	46
Figure 4-2.Code instructions .....	46
Figure 4-3.GND measurements time .....	47
Figure 4-4.GND detecting time .....	47
Figure 4-5.Open circuit measurements time.....	48
Figure 4-6.Open Circuit detecting time .....	48
Figure 4-7.Vdd measurements time.....	49
Figure 4-8.Vdd detecting time .....	49

# LIST OF TABLES

Table 2-1. Transmit Status and Control (Microchip, 2009).....	29
Table 2-2. Receive Status and Control (Microchip, 2009) .....	29
Table 2-3. Baud Rate Control (Microchip, 2009).....	29
Table 2-4. LCD pins description.....	34
Table 3-1. Values observed during tries .....	42
Table 3-2. Reference values for faults .....	42



# LIST OF ACRONYMS

ADC = Analog to Digital Converter  
ADRESH = Analog-to-Digital Result High  
ADRESL = Analog-to-Digital Result Low  
CPU = Central Processing Unit  
EUSART = Enhanced Universal Synchronous / Asynchronous Receiver Transmitter  
EUSART = Enhanced Universal Synchronous Asynchronous Receiver Transmitter  
I2C = Inter Integrated Circuit  
IC = Integrated Circuit  
ICSP = In Circuit Serial Programming  
IDE = Integrated Development Environment  
LCD = Liquid Crystal Display  
LED = Light-Emitting Diode  
PCB = Printed Circuit Board  
USB = Universal Serial Bus  
PWM = Pulse Width Modulation  
RAM = Read Access Memory  
RD = Read  
ROM = Read-Only Memory  
RSR = Receive Shift Register  
SPI = Serial Peripheral Interface  
SMD = Surface-Mounted Device  
USB = Universal Serial Bus  
WR = Write



# CHAPTER 1

## INTRODUCTION

### 1.1 THESIS MOTIVATION

In our days, there are more and more electronic devices all around us. Most of them actually are a lot of interconnected electronic components, called Integrated Circuits (IC). There are devices with number of 12 up to 144 pins.

A big trouble for all embedded programmers is that, sometimes, on the hardware, there are all kinds of short-circuits, or interrupted traces. Especially when the circuit is designed at home, there are more chances that this type of trouble could appear.

In order to better understand this project, there will be defined the following statements:

A PIN is the metal external leg of an IC. The pin connects the Programmable Logic Device or other type of IC to a circuit board and other devices.

A FAULT is a manufacturing defect which can cause the circuit malfunction or fail. In-circuit board test operations are intended to detect circuit board assembly fault, typically bent, open or shorted device pins.

This application has been designed to be used on an IC that cannot be easily removed from its place, like the ones that are inside the car.

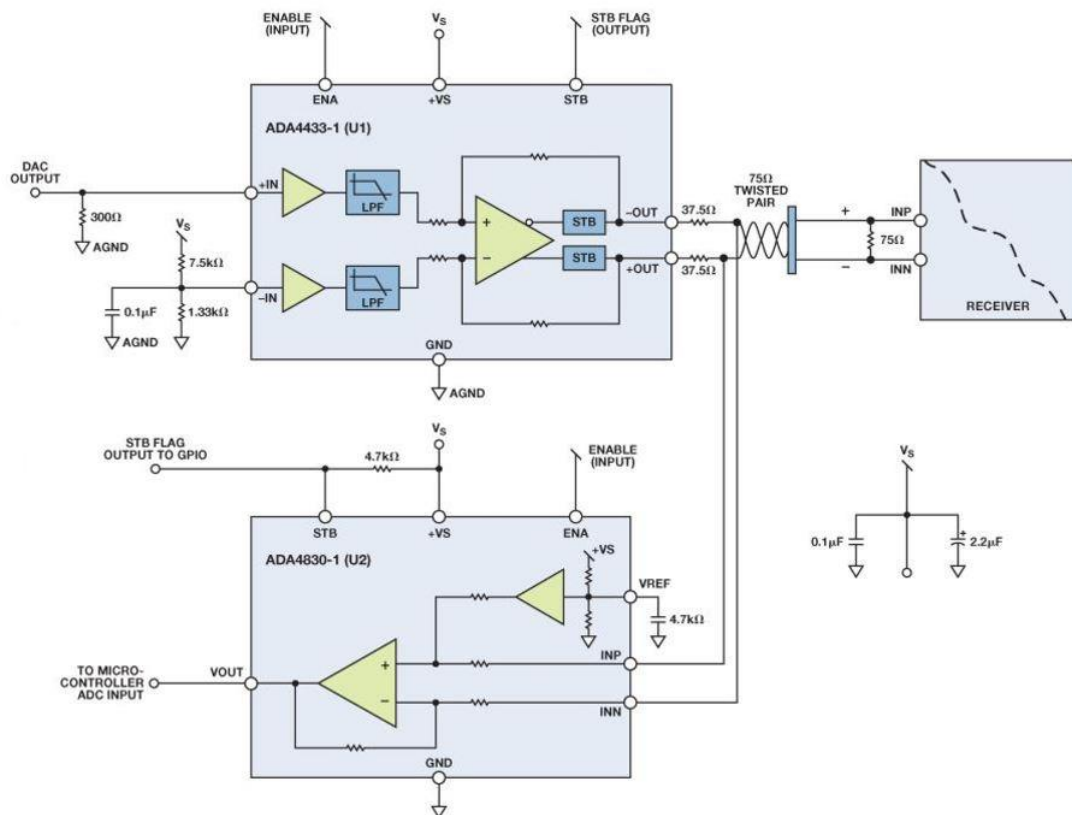
## 1.2 METHODS OF DETECTING FAULTS

In order to detect this type of problems there are hardware equipments and software applications.

The most common method used is to measure the voltage and the current on every pin of the IC using a multi-meter and a power supply IC (Integrated Circuits). This method is working for devices with small number of pins, but for devices with hundreds of pins it is very time consuming.

Another method is to use OptoTherm EL infrared circuit board inspection board, which detects where the board is heating up. This method needs a special equipment and specialized software. The disadvantage of this tool is that not anyone can afford it.

Another method of detecting fault in integrated circuit is used in some of the car's board.



**Figure 1-1. Wire diagnostic circuit (Nisbett, 2012)**

The principle of this circuit is that ADA4433-1 (U1) fully integrated video reconstruction filter as part of the video transmission signal chain and an ADA4830-1 (U2) high-speed difference amplifier as the detection circuit. The ADA4433-1 features a high-order filter with a  $-3$ -dB cutoff frequency of 10 MHz, 45-dB rejection at 27 MHz, and an internally fixed gain of 2 V/V. It has excellent video specifications, overvoltage protection (STB) and overcurrent protection (STG) on its outputs, and low power consumption. (Nisbett, 2012)

I am working with microcontrollers every day and to apply my knowledge acquired in studding years, for my diploma project I choose to create a software method of detecting the short circuit on a Printed Circuit Board.



### 1.3 APPLICATION DESCRIPTION

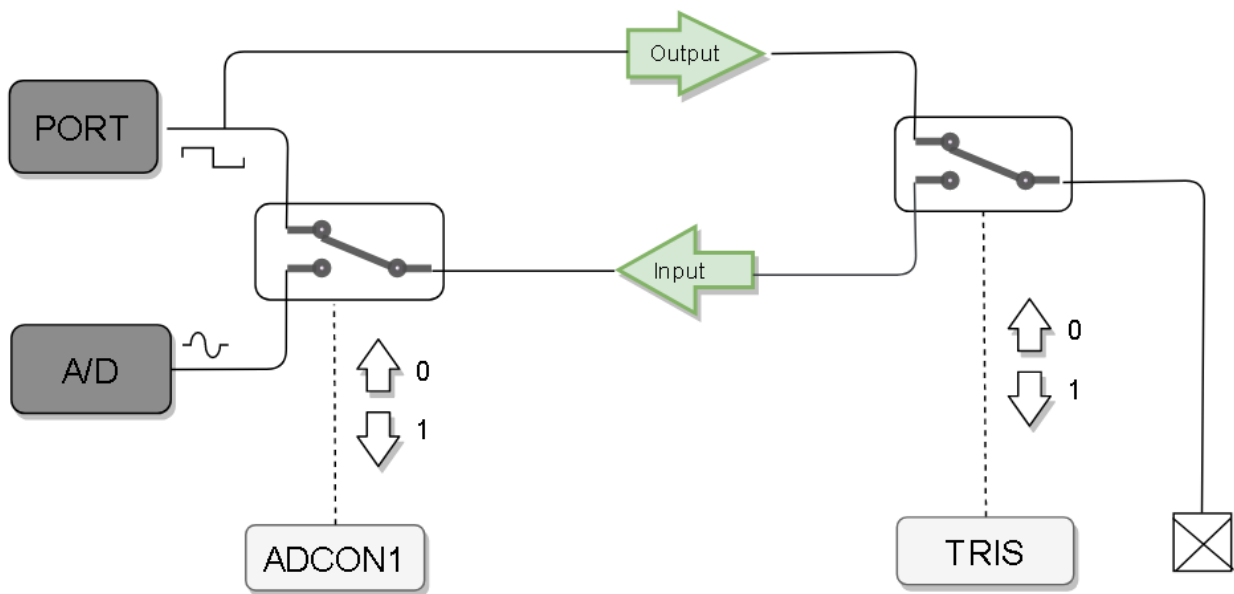
This project has the purpose to announce the user before he will try to program its hardware, if there are any faulty pins. The results will be communicated to signal by two methods: an LCD display and a serial communication.

The serial one is an UART communication, which will display the result on RS232 terminal. For this project “Hercules” software (HW group) will be used.

The project can be applied for the pins that have also a function of ADC. In order to detect a short, it must be processed the signal read on the pin. The known values of any I/O pin are: input HIGH or LOW and output HIGH or LOW. These values are known, these can be used as references for the values read with analog function. The analog signal, which has been read, is compared with the reference values.

For a LED connected to the pin has been found that for a functional state, the value read when the pin is an output pin, it has to be between 0 and 0.5V, and when the pin is an input pin, the interval is [1.1 – 1.5]V. Both conditions must be fulfilled. If one of the conditions is not true than that pin is faulty.

In figure 1-2 it is showed a simple internal scheme of one of the microcontroller’s pin. It can be observed that there are internal switches, which can be changed if the control registers TRIS and ADCON1 are modified.

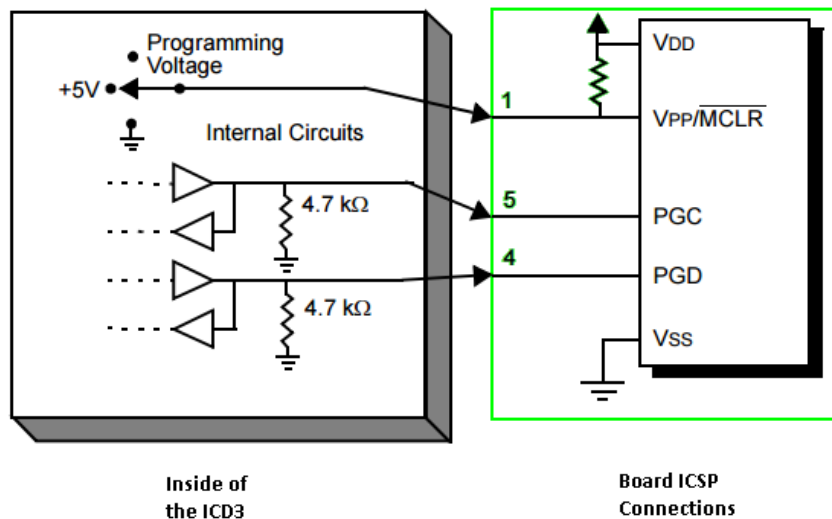


**Figure 1-2. I/O with A/D function**

In subchapter 2.1.2 ADC module, will be explained why an analog read recommended than a PORT register read.

The simulation of a short can be realized, because every pin has a protection circuit. This is composed by 2 diodes D1 which is opening at high voltages and D2 which will open at negative voltage values. These diodes will conduct the current outside the microcontroller, into the power source.

For this project the power supply is the ICD3 tool which has a hardware overvoltage/overcurrent protection. For PGD and PGC, ICD3 has internal pull-down resistors, with a value equal to 4.7kΩ. It can supply a maximum current of 10mA and the voltage range is 3-5V. This setup is showed in Figure 1-3.



**Figure 1-3. ICD3 internal pull-ups (Microchip Technology Inc., 2012-1014)**

For programming the ICSP (In Circuit Serial Programming) connection is used. There are 6 pins used to connect the ICD3 with the microcontroller. These pins are: GND,  $V_{dd}$ , MCLR ( $V_{pp}$ ), PGD, PGC and the 6<sup>th</sup> one (PGM) pin is not used in this application.

“Vpp pin connects to the reset input of the pic microcontroller labelled MCLR. During programming or verify this signal is raised to the programming voltage (13.5V) - or  $V_{CC}+3.5V$ . This signals to the microcontroller that programming/verification is about to start and for older parts, supplies current.” (Details of PIC ICSP and how to use it for pic microcontrollers, 2005)

PGC states for programming clock signal, and PGD is programming data signal. Actually these two wires are used for transmit the instruction to the microcontroller.

“Data (PGD) and clock (PGC) transmit data to the pic micro. First data is sent either high or low voltage (0/1). After a suitable time the clock is strobed low to high - rising edge clocking the data into the microcontroller. PGD is also the line driven by the pic micro during verify i.e. it is bi-directional.” (Details of PIC ICSP and how to use it for pic microcontrollers, 2005)

## CHAPTER 2

### COMPONENTS DESCRIPTION

#### 2.1 MICROCONTROLLER

A microcontroller is a computing microsystem which can have, on the same IC, different specialized components and it is used for embedded applications. Microcontrollers are also named: “the heart of an embedded system”.

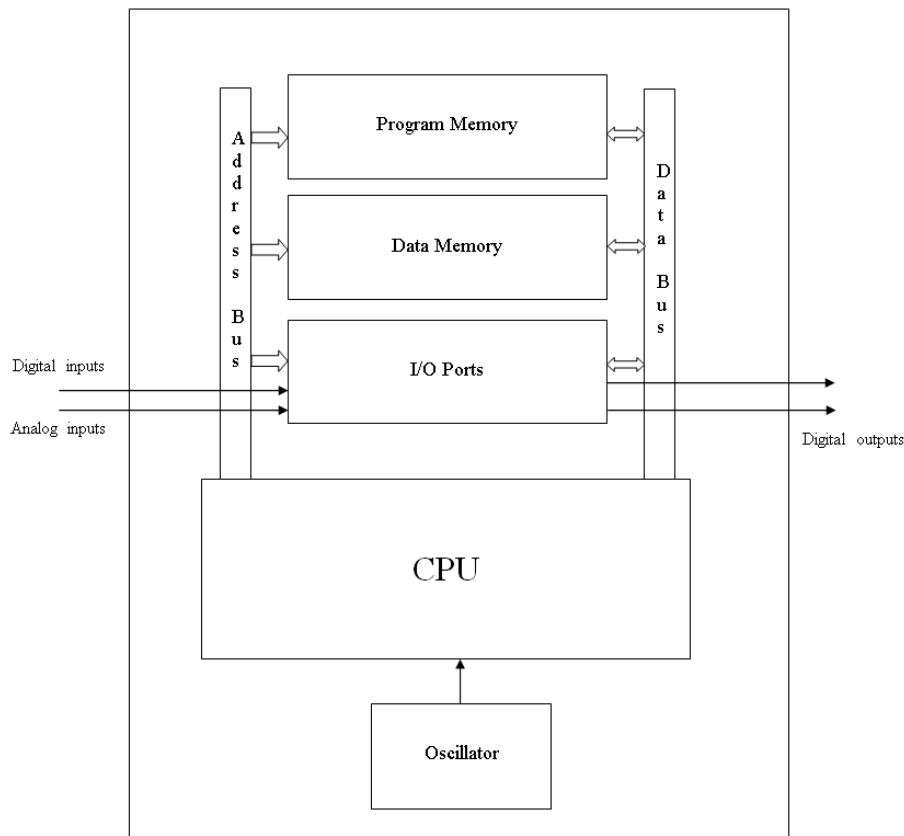
The main components of a microcontroller are: CPU(s), Memory, Oscillator and Peripherals, all are encapsulated in the same IC. There are microcontrollers with different performance CPUs, a different number of specialized peripherals, and different types of memories depending on their utilization.

