P-2591

SIMULATION OF OSCILLOSCOPE

A PROJECT REPORT

Submitted by

A. RAGHAVENDRA 71205205040

N. SATHISH 71205205050

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

MAHATMA GANDHI · CENTRAL LIBRARY · KCT CBE-6

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2009

P-2591

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**SIMULATION OF OSCILLOSCOPE**" is the bonafide work of "**RAGHAVENDRA A and SATHISH N**" who carried out the project work under my supervision.

SIGNATURE

Dr. S. Thangasamy

DEAN

Department of
Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore – 641006.

SIGNATURE

Mrs. B. Aruna Devi.,

SUPERVISOR

Department of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641006.

The candidates with University Register Numbers **71205205040 &** **71205205050** examined by us in the project viva-voce examination held on

28th APRIL, 2009

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere thanks to our chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc., F.I.E.,** Vice Chairman **Dr. K. Arumugam B.E., M.S., M.I.E.,** Correspondent **Shri.M.Balasubramaniam** and **Joint Correspondent Dr. A. Selvakumar** for all their support and ray of strengthening hope extended. We are immensely grateful to our principal **Dr.Joseph V Thanikal, B.E., M.E., Ph.D., PDF., CEPIT.,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr. S. Thangasamy, B.E.(Hons), Ph.D.,** Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during course of this project

We also extend our heartfelt thanks to our project coordinator **Asso Prof.L.S.Jayasree Ph.D.,** Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide **Mrs. B. Aruna Devi, M.E.,** Lecturer, Department of Information Technology, for her everlasting counseling and untiring help throughout this project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project.

Finally we thank **the Almighty, our parents, family members and friends** for the blessings they have showered upon us.

DECLARATION

We,

A. RAGHAVENDRA	**71205205040**
N. SATHISH	**71205205050**

Declare that the project entitled **"SIMULATION OF OSCILLOSCOPE"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) degree, is a record of original work done by us under the supervision and guidance of **Mrs. B. Aruna Devi M.E.,** Lecturer, Department of Information Technology, Kumaraguru College of Technology , Coimbatore.

Place: Coimbatore

Date:

[A. Raghavendra]

[N. Sathish]

Project Guided by

[Mrs.B. Aruna Devi., M.E.]

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vii
	LIST OF TABLE	viii
	LIST OF FIGURES	ix
	LIST OF SYMBOLS	x
1.	**INTRODUCTION**	1
	1.1 The Existing System	2
	1.2 The Proposed System	3
2.	**PROJECT PLAN AND SPECIFICATION**	4
	2.1 Project Analysis	5
	2.2 Project Plan	6
	2.3 Hardware Specification	8
	2.3.1 Component Description	9
	2.3.2 PIC Description	13
	2.3.3 Microcontroller Advantages	19
	2.4 Software Specification	20
	2.4.1 MPLab Utility Description	20
	2.4.2 HITECH C Compiler Description	23
	2.4.3 User Interface Description	26
3.	**SYSTEM FUNCTIONALITY**	30
	3.1 Hardware Functionality	31
	3.1.1 Flashing Microcontroller Code	31
	3.1.2 Circuit Operation	32
	3.2 Software Functionality	33
	3.2.1 Acquire values from the controller	33
	3.2.2 Plotting the graph	35

CHAPTER NO.	TITLE	PAGE NO.
4.	**ADVANTAGES AND LIMITATIONS**	36
	4.1 Advantages	37
	4.2 Limitations	38
5.	**PRODUCT TESTING**	39
	5.1 Test Outputs	42
6.	**FUTURE ENHANCEMENTS**	45
7.	**APPENDIX**	47
	7.1 Microcontroller Code	48
	7.2 User Interface Code	64
8.	**REFERENCES**	98

ABSTRACT

The PC Based Digital Oscilloscope provides a comprehensive solution that can allow signal voltages to be viewed as a two-dimensional graph of one or more electrical potential differences (vertical axis) plotted as a function of time or of some other voltage (horizontal axis) using a Windows Based PC or a Windows Based Laptop. The signal values from a circuit is fed into the computer using USB port which is accompanied by a protective circuit that could remove any harmful signals from the circuit whose voltage needs to be tested which could be harmful to the computer's interfacing circuits. If we integrate the system with a PC, it is possible to acquire all the advantages, like portability of the system, cost-efficient and utilization of other features like signal store helps to make use of the system to a larger extent than a traditional Oscilloscope.

LIST OF TABLES

S. NO	TABLE NO.	TABLE NAME	PAGE NO.
1.	Table 2.3.2.1	PIC I/O PINOUT DESCRIPTION	17
2.	Table 2.3.2.2	I/O PINOUT DESCRIPTION (cond.)	18

LIST OF FIGURES

S. NO	FIGURE NO	FIGURE NAME	PAGE NO
1.	Fig 2.2.1	Hardware Functional Block Diagram	6
2.	Fig 2.3.1.1	LF353 Pin Diagram	10
3.	Fig 2.3.1.2	ICL7660A Pin Diagram	11
4.	Fig 2.3.1.3	MCP6S91 Pin Diagram	13
5.	Fig 2.3.2.1	PIC18F2550 Pin Diagram	15
6.	Fig 2.3.2.2	PIC18F2550 Functional Block Diagram	16
7.	Fig 2.4.1	Signal Processing Software Unit	21
8.	Fig 2.4.2.1	Application Process Sequence	24
9.	Fig 2.4.2.2	Signal Processing Steps	25
10.	Fig 2.4.2.3	User Interface Screenshot	26
11.	Fig 2.4.2.4	Signal properties Screenshot	27
12.	Fig 2.4.2.5	Sampling Properties Screenshot	27
13.	Fig 2.4.2.6	Export signal Screenshot	28
14.	Fig 5.1.1	AC Input Screenshot	42
15.	Fig 5.1.2	DC Input Screenshot	43
16.	Fig 5.1.3	Square Wave Input Screenshot	44