Microcontrollers can be programmed with a specific set of instructions. This set is different from one family of microcontroller to another one. This set contains logic operations, arithmetic instructions, memory access commands, and specific instructions for accessing and configuration of peripherals modules and I/O ports.

For this type of instruction set, there are specific software programs, a proper environment and specific hardware instruments, which realizes the connection between microcontroller and software program.

These tools are used for programming the IC and also for verifying, in real time, the functionality of the written embedded code, though debugging. The user is able to run the code step-by-step, this thing being useful for visualization of the specific variables values, the written values in the data memory and other resources of the microcontroller.

The chosen microcontroller for this project is PIC18F4550 (Figure 2-2.), created by Microchip Technology Inc. factory, with a RISC architecture (Reduced Instruction Set Computer). First letters presented in the name of the microcontroller: “PIC” means: Peripheral Interface Controller.



**Figure 2-1. Block Diagram of a Microcontroller**

It has been chosen for the fact that it has a DIP packaging with 40 pins; it has 100,000 Erase/Write Cycle Enhanced Flash and 1,000,000 Erase/Write Cycle Data EEPROM Memory Typical, it can easily protect the code with the Programmable Code Protection module, it can be power either with single supply of 5V directly from In-Circuit Serial Programming or with an external power supply starting from 4.2V up to 5.5V. There is also a low-voltage part (PIC18LF4550), which can be used for applying the software project for devices which are powered with a low voltage (2.0V up to 5.5V). (Microchip, 2009).



**Figure 2-2. PIC18F4550 DIP packaging (Microchip, 2009)**

This microcontroller has an internal oscillator which works on a frequency of 8MHz, but it also has eight derived user-selectable frequency, from 31kHz to 8MHz. For the oscillator, the

highest frequency has been chosen, because it is needed to be able to read the pin status as fast as possible.

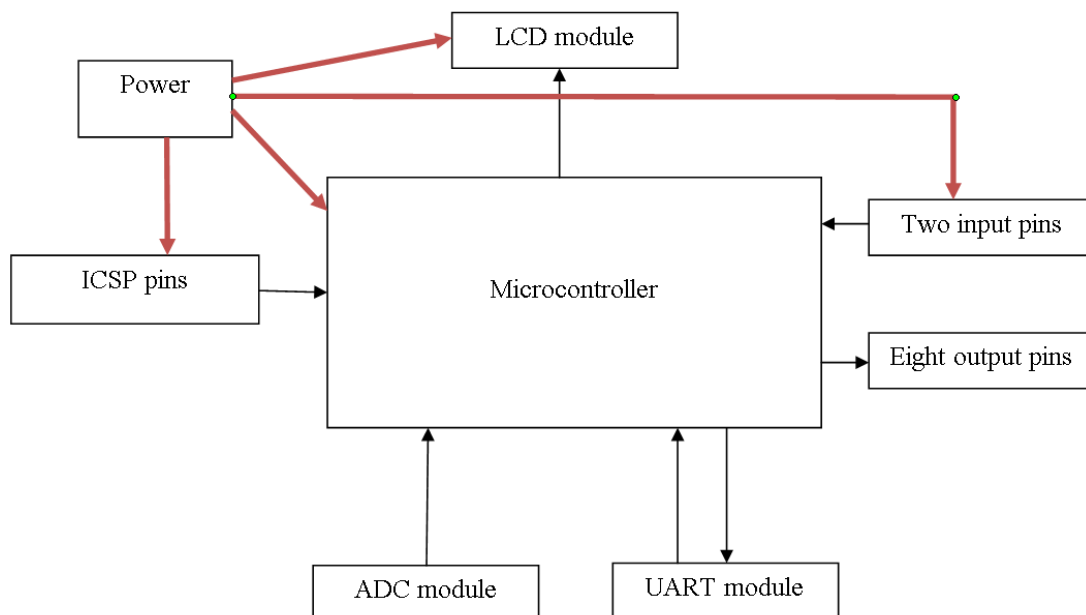
In this application, the microcontroller will be powered with a 5V voltage and it will have the working frequency set at 8MHz.

PIC18F4550 has the following peripheral blocks:

- Universal Serial Bus (USB)
- Three external interrupts, this means it has 2 pins for external interrupts and 4 pins for interrupt-on-change.
- Four timer modules
- Capture / Compare / PWM modules
- Enhanced Universal Synchronous / Asynchronous Receiver Transmitter (EUSART), with an automatic selectable band
- A serial port which can be used for SPI / I2C protocols
- 10 bit Analog-to-Digital Converter, with 13 channels having programmable acquisition time
- Five bidirectional input / output ports

The hardware part of the project contains a Microchip device, so the software part will be created with the MPLAB X tool, also created by Microchip.

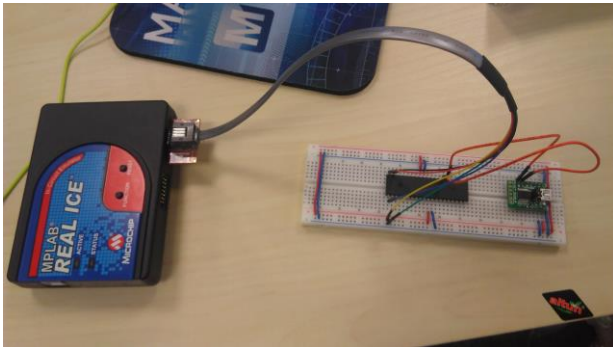
### 2.1.1 Pin Connections



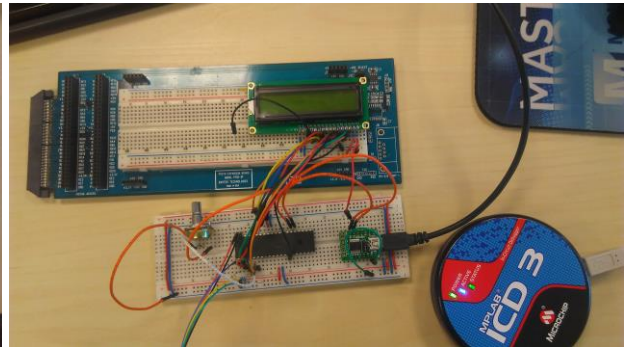
**Figure 2-3. Block diagram for this project**

The hardware part of this project has been chosen to have two methods of returning the results to the user, a small consumption of resources and to have a small number of hardware components.

The first test was made on a simulated circuit placed on a breadboard (Figure 2-4), testing if the programmer REAL ICE will be suitable for my short tests and Figure 2-5 where is used an ICD3 as a programmer. It was decided that the programmer used will be the ICD3, because has a better protection. It will be discussed more about the ICD3 in Chapter 3.3.



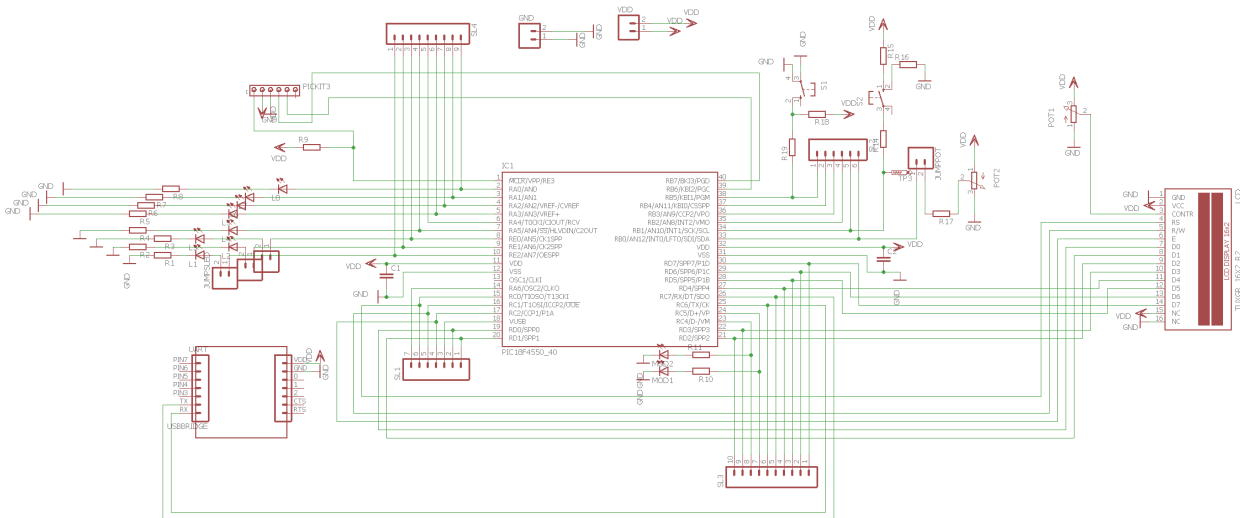
**Figure 2-4. Breadboard circuit with REAL ICE**



**Figure 2-5. Breadboard circuit with ICD3**

After this simulation circuit on breadboard was made the schematics (Figure 2-6). It was a good practice that the hardware part was made on the breadboard, because it is easier to change the designed components.

The schematics and layout were made in “Eagle 7.6” (Eagle) software. “EAGLE is a scriptable electronic design automation application with schematic capture, printed circuit board layout, auto-router and computer-aided manufacturing features. EAGLE stands for Easily Applicable Graphical Layout Editor (German: Einfach Anzuwendender Grafischer Layout-Editor) and is developed by CadSoft Computer GmbH. Cadsoft Computer GmbH was acquired by Autodesk Inc. in 2016.” (Eagle, 2017)

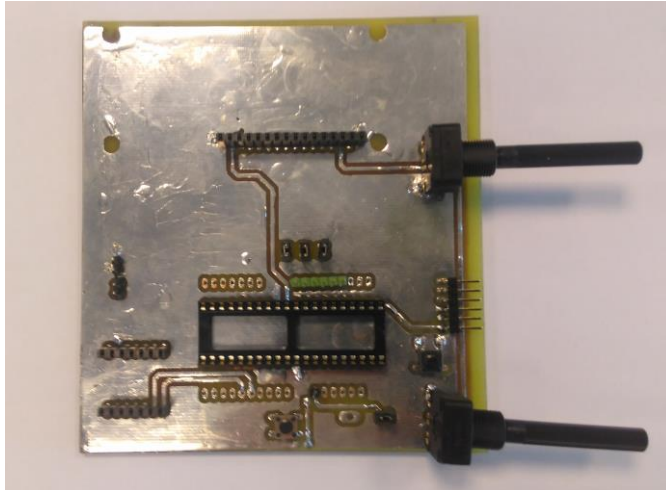


**Figure 2-6. Project Schematics (Annex1)**

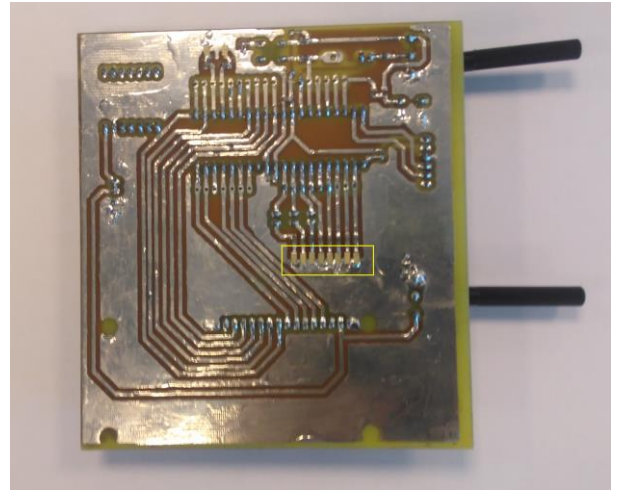
The next step for this project was to create a printed circuit board (Figure 2-7). It was needed a test like this, because there were doubts that my circuit will be able to handle a short simulation. Because it was a home-made circuit, there were a lot of mistakes, but there were important lessons to learn from each of it.

The visible mistake is that the SMD LEDs had the footprint on the bottom side of my PCB. Then there was discovered that my potentiometers had a value too big for what needs had the project. Then it was observed that the button used for interrupt had a debounce for each push, and the software code never enter in the main function. This will be discussed in Chapter 3.4.

Another change was made for programming pins. These are the pins which are linked to the programmer (ICD3). First I wanted to use a “J-Tag connect” wire, but this would have needed some extra vias and for the home-made circuit was a big work to do.



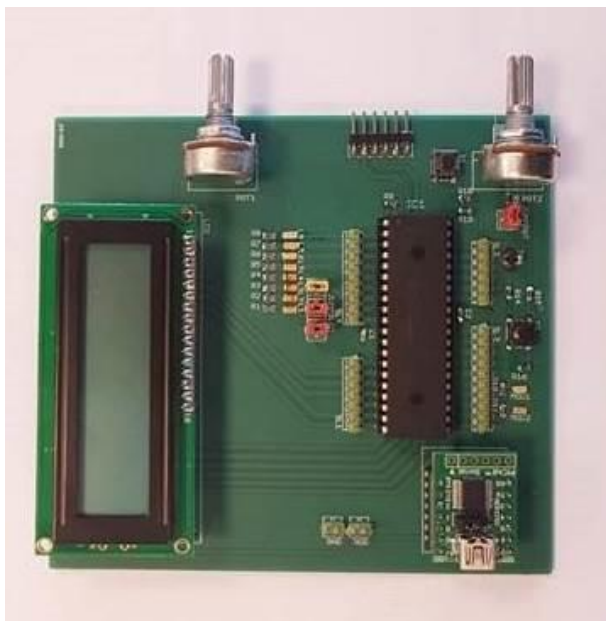
a) Top view



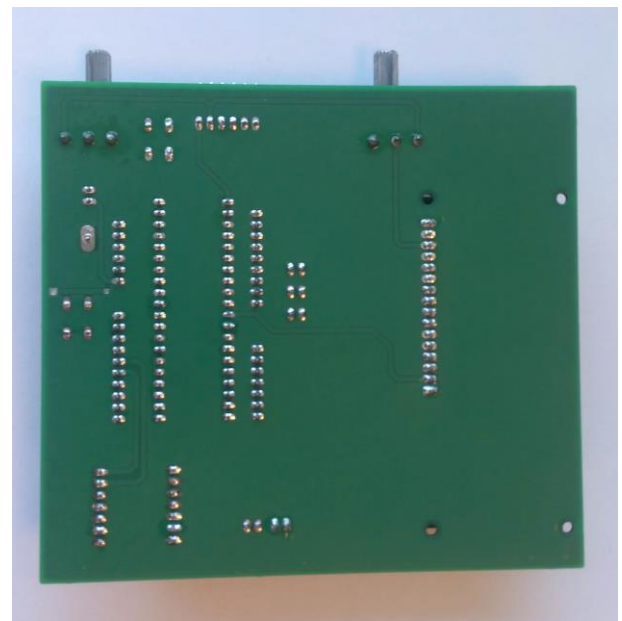
b) Bottom view

**Figure 2-7. Home-made PCB**

After the adjustment for the schematics and layout, the Gerbers files were sent to EuroCircuits factory to create the project (Figure 2-8 and Figure 2-9). The board came un-populated, not like in the picture.



**Figure 2-9. Top view Final PCB**



**Figure 2-8. Bottom view PCB**



The hardest part for this PCB was to solder the SMD LEDS, because the hot air station was used. Sockets have been used for the components that can be damaged: LCD, PIC, and MCP2200 Breakout Module.

For an easier simulation of a short there are GND and VDD pins and for microcontroller's pin there are vertical headers, so it can be used only a mother-mother wire to simulate the short. For interrupt routes, jumpers have been used.

For connecting the ICD3, a header with 6 pins has been used. To this header will be connected an adaptor: "RJ-11 to ICSP Adapter" (Figure 3-4).

### 2.1.2 ADC Module

Analog to Digital Converter (ADC) is a device that converts an analog quantity (continuous voltage) to discrete digital values.

This is very useful because it allows the microcontroller to read an analog signal, from a specific pin of the integrated circuit, when we want to do some processing on physical quantities, which are normally analog in nature. (George, 2015)

The ADC module allows the microcontroller to read an analog signal, from a specific pin of its, and to process that value, depending on the software code that has been written in the microcontroller (Figure 2-2).



Figure 2-10.ADC Symbol (George, 2015)

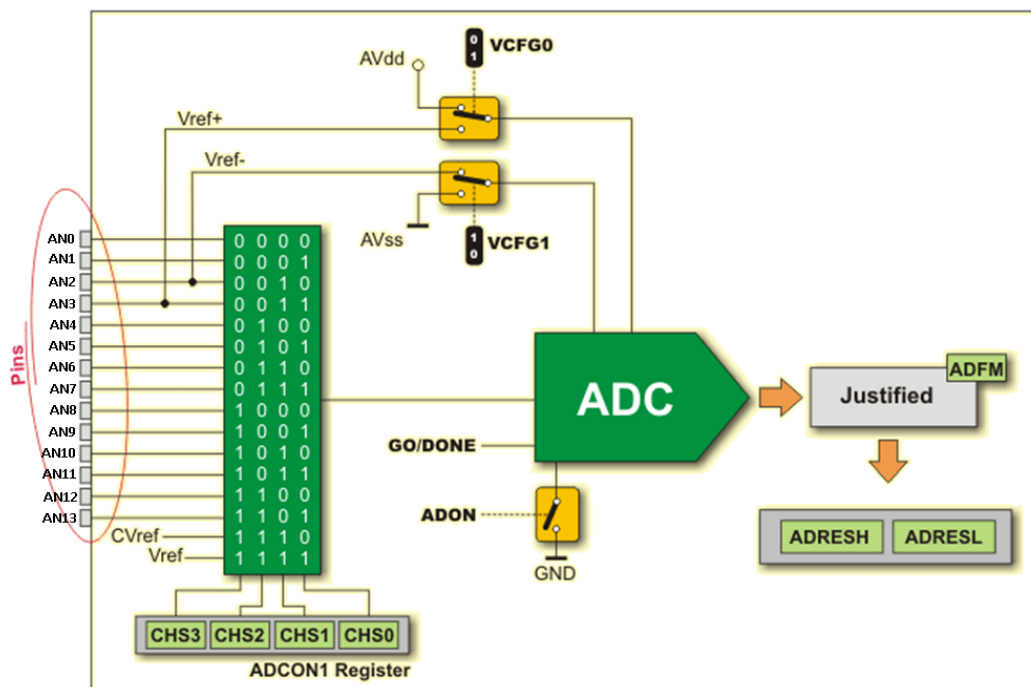


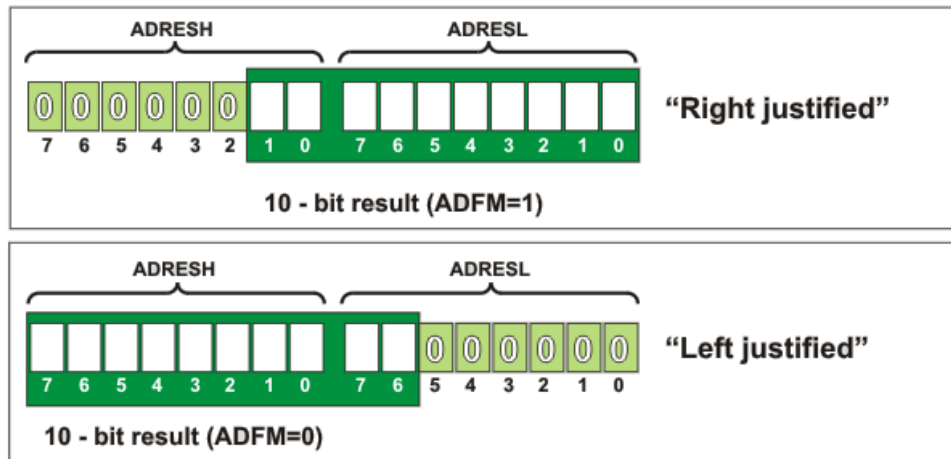
Figure 2-11.A/D Block Diagram (Verle, 2008)

When this module is used, first thing is to choose a pin of the microcontroller, which has an A/D function. As it is described in Figure2-3, there is needed to be chosen a reference voltage and the work frequency for this module. After microcontroller receives the instruction to



convert, the program will wait, that the data is received correctly, a period of time which depends on the work frequency.

The value read on the input-analog pin is converted by the ADC module into a 10 bits digital value. The conversion of the signal is made after CHS3:CHS0 (Analog Channel Select bits) are set, in order that the converter will know which pin must be read. The signal is then passed to the A/D convertor, which through successive approximations will generate a 10 bits digital result. This result is stored in a pair of 8 bit registers: ADRESH (Analog-to-Digital Result High) and ADRESL (Analog-to-Digital Result Low). The user will be able to access these registers, and their values can be stored in different variables, for a later usage. A/D result format can be left justified or right justified (Figure2-4.). This setup is necessary because the A/D module store the 10 bit results in two 8 bits registers which have a capability of 16 bits.

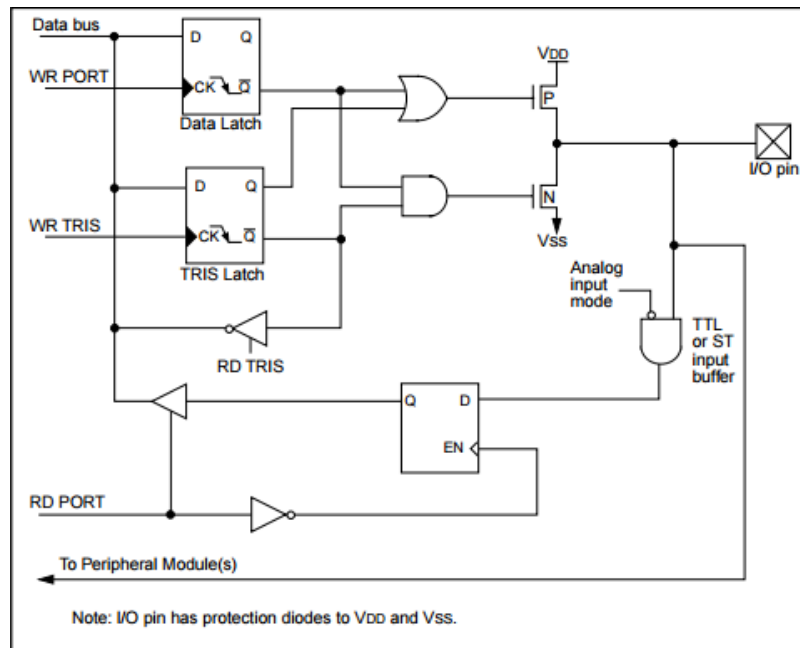


**Figure 2-12. Left / Right Justified (Verle, 2008)**

“For the ADC to meet its specified accuracy, it is necessary to provide a certain time delay between selecting specific analog input and measurement itself. This time is called “acquisition time” and mainly depends on the source impedance”. (Verle, 2008)

Pins used for ADC module usually have multiplexed one or more on them, additional functions, and those pins won't be used as a general purpose I/O pin. To indicate the direction of the pin, there is TRIS register. When this is set (has the value 1) means that the pin is an Input pin (1 == I) and when is cleared the pin is an output one (0 == O). „The PORT register is the latch for the data to be output. When the PORT is read, the device reads the levels present on the I/O pins (not the latch). This means that care should be taken with read-modify-write commands on the ports and changing the direction of a pin from an input to an output.” (Microchip Technology Inc., 1997)

This means, that if the user want to verify with a regular reading of the pin, the value read from the PORT register, it will be a digital value, which has two states: 0 or 1. But in order to detect a faulty pin, the microcontroller needs analog values. In figure 2-7 can be seen that the analog signal will be transmitted to the AD convertor without any processing. Eventhough there is a disadvantage: the analog value needs time to be converted, there are useful information which will be compared with the reference values.



**Figure 2-13. I/O port operation (Gottardo, 1999-2013)**

All the modification it was spoken about, are accessible in several registers. For PIC18F4550 the registers associated with the ADC module are the following:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

### 2.1.3 UART Module

The two EUSART modules (Enhanced Universal Synchronous/Asynchronous Receiver Transmitter) are serial communication interfaces which contain shift registers, buffers where the data are kept, in order to be transmitted. A serial communication means that the multi-bit word is transmitted bit after bit.

This module provides Full Duplex (there are one wire for transmitting and one wire for receiving) communication between two devices. The working principle of this type of communication is convert a serial data flux, which is a digital signal, into an 8 bits values, which can be stored and processed by the microcontroller. It is able to convert a sequence of 8 bit values into a digital signal, which can be sent to other devices.

EUSART can be configured in following modes:

1. UART – Asynchronous (Full Duplex)
2. USRT Master – Synchronous (Half Duplex)
3. USRT Slave – Synchronous (Half Duplex)



**Figure 2-14. Transmission Types**

For the last two modes, one wire is used for data and another one for clock signal. This means that the transmission is a Half-Duplex one, and therefore one device will be “Master” telling the “Slave” when is ready to receive dates and when it is ready to transmit.

First mode it doesn’t need a “Master” and a “Slave”, because there are two different wires for transmitting and receiving and is an asynchronous mode.

This project used the asynchronous mode, in a simplex transmission, because it is used to display the pin state, if there is any faulty pin (only the a board sends dates to PC). UART module is connected to a USB Bridge (MCP2200). This bridge send the dates, using an USB cable, to the PC and, respectively, to the RS232 terminal. For this project “Hercules” terminal (HW group) has been used.

UART interface is composed by:

- TX Transmitter (Figure 2-15) -> this is able to convert the data bytes received from Central Processing Unit, into a sequence of serial dates, which will be transmitted on microcontroller’s TX pin.
- RX Receiver (Figure 2-16) -> this is able to convert a serial sequence of bits, received on RX pin of the microcontroller, into bytes of data, which are stored in RCREG and can be used by the program.

For the UART’s transmitter block diagram (Figure 2-15), Transmit Shift Register (TSR) is the most important component, because it cannot be software accessed and TSR is a temporary buffer which is loaded with data if ther eis not a transmission in progress, and new data is loaded in TXREG.

TXREG register represent the 8 bits buffer of the Transmitter, it is used by the program to load the information that we want to transmit.

“Using 2 registers allows faster the transmission of the data. Once the TXREG register transfers the data to the TSR register, the TXREG register is empty and flag bit, TXIF is set.” (MicrocontrollerBoard, 2008)

SPBRG – sets the desired baud rate of transmission

TXIE – allows interrupts when TXREG is empty and TXIF is set

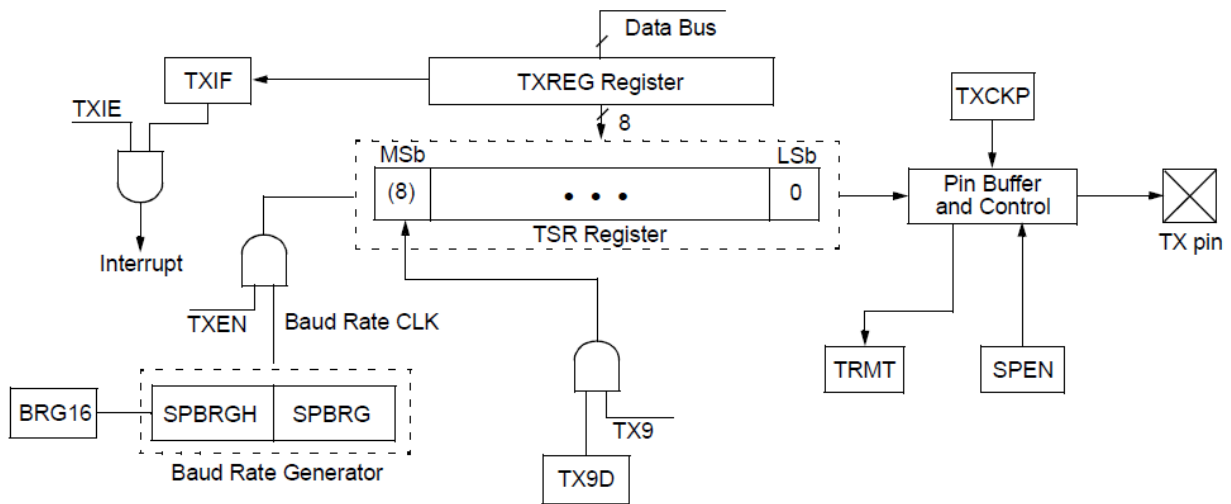
TXEN – Enabling the transmitter module

TX9D – enable 9 bits words transmission

TRMT – shows the status of TSR register (empty or full)

TXCKP – allows the TX signal to be inverted

SPEN – enable the asynchronous serial port



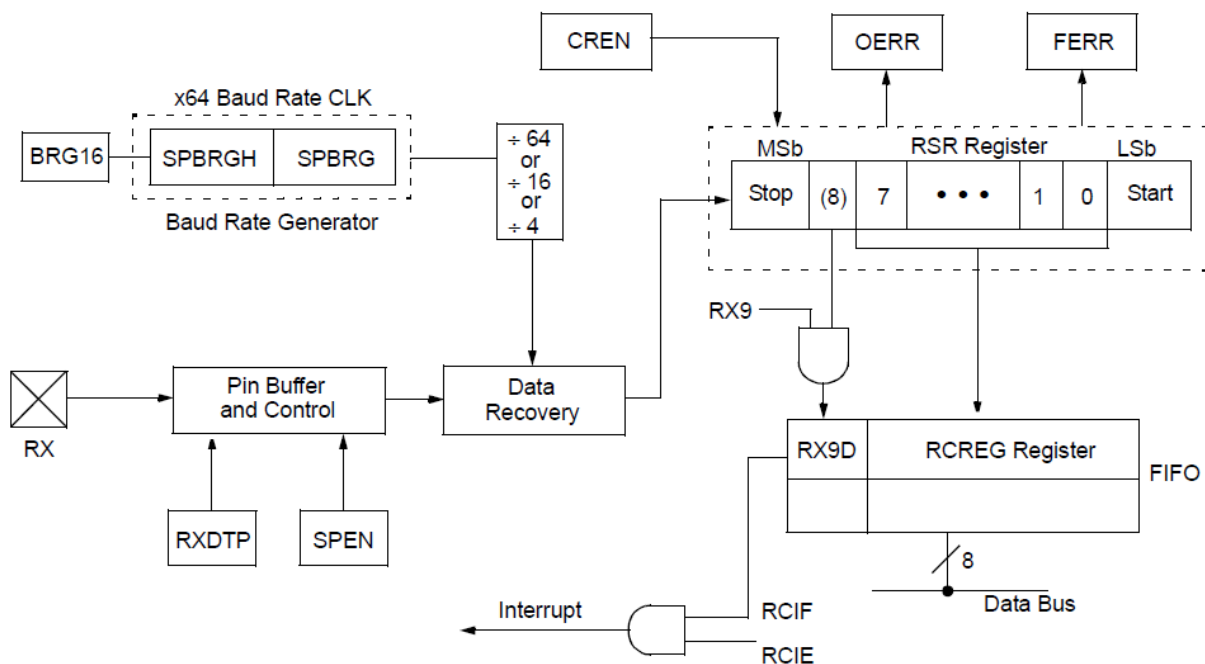
**Figure 2-15. EUSART Transmit Block Diagram (Microchip, 2009)**

After receiving the data in the shift register RSR (Receive Shift Register) (Figure 2-16), the information is loaded at the same time into the register RCREG. Also RSR is not available to be modified by the software.