LIST OF SYMBOLS

S. NO	SYMBOL	DESCRIPTION
1.	PIC	Peripheral Interface Controller
2.	USB	Universal Serial Bus
3.	CRO	Cathode-ray oscilloscope
4.	PC	Personal Computer
5.	I/O	Input/output
6.	AC	Alternating Current
7.	DC	Direct Current
8.	J-FET	Junction Field Effect Transistor
9.	KB	Kilo Bytes
10.	ADC	Analog to Digital Converter
11.	MB	Mega Bytes
12.	D/A	Digital to Analog Converter
13.	MHz	Mega Hertz
14.	CMOS	Complementary metal–oxide–semiconductor
15.	RC	Resistance – Capacitance
16.	PDIP	Plastic Dual-In-line Package
17.	SOIC	Small-Outline Integrated Circuit
18.	SPI	Serial Peripheral Interface
19.	PGA	Programmable Gain Amplifier
20.	MSOP	Mini Small Outline Package
21.	RISC	Reduced Instruction Set Computer
22.	USART	Universal Synchronous Asynchronous Receiver Transmitter
23.	RAM	Random Access Memory
24.	CPU	Central Processing Unit

INTRODUCTION

1. INTRODUCTION

An oscilloscope is considered to be one of the essential devices to test and measure signals that arise. The integration of computers and laptops with the present-day testing measurement & instrumentation world has opened up the door for "virtual instrumentation". When the testing equipments of a circuit are integrated with a system which supports portability, scalability and cost-efficient, it is much easier to test and debug a circuit with an improvised solution rather than a traditional system.

1.1 THE EXISTING SYSTEM:

A typical cathode ray oscilloscope is limited in various aspects, in-terms of cost, portability and maintenance. It would cost more to purchase a cathode ray oscilloscope and in terms of test and measurement purposes, we need a large number of them for e.g. in educational institutions. Also the size of a normal CRO is bigger when compared with other traditional testing equipments. Thus it is much costlier if the need of those instruments is in large volumes.

Also the Oscilloscopes which are not much costlier does not support additional features like signal store, support of more than one input channels and huge size. To avoid the constraints involved it is very much useful to provide a system that could overcome all these delimiters.

1.2 THE PROPOSED SYSTEM:

To overcome the problems that are faced due to traditional cathode ray oscilloscopes, we propose a solution that could use a PC as a plotting device and signal values are fed into the computer through USB as the communicating channel. The signal values that arise from the test circuit are processed using a Microcontroller and the circuit is interfaced with a PC and an application that would pick up those signal values that could be plotted onto that application similar to that of a traditional oscilloscope.

The main advantage of implementing this system is that the cost of implementation is very much low when compared to buying a traditional cathode ray oscilloscope being a software unit, supports high degree of portability when equipped in a laptop, it can be visualized as a low cost portable multi-functional oscilloscope. Since this system uses the computer monitor as the display unit, large and detailed information regarding the signal details could be projected in a colored format that could be easily read by the human eye.

By using this system, we can utilize the computer's extensive secondary memory to store signal information and to share it among different computers. This scenario is very much supported when a client and the developer are located at different locations and the developer can just send the output of the circuit in a signal store file and the client can just check whether that matches their requirement, and based upon the need of the client, the developer can be instructed to modify the circuit. In this scenario, this circuit is very much beneficial to both the developer as well as the client.

PROJECT PLAN AND SPECIFICATION

2. PROJECT PLAN AND SPECIFICATION

The PC based Oscilloscope system is developed using Microcontrollers as the heart of the Hardware circuit and Visual basic is used as a tool to acquire the signals from the circuit and plot them onto the graph at the user-interface. The signal that is fed as input to the PC that contains timing, voltages, slopes, curves, and spikes of an electronic signal are plotted on to the application and are displayed in a graphical format to the user.

2.1 PROJECT ANALYSIS

A system is something that maintains its existence and functions as a whole through the interaction of its parts. A system can be visualized as a set of interrelated components, each component characterized by properties that are selected as being relevant to the purpose.

- An Embedded System is a combination of hardware unit as well as software sub-systems that are used to achieve a single specific task.
- Embedded systems are computer systems that monitor, respond to, or control an external environment as events.
- The external Environment interacts to the system through media like sensors, actuators and other I/O interfaces.
- An Embedded system must meet timing & other constraints imposed on it by environment and by the other components that are connected directly and indirectly to it.

An embedded system is a microcontroller-based, software driven, reliable, real-time control system, autonomous, or human or network interactive, operating on diverse physical variables and in diverse environments and sold into a competitive and cost conscious market.

An embedded system cannot be visualized as a computer system that is used primarily for processing. The High-end embedded system - Generally 32 or 64 Bit Controllers can be programmed and made to interact directly with the Operating System. Some of the notable examples are Personal Digital Assistants and mobile phones. The Lower end embedded systems are generally 8 or 16 Bit Controllers used with a minimal Operating Systems and hardware layout designed for the specific purpose. A few examples are Small controllers and devices in every day life like Washing Machine, Microwave Ovens, where they are embedded in.