“Using 2 registers allows faster receiving of the data. While the information that was received being transferred into RCREG, the new information has already been received into the register RSR.” (MicrocontrollerBoard, 2008).

EUSART module has most of control bits in three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)



**Figure 2-16. EUSART Receive Block Diagram (Microchip, 2009)**

From TXSTA register it can be controlled if the device will be slave or master (CSRC), (for a synchronous transmission); if the information transmitted will be on 8 bits or 9 bits (Tx9), transmission can be enabled (TXEN); it is decided if it is a synchronous or asynchronous transmission (SYNC); SENDB is to send a break character and TRMT tells the status of the TSR (Table 2-1).

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN <sup>(1)</sup>	SYNC	SENDER	BRGH	TRMT	TX9D
bit 7							bit 0

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

**Table 2-1. Transmit Status and Control (Microchip, 2009)**

For receiver (Table 2-2) there are almost the same commands: SPEN allows enabling the serial port, it can be decided if the information word is on 9 bits or 8 bits (RX9); SREN tells to the microcontroller, if the device is a Master that next will be a single receive or not, CREN enable the receiver, and OERR is Overrun Error bit, which tells if there was an error or not.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

**Table 2-2. Receive Status and Control (Microchip, 2009)**

The BAUDCON register (Table 2-3) has the main function to control the baud rate of the transmission.

“Baud Rate represents the number of bits that are actually being sent over the media, not the amount of data that is actually moved from one DTE device to the other. The Baud count includes the overhead bits Start, Stop and Parity that are generated by the sending UART and removed by the receiving UART. This means that seven-bit words of data actually take 10 bits to be completely transmitted.” (Durda, 2014)

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

**Table 2-3. Baud Rate Control (Microchip, 2009)**

An important bit from this register is BRG16. It defines the timer that will divide the clock. By default, the frequency generator works on 8-bit mode, but if this bit is set, the frequency generator can be on 16-bit.

The clock configuration used in the EUSART module is determined by the SPBRGH and SPBRGL registers to determine the baud clock. The configuration also depends on the BRGH

bit of the TXSTA register for asynchronous mode, and for the synchronous mode, this bit is ignored. The SYNC bit in the TXSTA register is used to configure the clock mode to be synchronous or asynchronous.

## 2.2 USB BREAKOUT MODULE

“The MCP2200 is an USB-to-UART serial converter which enables USB connectivity in application that has an UART interface. The device reduces external components by integrating the USB termination resistors. The MCP2200 also has 256-bytes of integrated user EEPROM. The MCP2200 has eight general purpose input / output pins. Four of the pins have alternate functions to indicate USB and communication status.” (Microchip, 2017)

The MCP2200 Breakout Module has the following features:

- UART TX and RX signals
- UART RTS and CTS signals
- 8 General Purpose (GP) lines - configurable for GPIO or dedicated function operation
- User selectable power supply of 3.3V or 5V (up to 500 mA) by using a jumper
- DIP form-factor (0.6 inches overall row spacing between pins)
- ICD3™ Serial Analyzer header – used for UART communication only (Microchip, 2012)

This converter helps for the connection from EUSART module of a microcontroller to the PC. In figure 2-18 the connection between microcontroller and breakout module is represented.

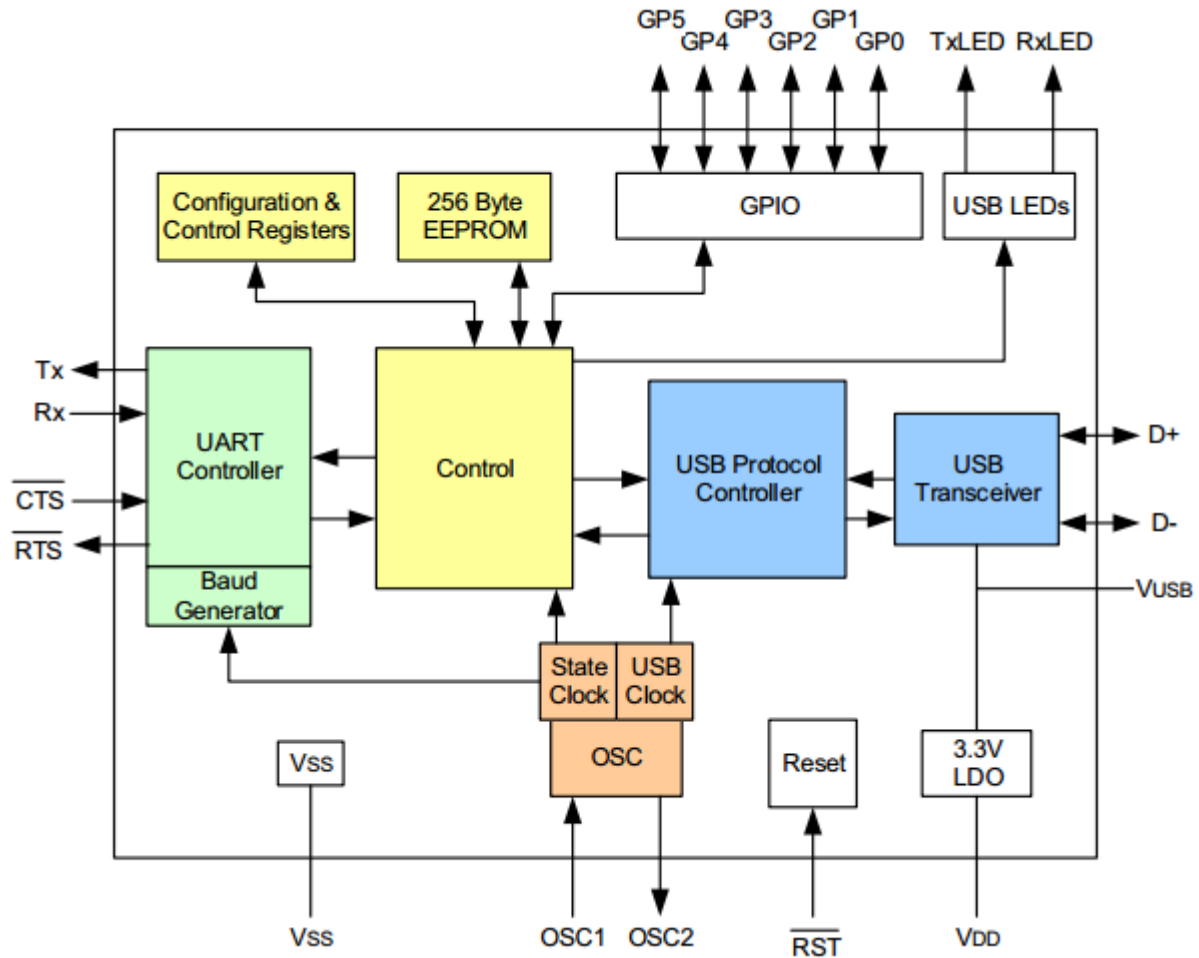


Figure 2-17.MCP2200 USB-to-UART Protocol Converter (MCP2200 USB-to-UART Serial Converter, 2010)

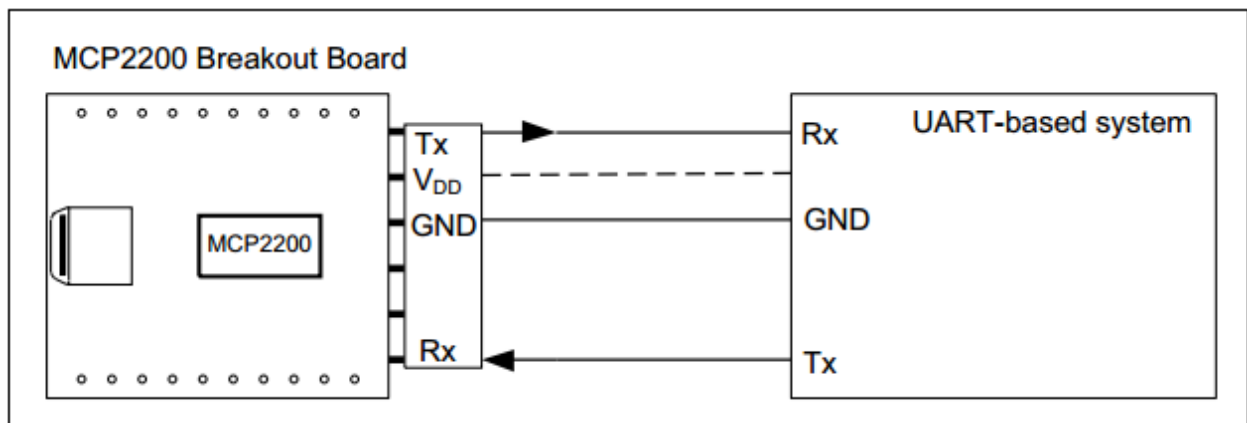


Figure 2-18.Typical usage diagram (Microchip, 2017)

The **Control Module** is the heart of this bridge. All other components are tied together and controlled via this module. The control module, manages the data transfer between the USB and UART, as well as the command requests generated by USB host controller and the commands for controlling the function of UART and I/O.

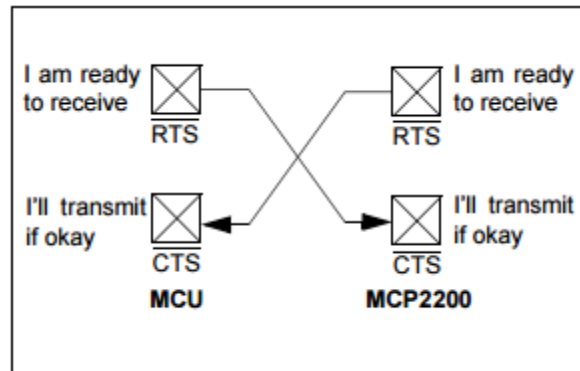
**UART Interface** has the RX and TX data signals connected to the TX and RX pins of the PIC. In addition it has RTS/CTS pins, which are controlling the data flow (figure 2-19).

**USB Protocol Controller** is full-speed USB compliant. This part controls the data flow from breakout module to the USB cable.

**USB Transceiver** is internally connected to the USB module. The transceiver obtains power from  $V_{\text{USB}}$  pin, which is internally connected to the 3.3V regulator.

**GPIO Module** is a standard 8-bit I/O port. These pins can be configured as:

- GPIO – individually configurable general purpose input or output
- SSPND – USB Suspend state
- USBCFG – indicates USB configuration status
- RxLED – indicates USB receive traffic
- TxLED – indicates USB transmit traffic



**Figure 2-19. RTS/CTS Connections Example (Microchip, 2017)**

**EEPROM Module** is a 256-byte array of non-volatile memory. The memory cells for data EEPROM are rated to endure thousands of erase/write cycles, up to 100k.

**Reset** pin provides a method for triggering an external Reset of the device. A Reset is generated by holding pin low. These devices have a noise filter in the Reset path which is able to detect and ignore small pulses.

**Oscillator** must be 12 MHz to provide the proper frequency for USB module.

## 2.3 LCD SETUP

For a second displaying method, an LCD (Liquid Crystal Display) has been used, which is an electronic display, in our days is very used.

For this project, a 16x2 LCD has been used. This means that the LCD has 2 lines of 16 characters each. In total it can display 32 characters. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

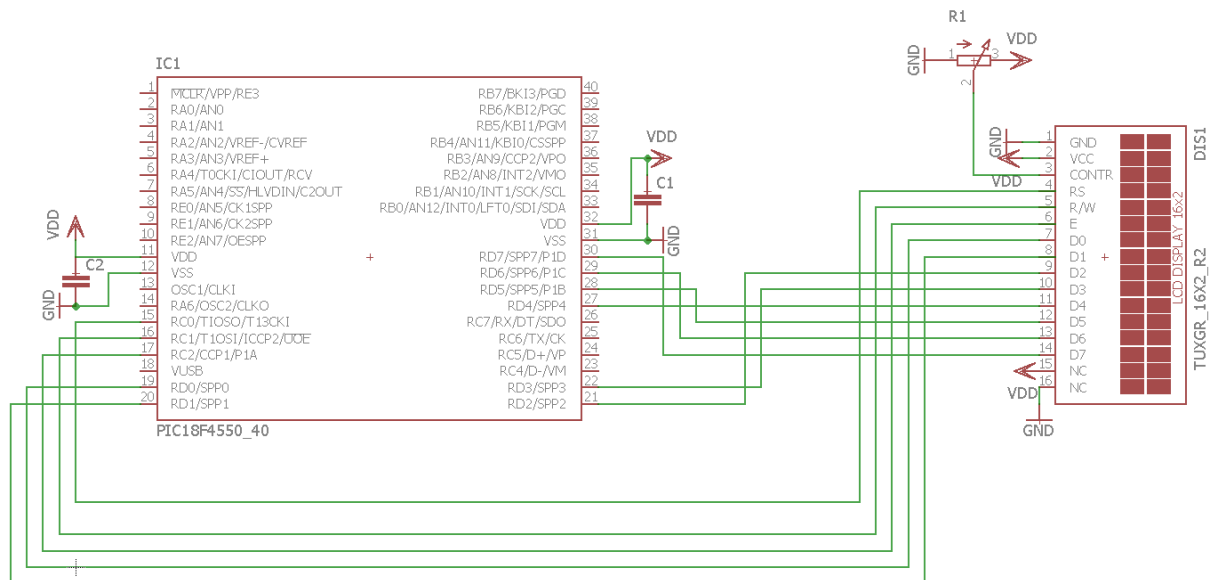
The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

In addition, there is a potentiometer which allows the user to change the contrast between the character and the background. It can be replaced by a PWM signal generated by the microcontroller, but it has been chosen to use a potentiometer, because it is easier to change manually, in order to have the best displaying.

This type of LCD has been chosen because it is compact, has low power consumption; it is power with 5V. Also it is the most used type of LCD, so this project has a bigger change to be implemented with success.

In Figure 2-20 it is showed how the LCD is connected to PIC18F4550. It is used PORTD for data signals and 3 pins of PORTC are used for command signals.





**Figure 2-20. Simplified schematic for LCD and PIC18F4550**

In Table 3-1 there are described all LCD's pins and their function. The 15th and 16th pins are connected to Vdd, even though the LCD used for this project doesn't have a backlight.

The data pins of the LCD are connected at a single port of the microcontroller: Port D. For commands lines: RS, R/W, E are used three pins from Port C of the microcontroller. An easily accessible by the user potentiometer is used for adjusting the LCD contrast.

LCD pin number	LCD pin Name:	Function of the pin:	Connected to PIC's pin:
1	GND	Ground (0V)	GND
2	V <sub>CC</sub>	Supply voltage; 5V (4.7V – 5.3V)	V <sub>dd</sub>
3	CONTR	Contrast adjustment; the best way is to use variable resistor such as a potentiometer. The output of the potentiometer is connected to this pin. Rotate the potentiometer knob forward and backwards to adjust the LCD contrast.	Potentiometer R1
4	RS	Selects command register when low, and data register when high	RC0
5	R/W	Low to write to the register; High to read from the register	RC1
6	E	Sends data to data pins when a high to low pulse is given; Extra voltage push is required to execute the instruction and EN(enable) signal is used for this purpose. Usually, we make it en=0 and when we want to execute the instruction we make it high en=1 for some milliseconds. After this we again make it ground that is, en=0.	RC2
7	D0	8-bit data pins	RD0
8	D1		RD1

9	D2	8-bit data pins	RD2
10	D3		RD3
11	D4		RD4
12	D5		RD5
12	D6		RD6
13	D7		RD7
14	D8		RD8
15	NC	For display with backlight $V_{CC}$	$V_{dd}$
16	NC	For display with backlight GND	GND

**Table 2-4.LCD pins description**

## CHAPTER 3

### SOFTWARE DESCRIPTION

In order to understand the project, the software development environment (MPLAB X IDE), base code project, displaying methods and detection function will be further described.

#### 3.1 MPLAB X INTEGRATED DEVELOPMENT ENVIRONMENT

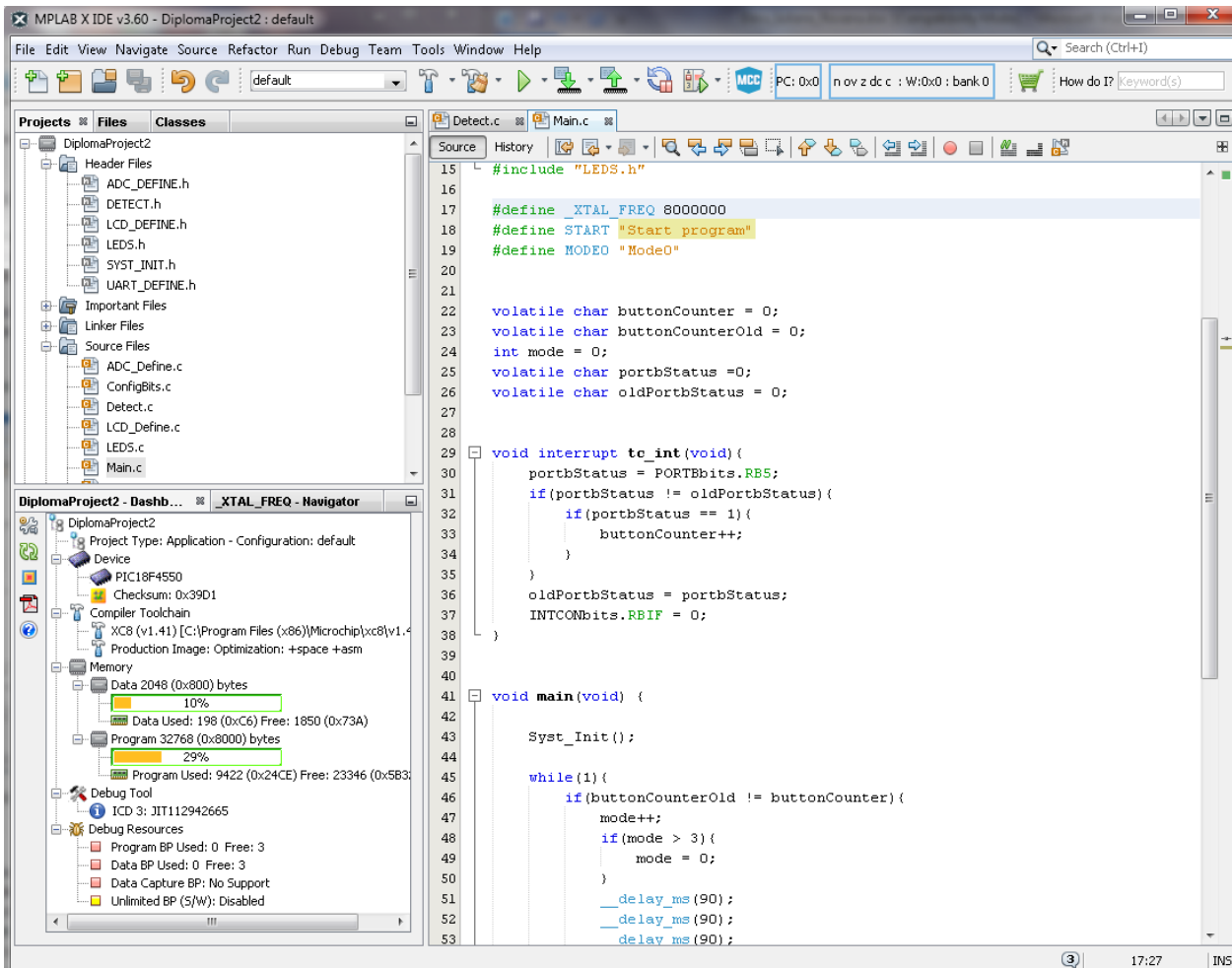
For this project it is used a microcontroller, produced by *Microchip Technology Inc.*, the software part for PIC18F4550 has been created in the development environment MPLAB X IDE (*Integrated Development Environment*) (Figure 3-1), also a *Microchip Technology* product.

MPLAB X IDE is a software program, which can be installed on any computer and it is produced for Windows, Linux and Mac operational systems. IDE tells that this software is a proper environment for devices which are already incorporated in a circuit. In other words, MPLAB X IDE is suitable to develop code – which is the intelligence of embedded system applications – for embedded microcontrollers.

This development environment allows the user to write embedded code in C or Assembly programming language, to run and debug the code step-by-step, (instruction by instruction), to view the memory and the values of the variables stored in microcontroller.

“An embedded system is typically a design that uses the power of a small microcontroller. These microcontrollers combine a microprocessor unit (like the CPU in a personal computer) with

some additional circuits called peripherals, plus some additional circuits, on the same chip to make a small control module requiring few other external devices. This single device can then be embedded into other electronic and mechanical devices for low-cost digital control.” (Microchip , 2011-2015).



**Figure 3-1. Screen shot with MPLAB X IDE**

The typical tasks for developing an embedded controller application are:

1. Decide what microcontroller will be used for application
2. Determine what peripherals and pins are necessary
3. Design the associated hardware circuit
4. Choose the programming language and compiler
5. Write code
6. Compile, assemble and link the software using the assembler and/or compiler and linker to convert your code into “ones and zeros”
7. Test the code
8. Upload the code in microcontroller

“Of course, each of these steps can be quite complex. The important thing is to concentrate on the details of your own design, while relying upon MPLAB X IDE and its components to get through each step without continuously encountering new learning curves.” (Microchip Technology Inc., 2012-1014).

### 3.2 XC8 COMPILER

“Compiler” is called a software tool which translates the source code, written by the user in C or Assembly language, into machine code, which can be understood by a microcontroller.

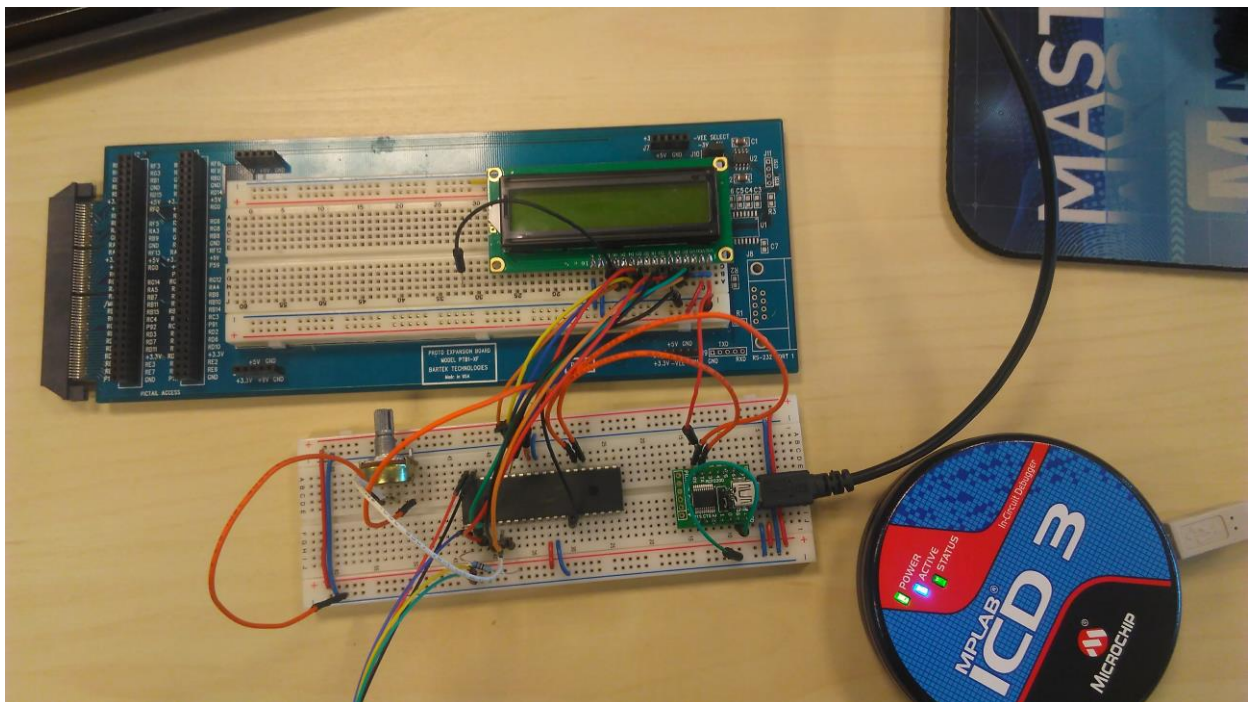
“The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language or machine code). If the compiled program can run on a computer whose CPU or operating system is different from the one on which the compiler runs, the compiler is known as a cross-compiler. More generally, compilers are a specific type of translator.” (Wikipedia, 2017)

XC8 Compiler is suitable for 8 bit PIC microcontroller and it is integrated in MPLAB X development environment. This compiler contains special instructions for every 8 bit device.

### 3.3 ICD3 PROGRAMMER/DEBUGGER

“The MPLAB ICD 3 In-Circuit Debugger is an in-circuit debugger that is controlled through a PC running MPLAB X IDE software on an OS. The application usage can vary from software development to hardware integration. The MPLAB ICD 3 In-Circuit Debugger is a complex debugger system used for hardware and software development of specific microcontrollers.” (Microchip, 2012-2014)

For this project, MPLAB ICD3 is used to test the program on breadboard circuit, like it is showed in Figure 3-2.

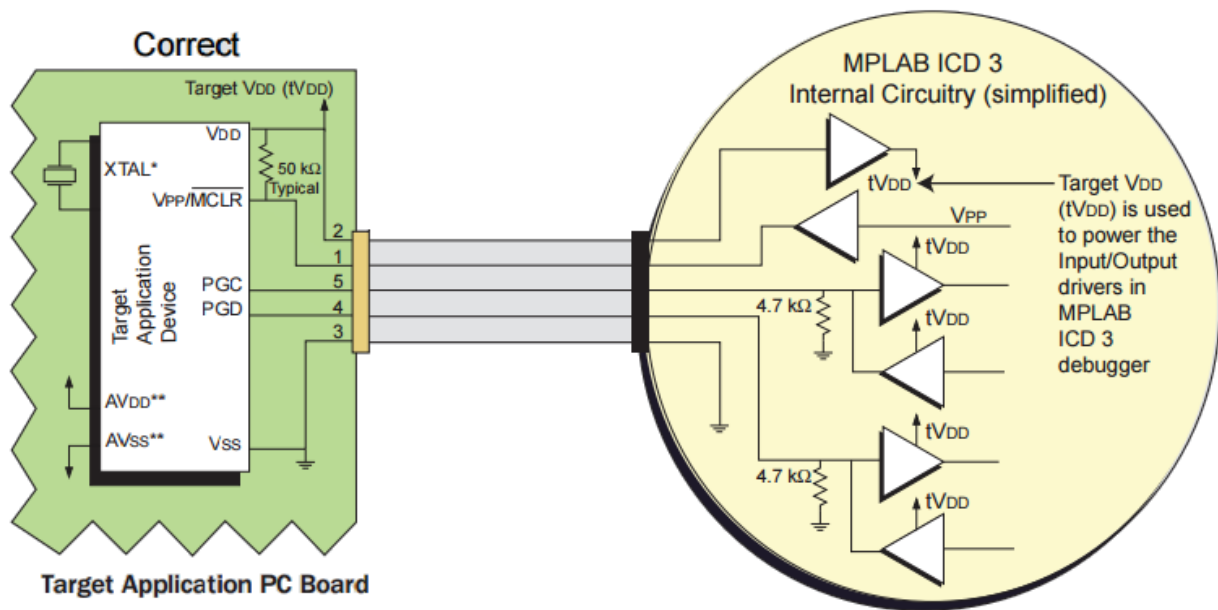


**Figure 3-2. Breadboard simulation circuit**

During the code testing, there have been damaged three Breakout Module and five pics. The ICD3 has pull-up protection on the lines, so it was a little hard to be damaged.

The breadboard circuit was powered from ICD3, and through USB Breakout Module MCP220 and USB cable, connected to the PC. The ICD3 was connected through an USB cable to the laptop. The laptop has on every port a short protection, which takes care that the laptop won't be burn. On the laptop was connected Breakout Module MCP220, which is using microcontroller

similar with PIC18F4550, it can lead the current through his diodes outside the PIC, when it is created a current loop, these 2 PIC have been burned.



**Figure 3-3. Correct connection to ICD3 and a simplified internal circuit (Microchip, 2012-2014)**

“The components of the MPLAB ICD 3 In-Circuit Debugger system are:

- MPLAB ICD 3 with indicator lights
- USB cable to provide communications between the debugger and a PC and to provide power to the debugger
- Cable to connect the MPLAB ICD 3 to a header module or target board
- ICD 3 Test Interface Board” (Microchip Technology Inc., 2012-1014)



**Figure 3-4. ICD3 and RJ-11 adaptor**

### 3.4 MAIN FUNCTION

The program is structured in 2 parts:

- Main Code (Figure 3-6.)
- I/O interrupts (Figure 3-5.)