2.2 PROJECT PLAN:

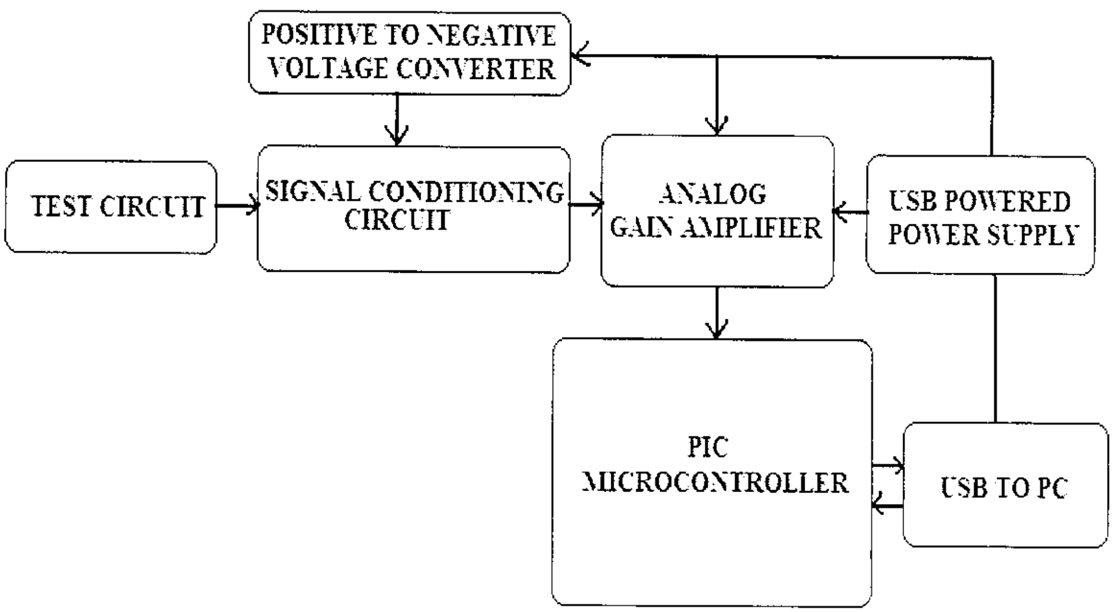


Fig 2.2.1 Hardware Functional Block Diagram

TEST CIRCUIT:

The output from a Test circuit that needs to be plotted and measured is fed in as the input to the circuit. Signals of low voltage AC and DC can be applied to test its signal frequency and voltage amplitude and plot the corresponding to the output application on the PC Monitor.

SIGNAL CONDITIONING CIRCUIT USING OPAMP:

The output that is received from the test circuit is fed into the signal conditioning circuit. The Signal Conditioning block serves as a Protective unit to the main circuit and makes uses of J-FET opamp to provide a wide bandwidth output, low input bias currents to be sent to the PC and offset currents. This voltage shifting amplifier results in high input impedance.

ANALOG GAIN AMPLIFIER:

The main function of this analog gain amplifier is to drive the analog to digital converters and to provide the required analog input to PIC Microcontroller. It provides a three – wire serial peripheral interface which helps the PIC to control the input received from the test circuit.

PIC MICROCONTROLLER:

The PIC Microcontroller that is used in the system is 18F2550. It has up to 32 KB of flash memory, 2KB of RAM, and 256 KB of EEPROM Extended Instruction set. It has USB Transceiver transfer rate of 1.5 MB/s to 12MB/s, along with multiple oscillation & power modes and an internal 31 KHz to external 48MHz clock oscillator. Along with these accompanies a 10-bit ADC Controller. This PIC microcontroller can be flashed with code based upon user requirement and to transfer input signals to USB port of PC.

2.3 HARDWARE SPECIFICATION:

The various specifications of the PC Oscilloscope system that has been designed are listed down as follows.

- Bandwidth of the system - 10 mV
- Number of Channels - 2
- Signal resolution - 10 bits
- Dynamic range - 44 dB
- Accuracy - ±5%
- Input ranges - ±100 mV to ±16 V
- Sampling rate - 10 μSecond to 100 mSecond
- Buffer memory - 10 samples to 500 samples
- Time base ranges - 10 μs/div to 999 ms/div
- Power requirement - Powered by USB port
- Operating Temperature - 20 °C to 45 °C
- Input type - Circuit Board connectors

2.3.1 COMPONENT DESCRIPTION:

LF353

LF 353 is a low cost, high speed, dual JFET input operational amplifier with an internally trimmed input offset voltage. They require low supply current yet maintain a large gain bandwidth product and fast slew rate. In addition, well matched high voltage JFET input devices provide very low input bias and offset currents.

These amplifiers are generally used in applications such as high speed integrators, fast D/A converters, sample and hold circuits and many other circuits requiring low input offset voltage, low input bias current, high input impedance, high slew rate and wide bandwidth. The devices also exhibit low noise and offset voltage drift.

Features:

- Internally trimmed offset voltage: 10 mV
- Low input bias current: 50pA
- Low input noise voltage: 25 nV/√Hz
- Wide gain bandwidth: 4 MHz
- High slew rate: 13 V/μs
- Low supply current: 3.6 mA
- High input impedance: 1012Ω

Pin Diagram:

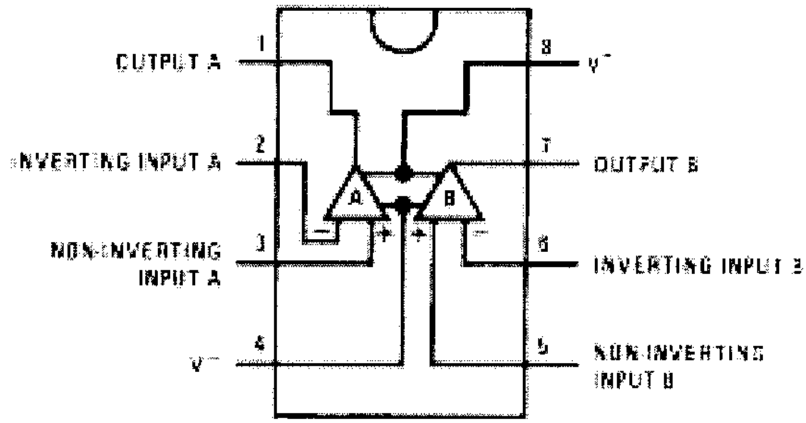


Fig 2.3.1.1 LF353 Pin Diagram

ICL7660A

The ICL7660A is a monolithic CMOS power supply circuit which offer unique performance advantages over previously available devices. The ICL7660A does the signal conversions with an input range of +1.5V to +12.0V resulting in complementary output voltage of -1.5V to -12.0V. Only 2 non critical external capacitors are needed for the charge pump and charge reservoir functions. The ICL7660A power supply can also be connected to function as voltage doublers and will generate output voltages up to +18.6V with a +10V input.

Contained on the chip are a series DC supply regulator, RC Oscillator, voltage level translator, and four output power CMOS switches. The oscillator, when unloaded, oscillates at a nominal frequency of 10 kHz for an input supply voltage of 5.0V. This frequency can be lowered by the addition of an external capacitor to the "OSC" terminal, or the oscillator may be overdriven by an external clock.

Features:

- Simple Conversion of +5V Logic Supply to ±5V Supplies
- Simple Voltage Multiplication (VOUT = (-) nVIN)
- Typical Open Circuit Voltage Conversion Efficiency 99.9%
- Typical Power Efficiency 98%
- Wide Operating Voltage Range - 1.5V to 12.0V
- ICL7660A 100% Tested at 3V
- Easy to Use - Requires Only 2 External Non-Critical Passive Components
- No External Diode over Full Temp. And Voltage Range

Pin Diagram:

MAHATMA GANDHI · CENTRAL LIBRARY · KCT CBE-6

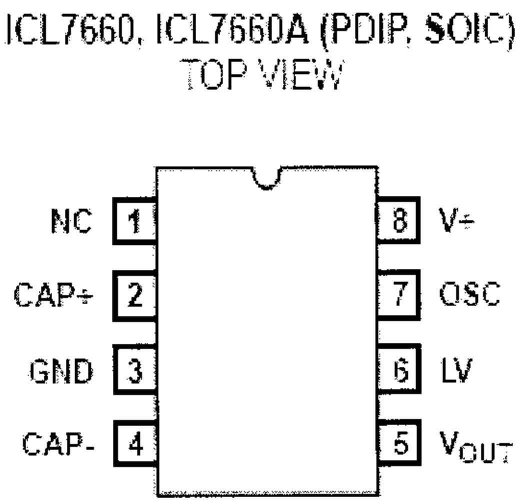


Fig 2.3.1.2 ICL7660A Pin Diagram

MCP6S91

MCP6S91/2/3 are devices that function as analog Programmable Gain Amplifiers (PGA's). They can be configured for gains from +1 V/V to +32 V/V and the input multiplexer can select one of up to two channels Through a SPI port. The serial interface can also put the PGA into shutdown to conserve power. These PGA's are optimized for high-speed, low offset voltage and single-supply operation with rail-to-rail input and output capability.

These specifications support single supply applications needing flexible performance or multiple inputs. The one-channel MCP6S91 and the two-channel MCP6S92 are available in 8-pin PDIP, SOIC and MSOP packages. The two-channel MCP6S93 is available in a 10-pin MSOP package. All parts are fully specified from -40°C to +125°C.

Features:

- Multiplexed Inputs : 1 or 2 channels
- 8 Gain Selections : +1, +2, +4, +5, +8, +10, +16 or +32 V/V
- Serial Peripheral Interface (SPI™)
- Rail-to-Rail Input and Output
- Low Gain Error: ±1% (max.)
- Offset Mismatch Between Channels : 0 µV
- High Bandwidth : 1 to 18 MHz (typ.)
- Low Noise : 10 nV/√Hz @ 10 kHz (typ.)
- Low Supply Current : 1.0 mA (typ.)
- Single Supply : 2.5V to 5.5V
- Extended Temperature Range : -40°C to +125°C

Pin Diagram:

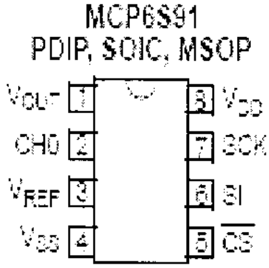


Fig 2.3.1.3 MCP6S91 Pin Diagram

2.3.2 PIC DESCRIPTION:

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology. The name PIC initially referred to "Peripheral Interface Controller".

PIC's are popular with developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.

Microcontroller Core Features:

- High-performance RISC CPU.
- USB V2.0 Compliant
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
- Enhanced Capture/Compare/PWM (ECCP) module:
- Enhanced USART module:

- Master Synchronous Serial Port (MSSP) module
- Programmable in both Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter Module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing
- Power saving SLEEP mode.
- Selectable oscillator options.
- Low-power, high-speed CMOS FLASH/EEPROM technology.
- Fully static design.
- In-Circuit Serial Programming (ICSP) .
- Single 5V In-Circuit Serial Programming capability.
- In-Circuit Debugging via two pins.
- Processor read/write access to program memory.
- Wide operating voltage range: 2.0V to 5.5V.
- Commercial and Industrial temperature ranges.
- Low-power consumption.

The PIC Microcontroller used in the project is the PIC18F2550. The features of this PIC controller are listed down.