PIC18F4550 has a feature called “interrupt-on-change” on some pins of the PORTB. This feature allows the microcontroller to have an additional external interrupt source. This feature is

triggered when any of the RB7:RB4 pins, configured as inputs, change level. For this application the interrupt source is a push button, connected on RB5. With this feature it is possible to change between Demo Code and Detection mode when running the application.

“An interrupt is a way of letting the controller knows that something important happened without having it to look at the possibility each time. This saves a lot of time; time spent in checking for that event. Besides saving time, it is more accurate in reading that event without any sampling issues.

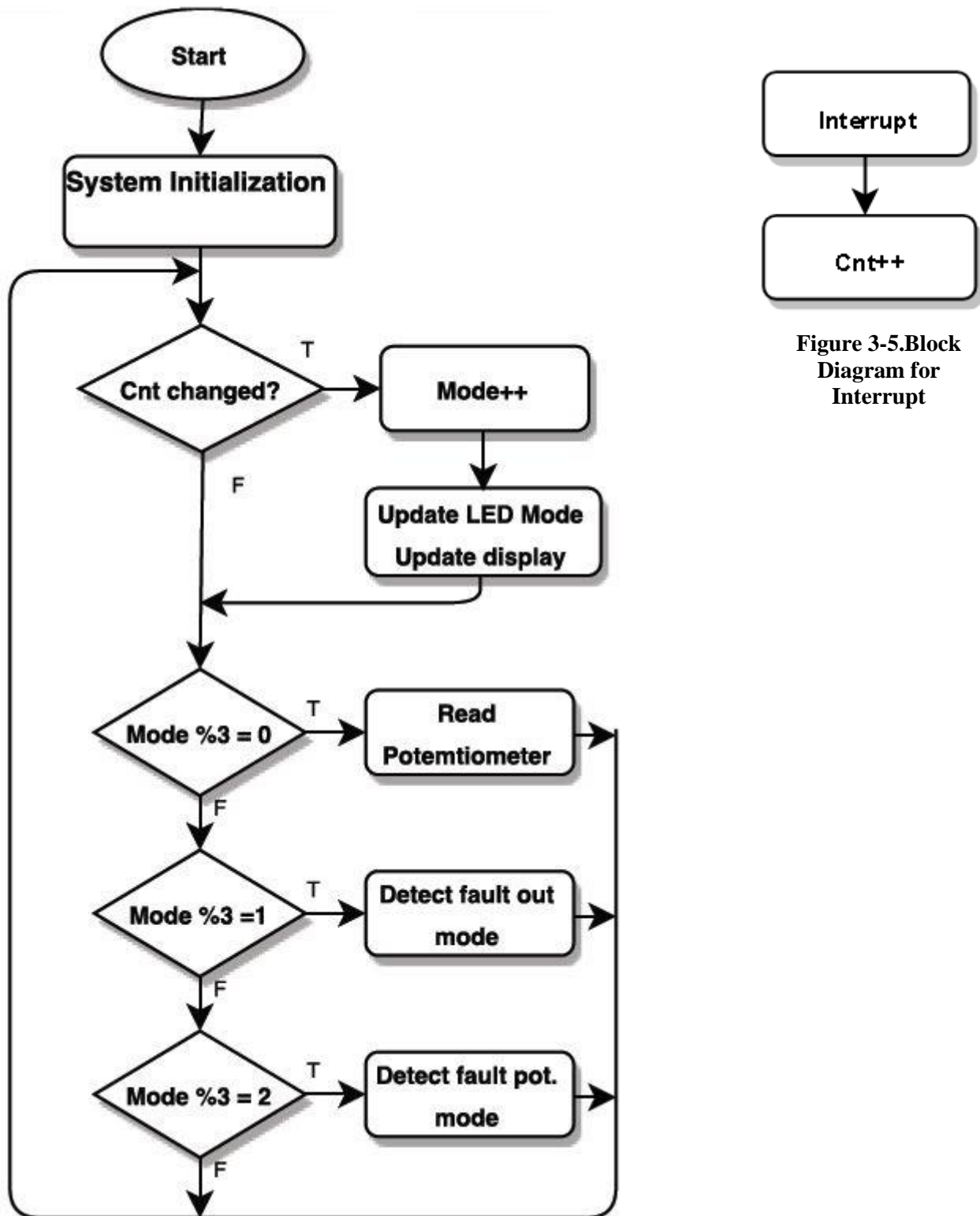


Figure 3-5. Block Diagram for Interrupt



**Figure 3-6.Main Block Diagram**

The idea is this, either you can keep looking at the power indicator on the wall and check if the power is back or switch on the fan and be notified once it is there.” (Siddharth, 2013)

As it can be observed in the block diagram, after the application starts, system initialization is required. Here are called the initialization function for: LCD, UART and ADC. In addition the interrupts are enabled, in order to be able to detect when the button is pressed, and a starting text is displayed on LCD and transmitted on UART.

Second part is an infinite loop. With the help of a “switch” instruction, the application can be made to execute the detection code, or the Demo Code.

*DetectMode1* is testing a pin connected to a LED. It will be discussed more in Chapter 2.6. *DetectMode2* wants to test a pin connected to a potentiometer.

### 3.5 READ POTENTIOMETER

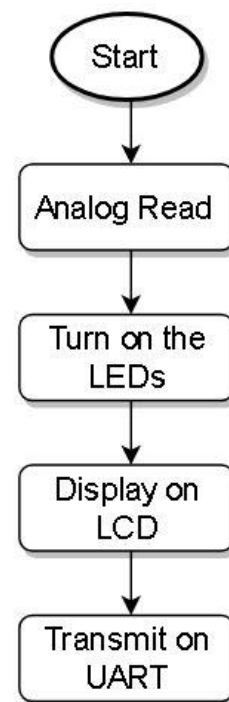
Because the purpose of this project is to create a software method to detect the faulty pins, a simple Demo Code was created in order to observe if this additional software will have any impact on the principal purpose of the microcontroller.

In Figure 3-7 can be observed the block diagram for Demo Code. On hardware there is a potentiometer, which is connected to a pin with an analog function on it. The analog value of the pin is read, then converted and transmitted to three peripherals.

One of them is UART module, which will send the data to the laptop and then to the RS232 terminal (Hercules).

The same data is displayed on the LCD and the third peripheral is composed by 8 output LEDs, on which will be displayed the digital 8 bit value.

Demo Code is a simple one, because it tests if the used peripherals for detection are somehow affected. The results will be discussed in the next chapter.

**Figure 3-7.Demo Code Block Diagram**

### 3.6 DETECT FAULT

In the main function block diagram, there are 2 modes of detecting the fault. First one is testing the fault on a *digital output pin* (a LED).

The second tested pin, in normal conditions, is an *analog input* (a potentiometer). The detection on a digital output pin will be described first.

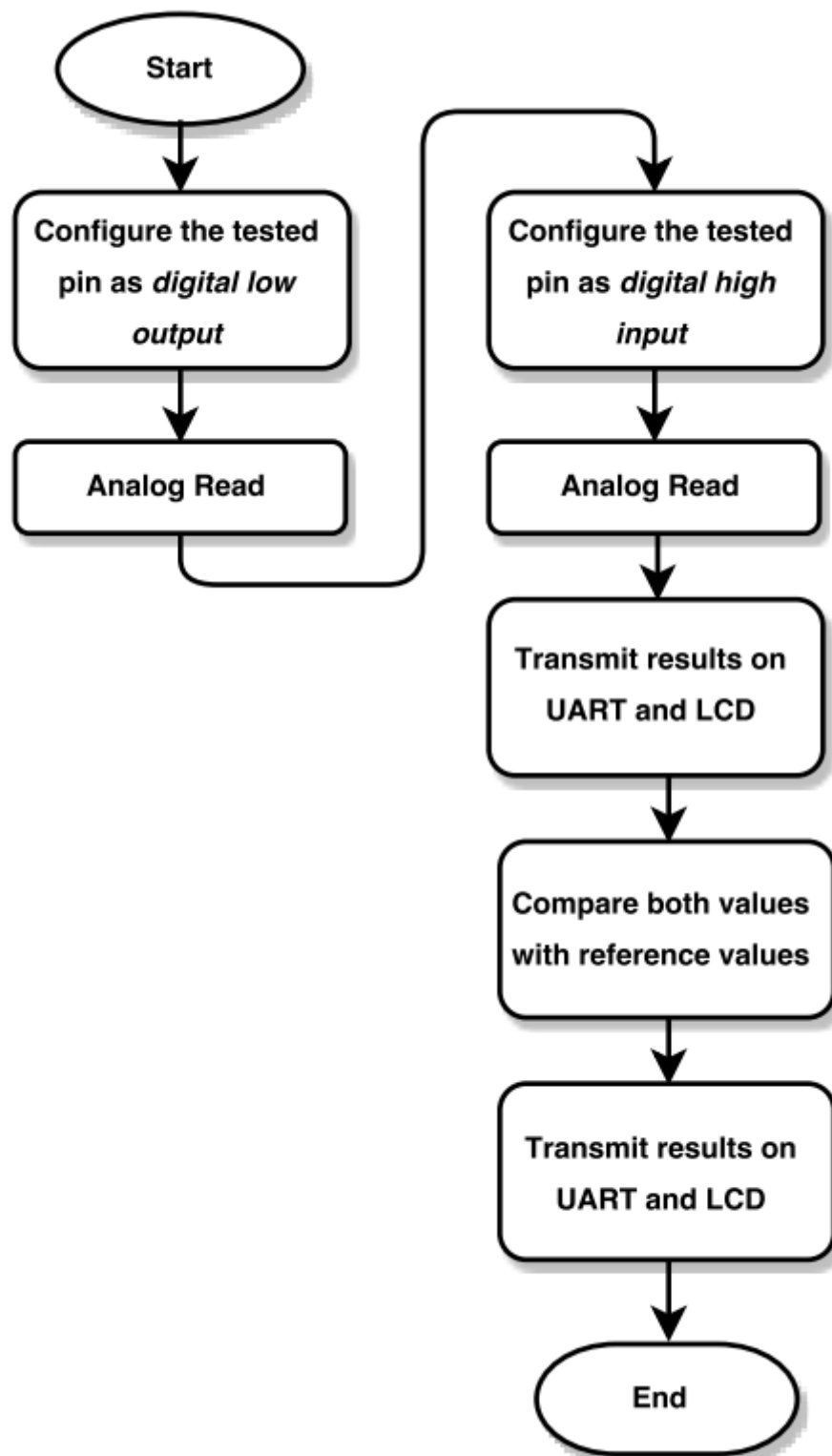
In Figure 3-8 it is showed the block diagram for detection software. It was specified before that for this project the tested pins have been chosen those who have and ADC function on it.

Before tests begin, a starting message which announces the user which pin is tested will be displayed on LCD and transmitted on UART. Also, almost in the same time, when the button is pressed a corresponding LED will let us know that the specific detection part has begun.

Because it is a software method, only internal microcontroller resources have to be. Because a pin is tested, there are the simplest ways of controlling the pin. The pins registers are: PORT register, TRIS register and LAT register.



TRISx is Data Direction Control register. “The TRISx register controls the direction of each pin in PORTx. For example, TRISB is the direction control for PORTB. If the TRIS bit for an I/O pin is ‘1’, then the pin is an input.” (Microchip, 2008)



**Figure 3-8.Detection Block Diagram**

The LAT register holds the value written to the PORT irrespective of the voltage level on the port pins.

PORT register represent the state of all 8 pins. With this register, it can set the pin low or high. It is also used to read the value on the pin, but the drawback is that the microcontroller first read all 8 bits of the respective PORTx and it will be read the actual state of all 8 pins, just like on the pins it is a logic probe. But it is not a very used method because this measurement can be influenced by some components.

In order to avoid this confusion, it has been chosen that the pin to be read with the ADC module. There will be two analog measurements. TRIS allows setting the pin as input or output and PORT is changing the pin's. It was tried to detect a fault when the pin is *digital input high* for the first measurement, and *digital input low*, but there wasn't enough information in order to detect if it is a fault or not.

Another try was the combination: *digital output high* and *digital output low*, but there wasn't enough to detect the type of the fault.

During the tests, it has been observed that for normal functioning conditions there are standard values, but these values are different depending on what is connected on the respective pins. As an example in Table 3-1 the reference values for a LED are showed.

	Minimum Value	Maximum Value
<b>High</b>	1.2 V	1.4 V
<b>Low</b>	0 V	0.5 V

**Table 3-1.Values observed during tries**

After another tries, it was decided that two measurements are enough. After this conclusion, and measurements during the fault (short to Vdd, short to GND, open), it was established the values presented in Table 3-2.

	Input	Output
<b>Short to Vdd [V]</b>	<b>4.91</b>	<b>4.91</b>
<b>Short to GND [V]</b>	<b>0</b>	<b>0</b>
<b>Open (Interrupt) [V]</b>	<b>0</b>	<b>4.91</b>

**Table 3-2.Reference values for faults**

First measurement is when the pin is configured as a *digital low output*.

Second step is to read analog the pin, when it is a *digital high input*. This value is also displayed on LCD and transmitted through UART.

After these measurements, the two values are transmitted on UART and displayed on LCD.

As it is figured in the block diagram (Figure 3-8.) the next step is to **compare the analog values with reference values**. The type of the fault is detected, displayed on LCD and transmitted through UART to the RS232 Terminal.

With the values presented in Table 3-1 and Table 3-2 it was possible to detect the fault and also what type of fault is present.

Like an observation, the working principle is the same for any pin which has analog function on it. The difference is that the values for normal conditions, depends on what is connected right next to the pin.

In Figure 3-9 it can be observed that for different type of fault there are different values displayed.

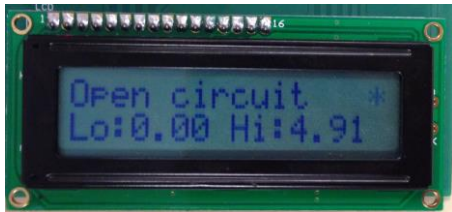
For the second detect fault mode, it was decided that the value of the potentiometer must be different of zero or maximum.