Features of PIC18F2550:

- USB V2.0 Compliant - (1.5 Mb/s) & (12 Mb/s)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Streaming Parallel Port (SPP) for USB streaming transfers
- Four Crystal modes, including High Precision PLL for USB
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor - safe shutdown if any clock stops

Power Management Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μA typical
- Sleep mode currents down to 0.1 μA typical
- Timer1 Oscillator: 1.1 μA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA typical
- Two-Speed Oscillator Start-up
- Two External Clock modes, up to 48 MHz

Pin Diagram:

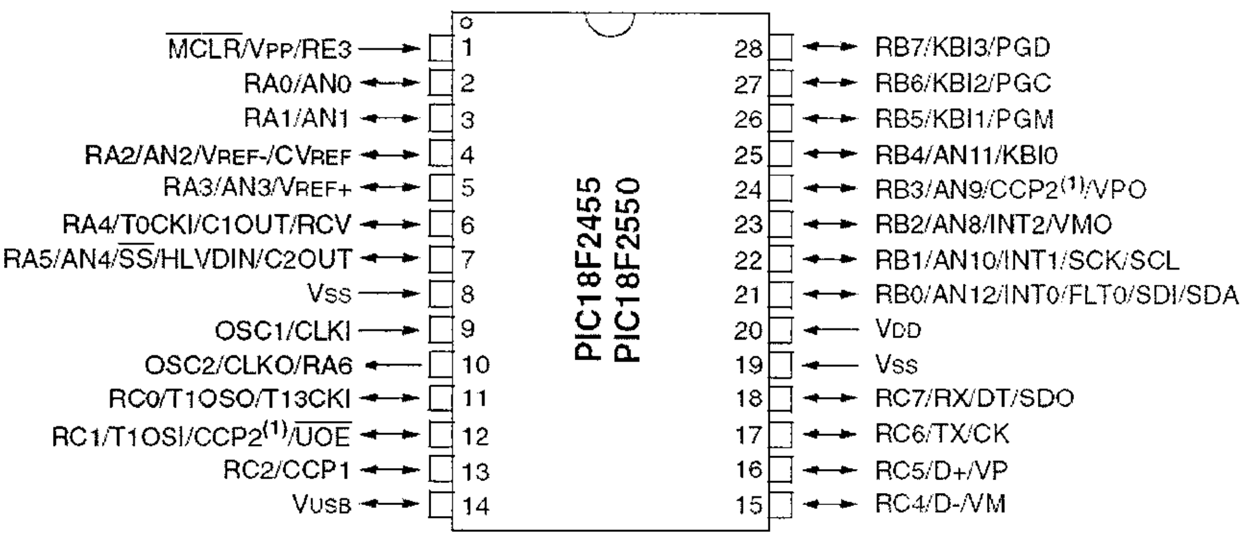
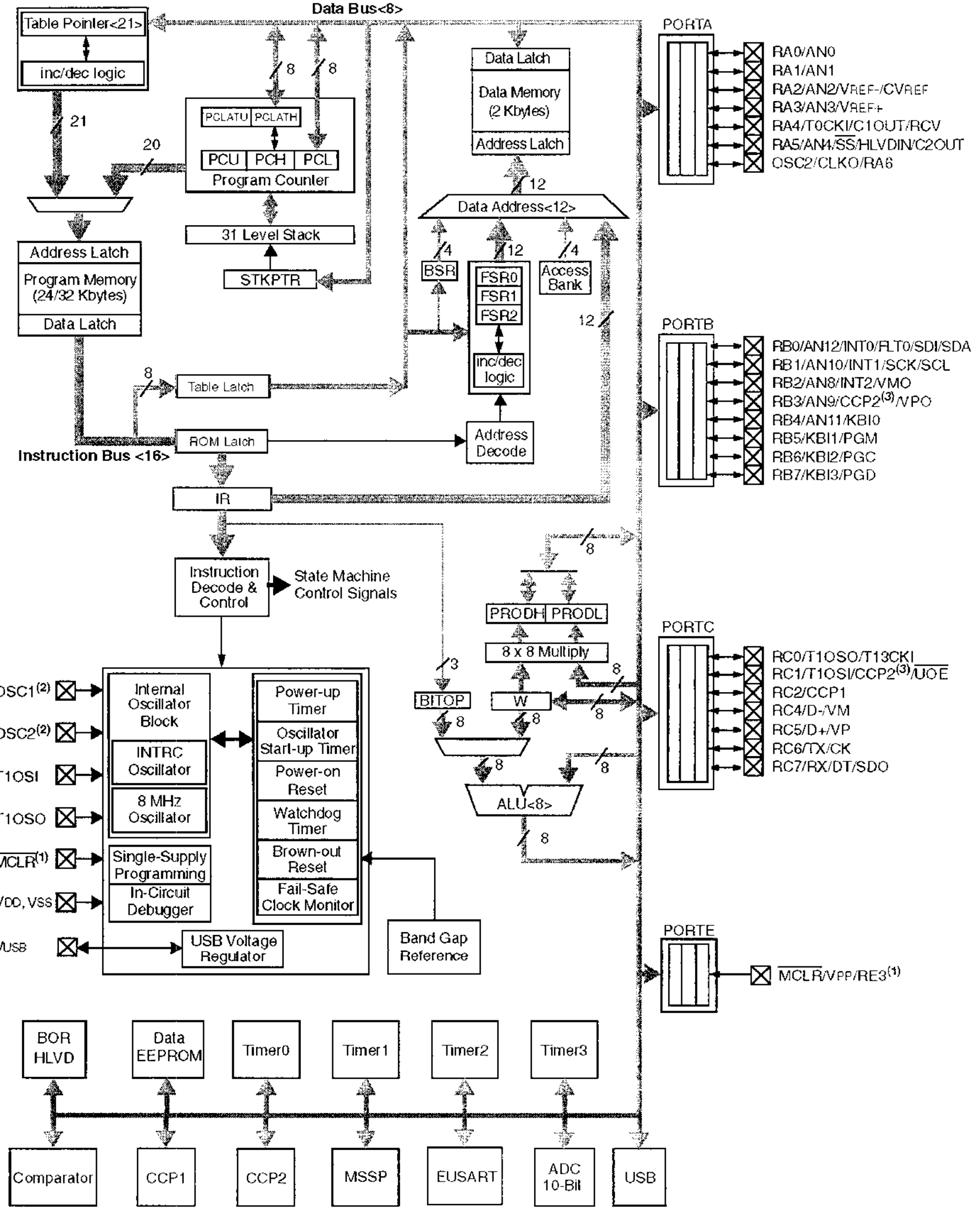


Fig 2.3.2.1 PIC18F2550 Pin Diagram

Functional Block Diagram:



I/O PINOUT DESCRIPTION:

Pin Name	Pin Number PDIP, SOIC	Pin Type	Buffer Type	Description
$\overline{\text{MCLR}}$/VPP/RE3	1			Master Clear (input) or programming voltage (input).
$\overline{\text{MCLR}}$		I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP		P		Programming voltage input.
RE3		I	ST	Digital input.
OSC1/CLKI	9			Oscillator crystal or external clock input.
OSC1		I	Analog	Oscillator crystal input or external clock source input.
CLKI		I	Analog	External clock source input. Always associated with pin function OSC1. (See OSC2/CLKO pin.)
OSC2/CLKO/RA6	10			Oscillator crystal or clock output.
OSC2		O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO		O	—	In select modes, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	TTL	General purpose I/O pin.
RA0/AN0	2			
RA0		I/O	TTL	Digital I/O.
AN0		I	Analog	Analog input 0.
RA1/AN1	3			
RA1		I/O	TTL	Digital I/O.
AN1		I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4			
RA2		I/O	TTL	Digital I/O.
AN2		I	Analog	Analog input 2.
VREF-		I	Analog	A/D reference voltage (low) input.
CVREF		O	Analog	Analog comparator reference output.
RA3/AN3/VREF+	5			
RA3		I/O	TTL	Digital I/O.
AN3		I	Analog	Analog input 3.
VREF+		I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT/RCV	6			
RA4		I/O	ST	Digital I/O.
T0CKI		I	ST	Timer0 external clock input.
C1OUT		O	—	Comparator 1 output.
RCV		I	TTL	External USB transceiver RCV input.
RA5/AN4/$\overline{\text{SS}}$/HLVDIN/C2OUT	7			
RA5		I/O	TTL	Digital I/O.
AN4		I	Analog	Analog input 4.
$\overline{\text{SS}}$		I	TTL	SPI slave select input.
HLVDIN		I	Analog	High/Low-Voltage Detect input.
C2OUT		O	—	Comparator 2 output.
RA6	—	—	—	See the OSC2/CLKO/RA6 pin.

Table 2.3.2.1 I/O PINOUT DESCRIPTION

Pin Name	Pin Number PDIP, SOIC	Pin Type	Buffer Type	Description
				PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/FLT0/SDI/SDA	21			
RB0		I/O	TTL	Digital I/O.
AN12		I	Analog	Analog input 12.
INT0		I	ST	External interrupt 0.
FLT0		I	ST	PWM Fault input (CCP1 module).
SDI		I	ST	SPI data in.
SDA		I/O	ST	I^2C™ data I/O.
RB1/AN10/INT1/SCK/SCL	22			
RB1		I/O	TTL	Digital I/O.
AN10		I	Analog	Analog input 10.
INT1		I	ST	External interrupt 1.
SCK		I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL		I/O	ST	Synchronous serial clock input/output for I^2C mode.
RB2/AN8/INT2/VMO	23			
RB2		I/O	TTL	Digital I/O.
AN8		I	Analog	Analog input 8.
INT2		I	ST	External interrupt 2.
VMO		O	—	External USB transceiver VMO output.
RB3/AN9/CCP2/VPO	24			
RB3		I/O	TTL	Digital I/O.
AN9		I	Analog	Analog input 9.
CCP2(1)		I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
VPO		O	—	External USB transceiver VPO output.
RB4/AN11/KBI0	25			
RB4		I/O	TTL	Digital I/O.
AN11		I	Analog	Analog input 11.
KBI0		I	TTL	Interrupt-on-change pin.
RB5/KBI1/PGM	26			
RB5		I/O	TTL	Digital I/O.
KBI1		I	TTL	Interrupt-on-change pin.
PGM		I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	27			
RB6		I/O	TTL	Digital I/O.
KBI2		I	TTL	Interrupt-on-change pin.
PGC		I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	28			
RB7		I/O	TTL	Digital I/O.
KBI3		I	TTL	Interrupt-on-change pin.
PGD		I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Table 2.3.2.2 I/O PINOUT DESCRIPTION (cond.)

2.3.3 MICROCONTROLLER ADVANTAGES:

A computer-on-a-chip Microcontroller is a variation of a microprocessor which combines the processor core (CPU), some memory, and I/O (input/output) lines, packed up in one single chip. The computer-on-a-chip is called the microcomputer whose proper meaning is a computer using a (number of) microprocessor(s) as its CPUs, while the concept of the microcomputer is known to be a microcontroller. A microcontroller can be viewed as a set of digital logic circuits integrated on a single silicon chip. This chip is used for only specific applications.