**a) Short to GND**



**b) Short to Vdd**



**a) Open Circuit**



**b) Without Fault**

**Figure 3-9. Fault Values Displayed on LCD**



# CHAPTER 4

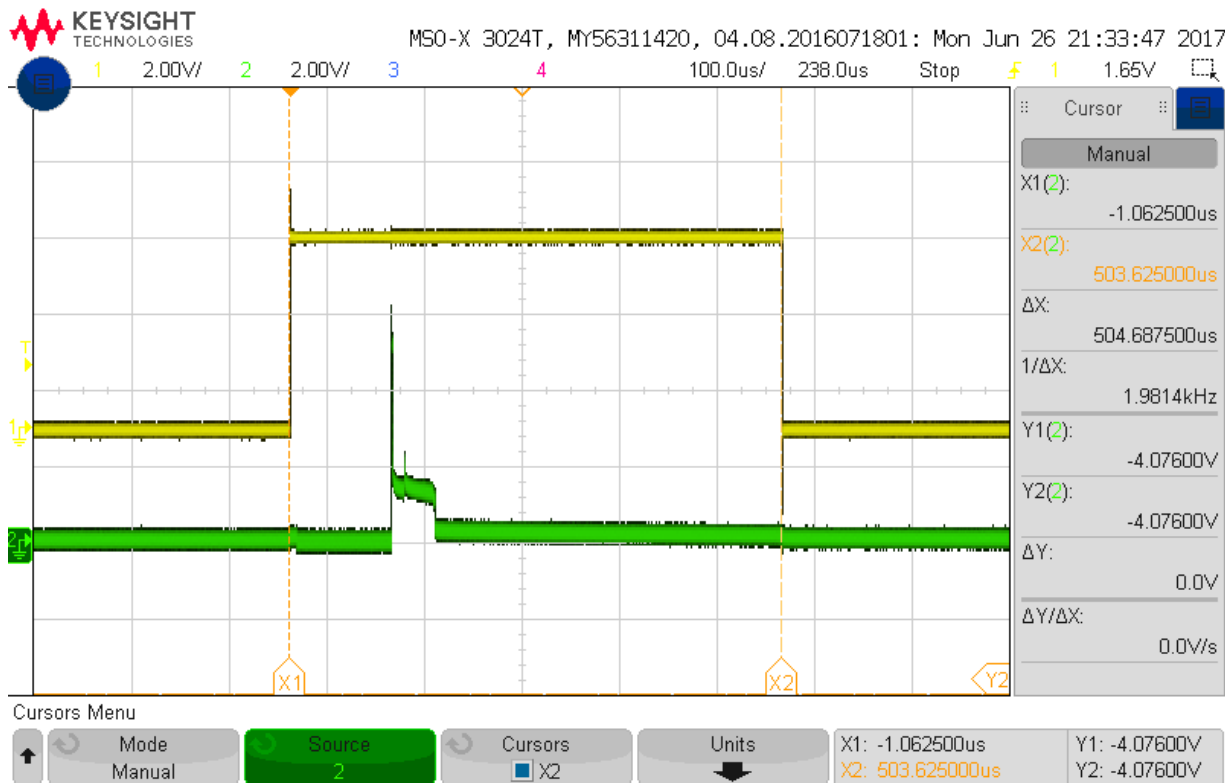
## MEASUREMENTS

The purpose of this project is to be integrated in any microcontroller, so it was needed to see if the additional code has any impact on the main application of the PIC. Also the most important aspect is to observe how time it takes to detect these types of fault.

In Figures from 4-3 to 4-8 is measured the time during analog reads and the time from starting the measurements and displaying the results. These measurements were made for the latest version of the software code, where some delays reduced or erased. It was tried to see which the minimum delays for this project is. In the final version, it can be found the delays in displaying a character on LCD; it is used in order to combat the debouncing of the button; also for allow the microcontroller to change the analog channel (it will be accessed once at the start of every detect fault mode); and the bigger one is used in *Read Potentiometer* mode, because the time of displaying the value on LCD was too fast, and it was hard for the user to extract useful information.

These delays modifications were possible because in the code the part of transmission on UART and displaying on LCD had been separated by the measurements and verification, in order to keep the similar parts together.

An advantage of separating the transmission is that it can be measured only the time between measurements for detecting. It was helpful because it can be easily adapted for other microcontrollers. Another advantage appears when it is enough only one type of transmission; it can be made, just by deleting some instructions.



**Figure 4-1. Functional mode, detecting time**

For further comparisons, in Figure 4-1 is showed the oscilloscope screen shot for a digital output pin, in normal working conditions. All these measurements are made for a digital output pin.

In order to be able to make these captures, it was used a trigger pin, like it is described in Figure 4-2. In this case it has been used RB2 pin. The commented lines are the measurements instructions.

```
LATBbits.LATB2 = 1;
TRISBbits.RB2 = 0;
/*...15 lines */
LATBbits.LATB2 = 0;
TRISBbits.RB2 = 0;
```

**Figure 4-2. Code instructions**

On every oscilloscope screen shot, there are figured the vertical cursors, and in the right of the image it can be observed “ΔX”, which calculate the difference between cursors X1 and X2, in order to measure more precisely the time.

For measurements screen shots, the green signal is the one measured at the tested pin, and the yellow one is the trigger signal.

The screen shots where it is visualized the detecting time, composed by the measurements time and displaying the results on LCD and RS232 terminal. If from these results are subtracted the initial time measurements, it can be observe that the time used for displaying the results is with two orders of magnitude bigger.