- A Microcontroller can Gather input from various sensors
- Signals received from sensors can be used to Process into a set of actions
- Use the output mechanisms on the Microcontroller to perform user-defined actions.
- The Microcontroller can be programmed to react based on user behavior and actions
- PIC Microcontrollers have Inbuilt ADC's to avoid usage of external ADC's
- Both RAM and ROM are packaged inside the same chip.
- Comparatively cheaper than Microprocessors.
- Multiple machines can be controlled simultaneously using one single chip.

2.4 SOFTWARE SPECIFICATION:

The PC based oscilloscope system requires the following software requirements for executing the application that displays the signal plot as connected to the signal acquisition circuit,

- Processor: Intel Pentium IV, or equivalent
- Memory: 64 MB RAM
- Operating system: Microsoft Windows XP SP2 (32-bit)
- I/O Ports: USB 1.1 compliant port
- Microsoft .Net framework 2.0 installed
- USB Drivers for the Microcontroller 18F2550

The following softwares were used in the development and testing process of the system:

- Microsoft Visual Basic 6.0
- MPLAB IDE v8.20
- HITEC C Compiler for PIC 18 Series Microcontroller
- WINPIC C Compiler

Various softwares used to develop the project are being discussed in this section.

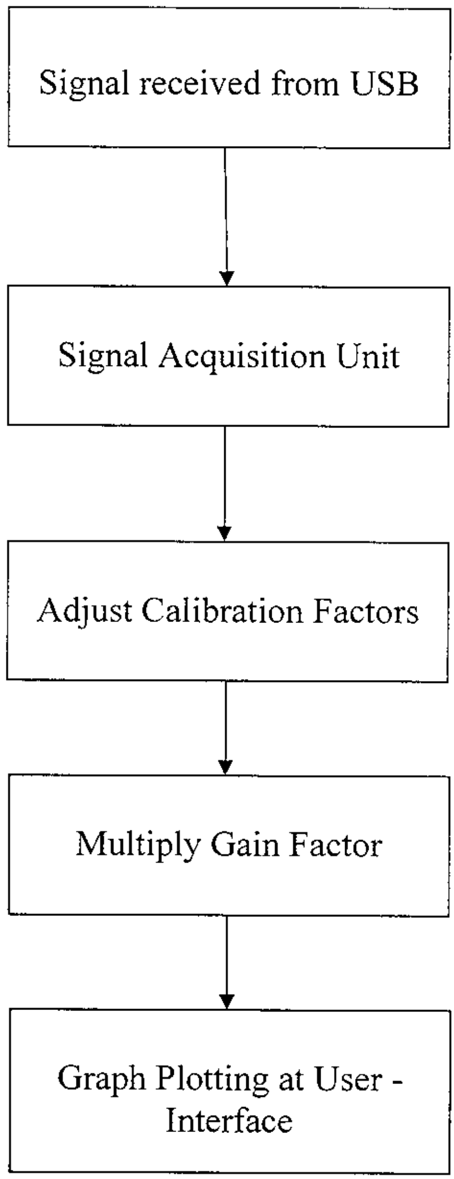


Fig 2.4.1 Signal Processing Software Unit

2.4.1 MPLAB UTILITY DESCRIPTION:

MPLAB IDE is a software program that runs on a PC to develop applications for Microchip microcontrollers. It is called an Integrated Development Environment, or IDE, because it provides a single integrated environment to develop code for embedded microcontrollers.

MPLAB Integrated Development Environment (IDE) is a comprehensive editor, project manager and design desktop for application development of embedded designs using Microchip PICmicro MCUs and dsPIC DSCs.

Two major features of MPLAB IDE are projects and workspaces. A project contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options. A workspace contains information on the selected device, debug tool and/or programmer, open windows and their location and other IDE configuration settings. The best way to set up your project and its associated workspace is by using the Project Wizard. This will set up one project in one workspace.

MPLAB IDE runs as a 32-bit application on MS Windows, is easy to use and includes a host of free software components for fast application development and super-charged debugging. MPLAB IDE also serves as a single, unified graphical user interface for additional Microchip and third party software and hardware development tools.

2.4.2 HITECH C COMPILER DESCRIPTION:

HI-TECH C PRO for the PIC18 MCU Family supports a number of special features and extensions to the C language which are designed to ease the task of producing ROM-based applications. HI-TECH Software makes industrial-strength software development tools and C compilers that help software developers write compact, efficient embedded processor code.

For over two decades HI-TECH Software has delivered the industry's most reliable embedded software development tools and compilers for writing efficient and compact code to run on the most popular embedded processors. HI-TECH's reliable development tools and C compilers, combined with world-class support have helped serious embedded software programmers to create hundreds of breakthrough new solutions.

HI-TECH PICC is a high-performance C compiler for the Microchip PIC micro 10/12/14/16/18 series of microcontrollers. HI-TECH PICC is an industrial-strength ANSI C compiler - not a subset implementation like some other PIC compilers. The PICC compiler implements full ISO/ANSI C, with the exception of recursion.

PICC can be run entirely from the Integrated Development Environment. This environment allows you to manage all of your PIC projects. You can compile, assemble and link your embedded application with a single step. After compilation of the code written in C Language, HITEC C Compiler converts into assembly language code and that assembly language code is flashed into the PIC Microcontroller.

The compiler reads the program in one language, the source language and translates into an equivalent program in another language, the target language. The translation process should also report the presence of errors in the source program.