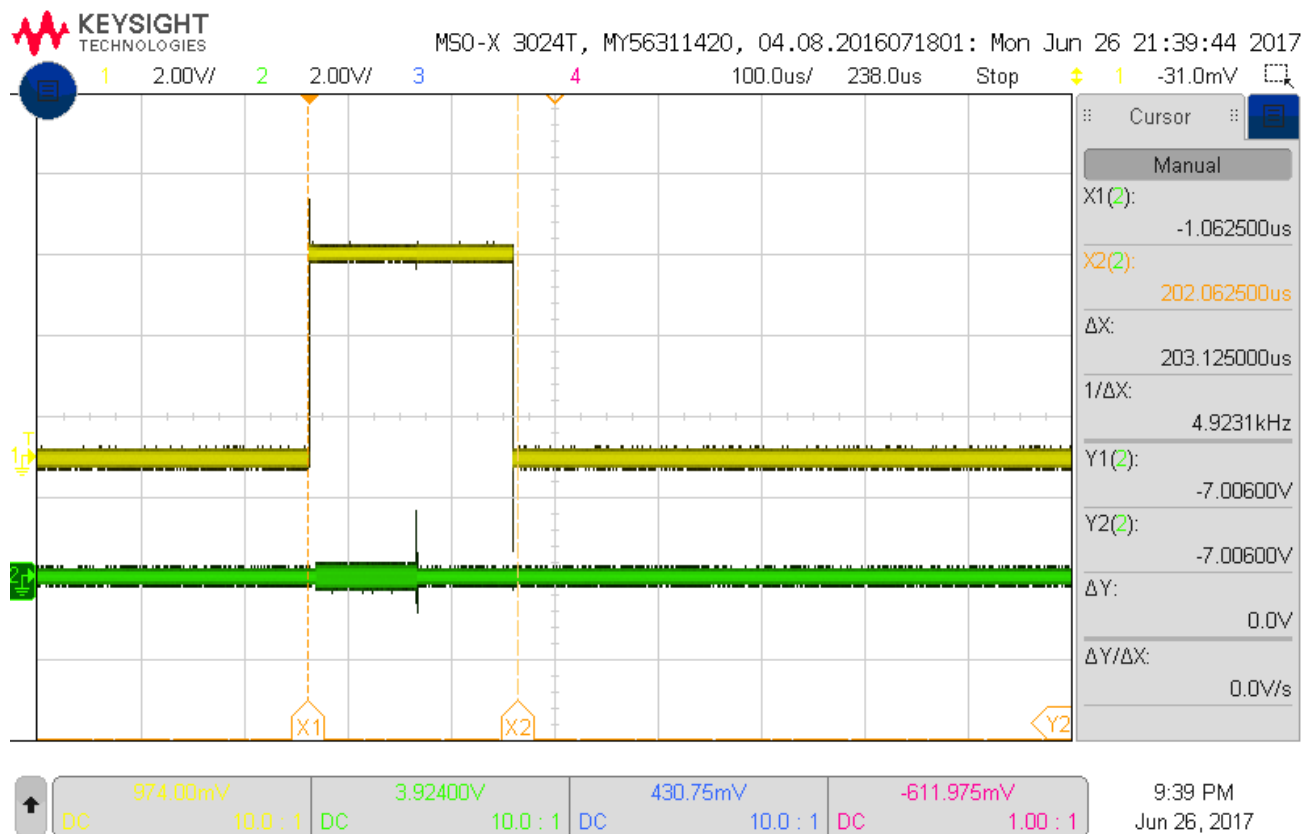


Figure 4-3.GND measurements time

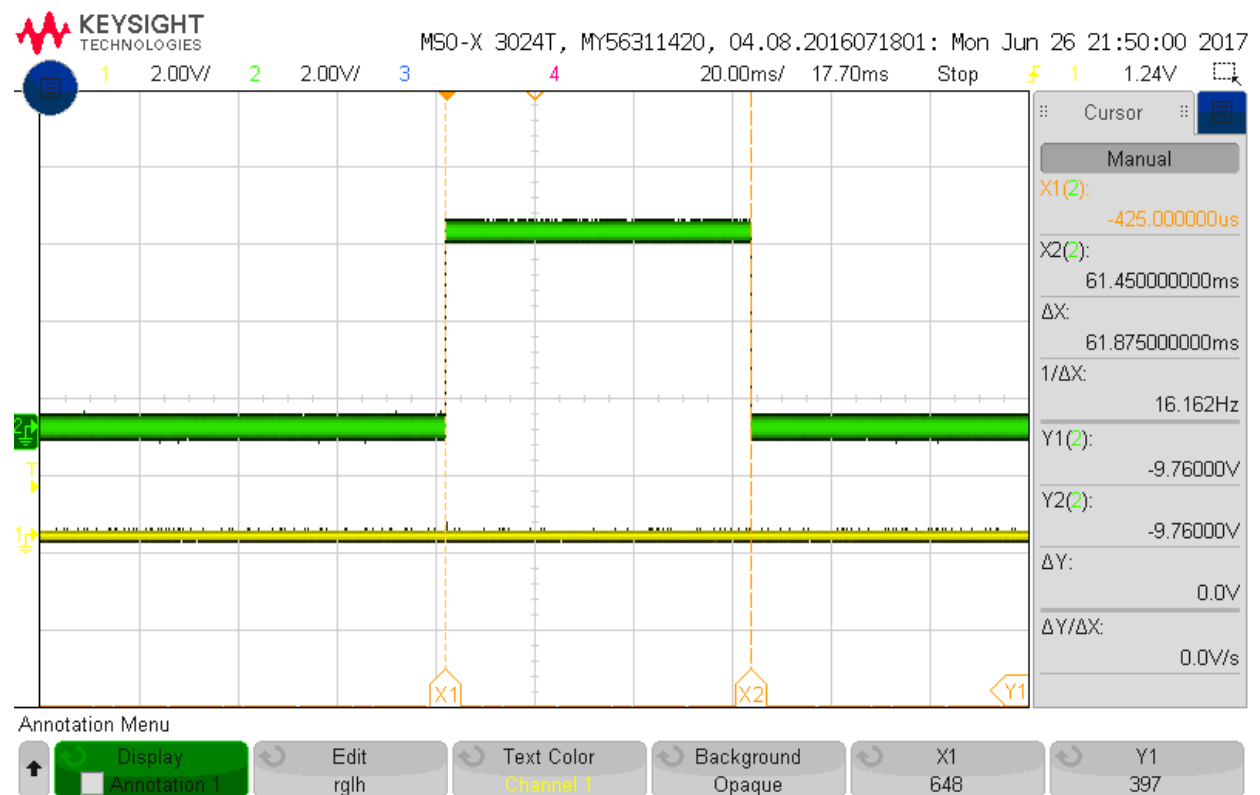


Figure 4-4.GND detecting time

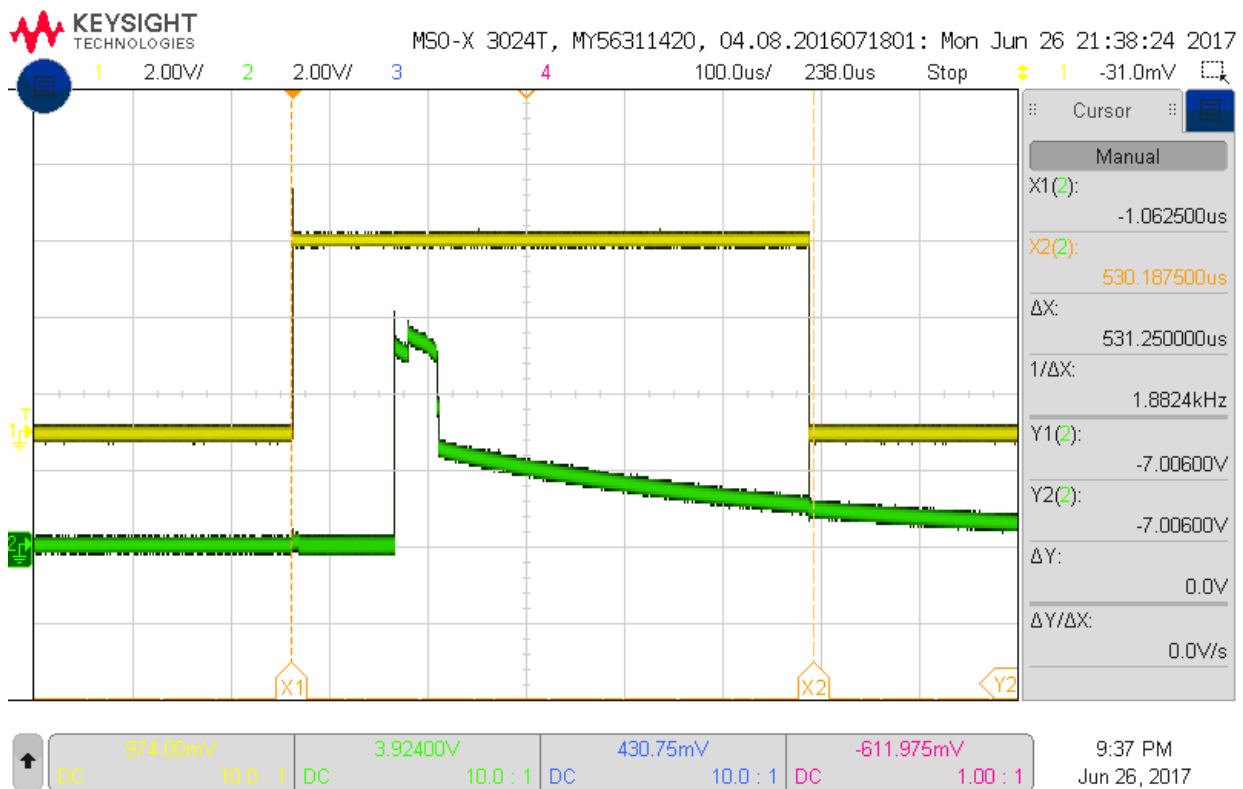


Figure 4-5.Open circuit measurements time

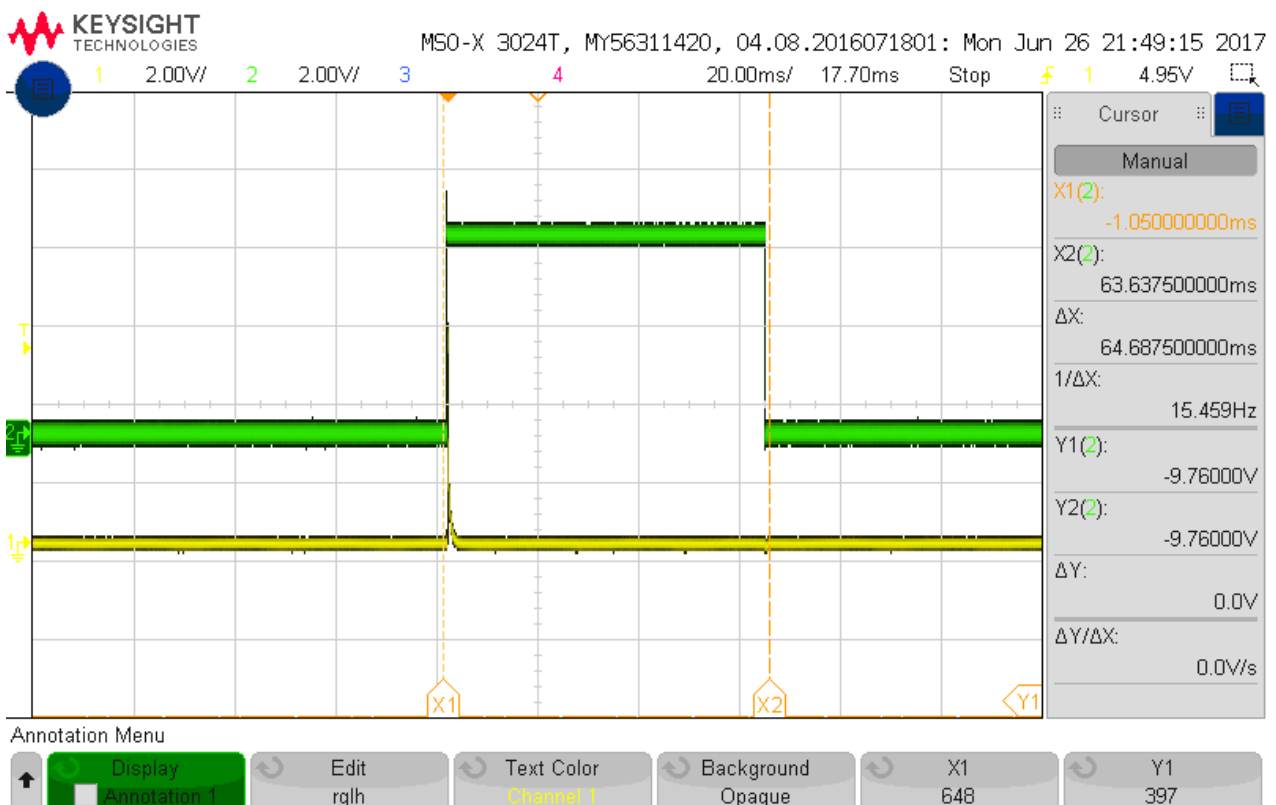


Figure 4-6.Open Circuit detecting time



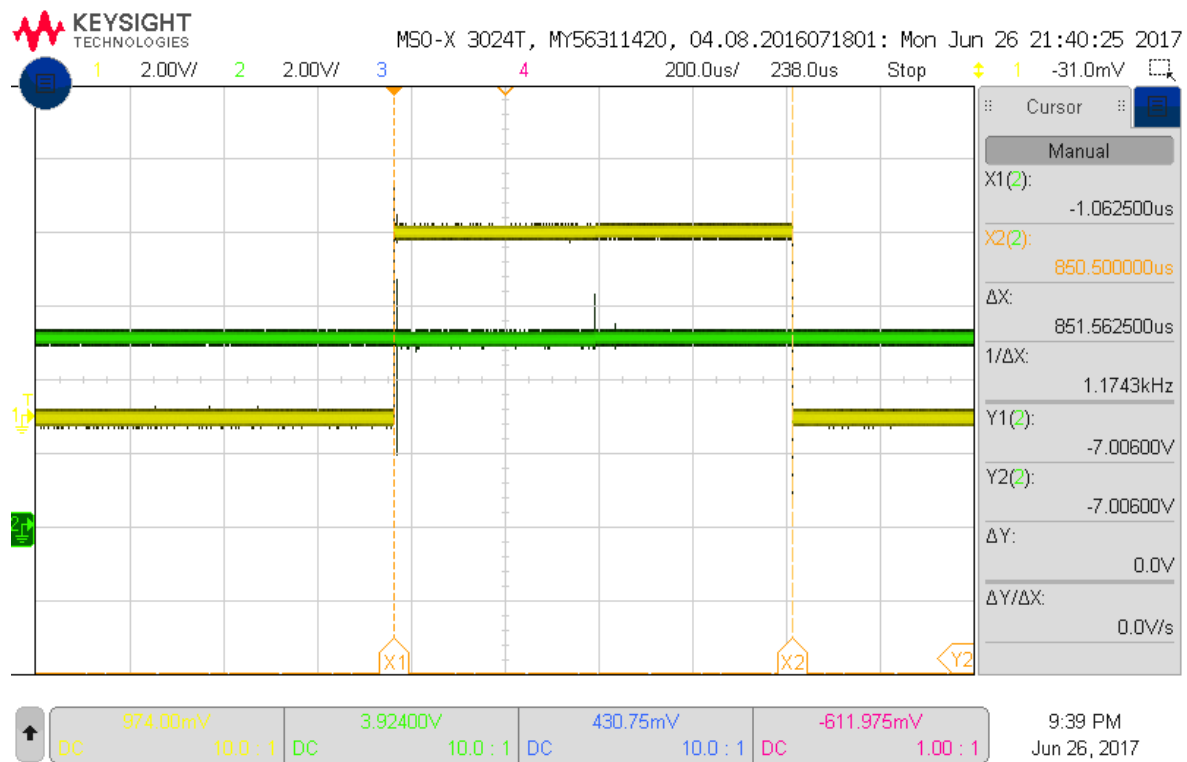


Figure 4-7.Vdd measurements time

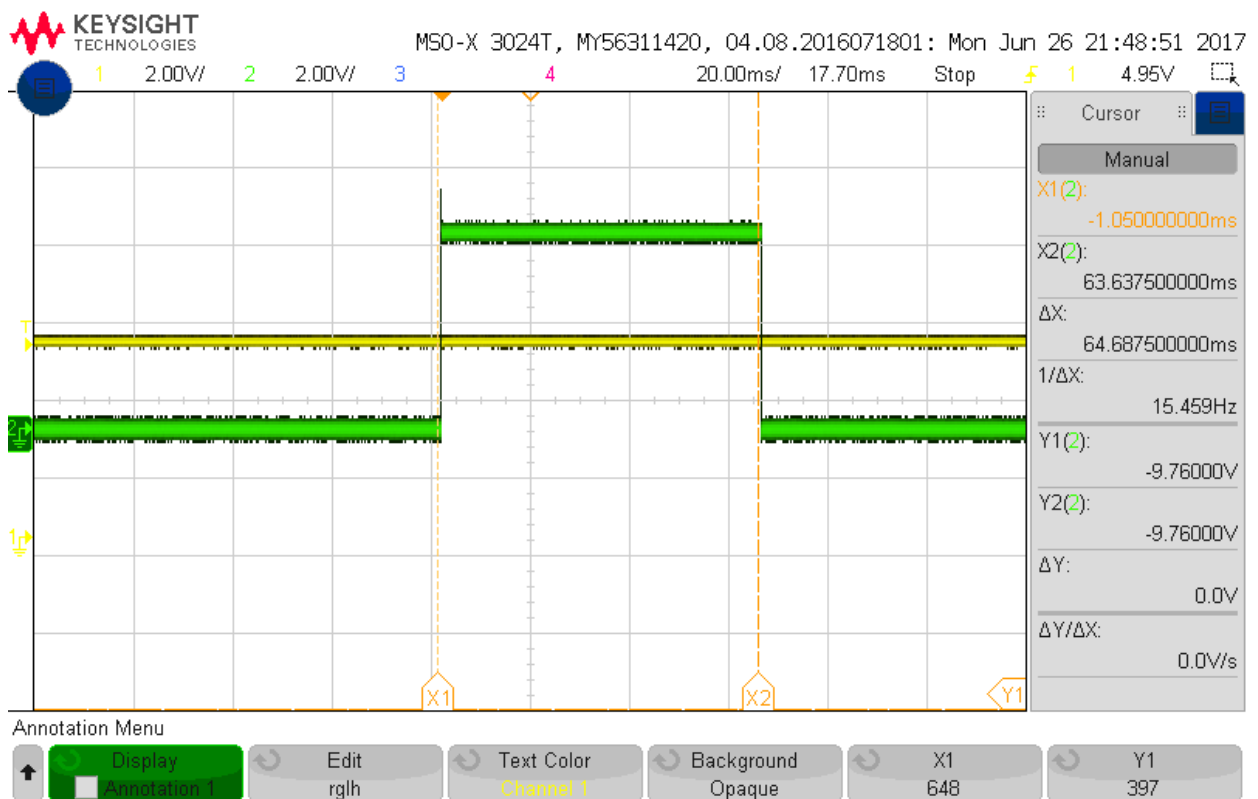


Figure 4-8.Vdd detecting time



# CONCLUSIONS

## INITIAL EXPECTATIONS AND FINAL RESULTS

At the start of this project I wanted to make an embedded code, reliable on any PIC microcontroller, which it should be run before the start of the application, in order to test all 40 microcontroller's pins. I wanted that every pin to be tested on following fault types: short to Vdd, short to GND, open circuit and another common fault, especially for SOIC package, if two adjacent pins are soldered together by mistake. This was my vision about this diploma project.

When I started to write the code for my vision project, I started to initialize every peripheral which would be further used. Second part was to create a single project from all these together. This part of the project took me the most time, but I learned how to program and control different peripherals of a PIC microcontroller. This first project is called *ReadAD* in the final project.

About the detection part, the results were different of my expectations. I wanted to test all 40 pins, but I found that the pins are used for different peripherals, there are different configurations, and there are different components connected directly to the pin.

Because the project is a software one, I had to limit on what methods of controlling the pins are available. These methods are that a pin can be digital (I/O) or analog. If the microcontroller make a digital read there is not useful information to detect the fault, because there are digital values. So I decided to use the analog function to test the pins.

My first idea of detecting a fault was to connect an analog pin to the others and test them one by one. But in these conditions the Vdd and GND pins were linked together. Even if I would

eliminate these special pins, this hardware connection would have an unexpected influence in normal working conditions.

The second idea was to search if there is any possibility to internally connect and disconnect one analog pin to all other pins. The PIC used for the project doesn't allow this internal connection.

The final idea was to test pins with their analog function. Because not all pins have analog function, the pins that don't have analog function were eliminated from the tested pins. The remained situations are when the pins are configured as: digital input, digital output or analog input pins.

At the end I was able to detect three types of fault (short to Vdd, short to GND, open circuit) for a digital output and analog input.

## PERSONAL CONTRIBUTION

For this project, my personal contribution was to design the hardware board, because I needed to have test points in order to simulate a fault; to select the required components, to create the development board, to create the software code. As a contribution can be the creation of block diagrams that are discussed in Chapter 3.

For the designed board, I created the electric schematic, layout and Gerber files in Eagle software. This software allows the user to create their components libraries. I created library for MCP2200 Breakout Module. This library consists in creating the symbol, the device and the package with the dimensions specified in data sheet. I have also implemented the required connections for the rest of the components.

The software code was designed by consulting the datasheet and respecting the requirements and the possibilities of each interface. The software for detection was also created by me. There were multiple measurements and implementation, described earlier, until the final results.

## REFERENCES

- Details of PIC ICSP and how to use it for pic microcontrollers.* (2005). Retrieved 2017, from Best-Microcontroller-Projects: <http://www.best-microcontroller-projects.com/pic-icsp.html>
- Durda, F. (2014, 04). *Serial and UART Tutorial*. Retrieved 2017, from Freebsd: [https://www.freebsd.org/doc/en\\_US.ISO8859-1/articles/serial-uart/](https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/)
- Eagle.* (n.d.). Retrieved from CadSoft: <http://www.cadsoft.de>
- Eagle.* (2017, May). Retrieved 2017, from Wikipedia: [https://en.wikipedia.org/wiki/EAGLE\\_\(program\)](https://en.wikipedia.org/wiki/EAGLE_(program))
- Eurocircuits. (n.d.). *Eurocircuits - fast and easy PCB procurement*. Retrieved April 2017, from Eurocircuits : <http://be.eurocircuits.com/shop/ecbasket/services.aspx>
- George, L. (2015, July 10). *Using ADC Module of PIC Microcontroller – Hi Tech C*. Retrieved 2017, from Electroome: <https://electrosome.com/adc-pic-microcontroller-hi-tech-c/>
- Gottardo, P. D. (1999-2013). *Course in PIC Microcontrollers Programming*. Retrieved 2017, from <http://www.hobbyprojects.com>: <http://www.hobbyprojects.com/projects/marco-gottardo-projects/lets-go-pic-digital-input-output-chap3.html>
- HW group. (n.d.). *Hercules SETUP utility*. Retrieved 2017, from HW group: [http://www.hw-group.com/products/hercules/index\\_en.html](http://www.hw-group.com/products/hercules/index_en.html)
- MCP2200 USB-to-UART Serial Converter.* (2010, 07 15). Retrieved 2017, from Digi-Key Electronics: <https://www.digikey.com/en/product-highlight/m/microchip-technology/mcp2200-usb-to-uart-serial-converter>
- Microchip . (2011-2015). *MPLAB® X IDE User's Guide*. Retrieved 2017, from Microchip: <http://ww1.microchip.com/downloads/en/DeviceDoc/50002027D.pdf>
- Microchip. (2008). *Read-Modify-Write (RMW) Problem with Mid-Range PIC Microcontrollers*. Retrieved 2017, from Microchip: <http://ww1.microchip.com/downloads/en/devicedoc/31009a.pdf>.
- Microchip. (2009). *PIC18F2455/2550/4455/4550 Data Sheet*. Microchip.
- Microchip. (2009). *PIC18F4550*. Retrieved April 2017, from Microchip Technology Inc.: <http://www.microchip.com/wwwproducts/en/PIC18F4550>
- Microchip. (2012). *MCP2200 Breakout Module User's Guide*. Retrieved 2017, from Microchip: <http://ww1.microchip.com/downloads/en/DeviceDoc/52064A.pdf>

- Microchip. (2012-2014). *MPLAB® ICD 3 USER'S GUIDE*. Retrieved 2017, from Microchip: <http://ww1.microchip.com/downloads/en/DeviceDoc/50002081B.pdf>
- Microchip. (2017). *MCP2200*. Retrieved 2017, from Microchip: <http://www.microchip.com/wwwproducts/en/en546923>
- Microchip Technology Inc. (1997). Section 9. I/O Ports. In *PICmicro MID-RANGE MCU FAMILY*. Microchip Technology Inc.
- Microchip Technology Inc. (2012-2014). *MPLAB® ICD 3 User's Guide for MPLAB X IDE*. Retrieved 2017, from Microchip: <http://ww1.microchip.com/downloads/en/DeviceDoc/50002081B.pdf>
- MicrocontrollerBoard. (2008). *PIC serial communication tutorial*. Retrieved 2017, from MicrocontrollerBoard: [http://www.microcontrollerboard.com/pic\\_serial\\_communication.html](http://www.microcontrollerboard.com/pic_serial_communication.html)
- Nisbett, D. (2012, July). *http://www.analog.com*. Retrieved June 1, 2017, from Analog Dialog: <http://www.analog.com/en/analog-dialogue/articles/diagnostic-technique-detects-open-and-short-circuits.html>
- Siddharth. (2013, August 27). *Interrupt On Change (IOC) in PIC Microcontrollers*. Retrieved 2017, from Embed Journal: <https://embedjournal.com/interrupt-on-change-ioc-in-pic-microcontrollers/>
- Verle, M. (2008). *PIC MICROCONTROLLERS – PROGRAMMING IN ASSEMBLY*. MikroElektronika.
- Wikipedia. (2017, June). *Compiler*. Retrieved 2017, from Wikipedia: <https://en.wikipedia.org/wiki/Compiler>

**Annex1**  
**schema**

## **Annex 2**

### **Layout**