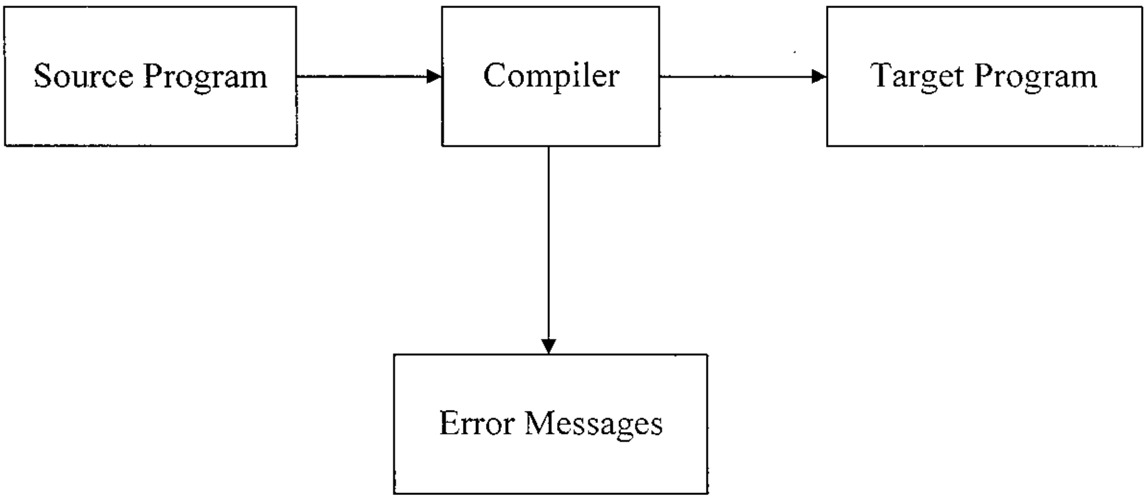
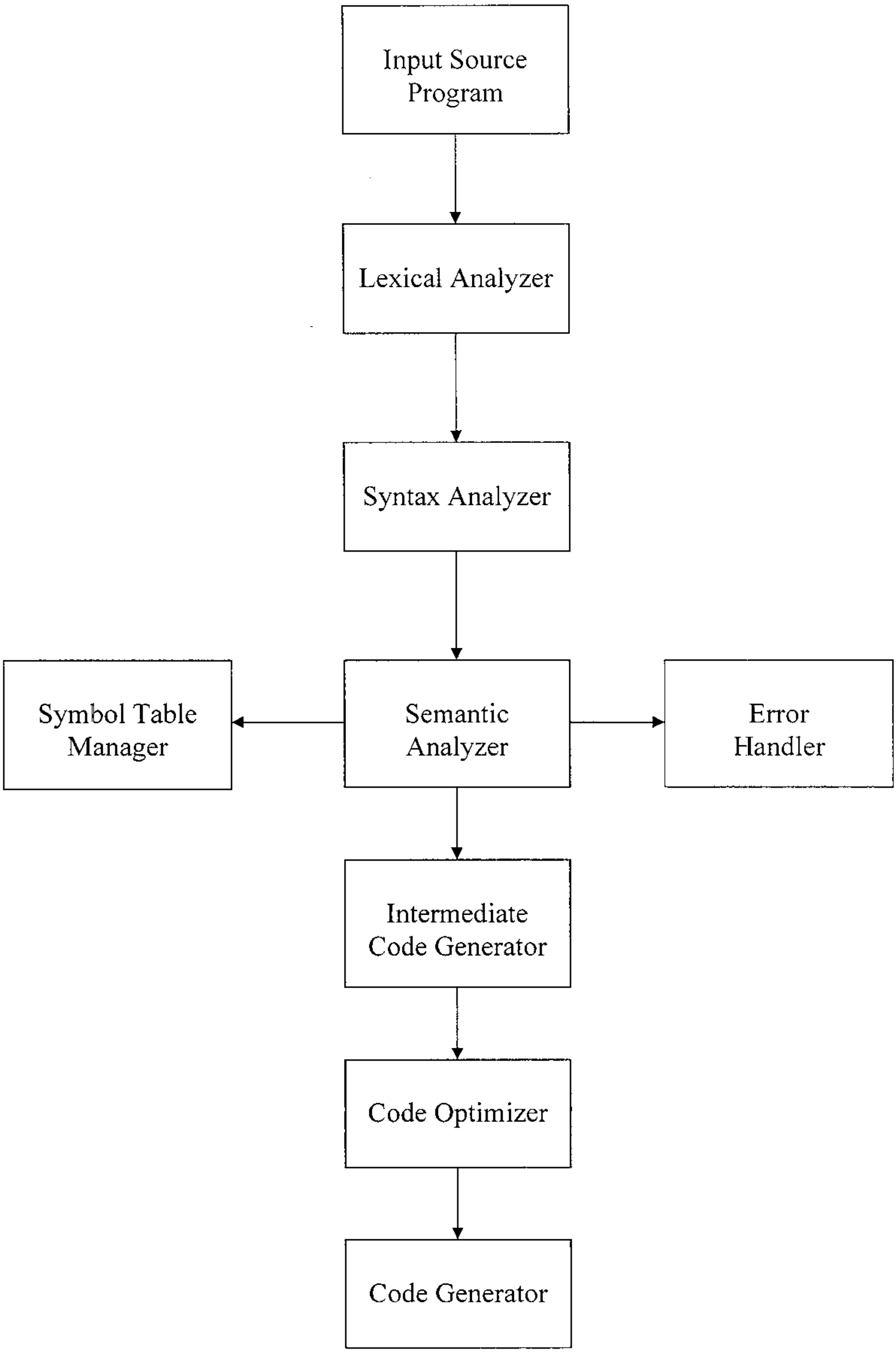


Fig 2.4.2.1 Application Process Sequence

There are two parts of compilation. The analysis part breaks up the source program into constant piece and creates an intermediate representation of the source program. The synthesis part constructs the desired target program from the intermediate representation.

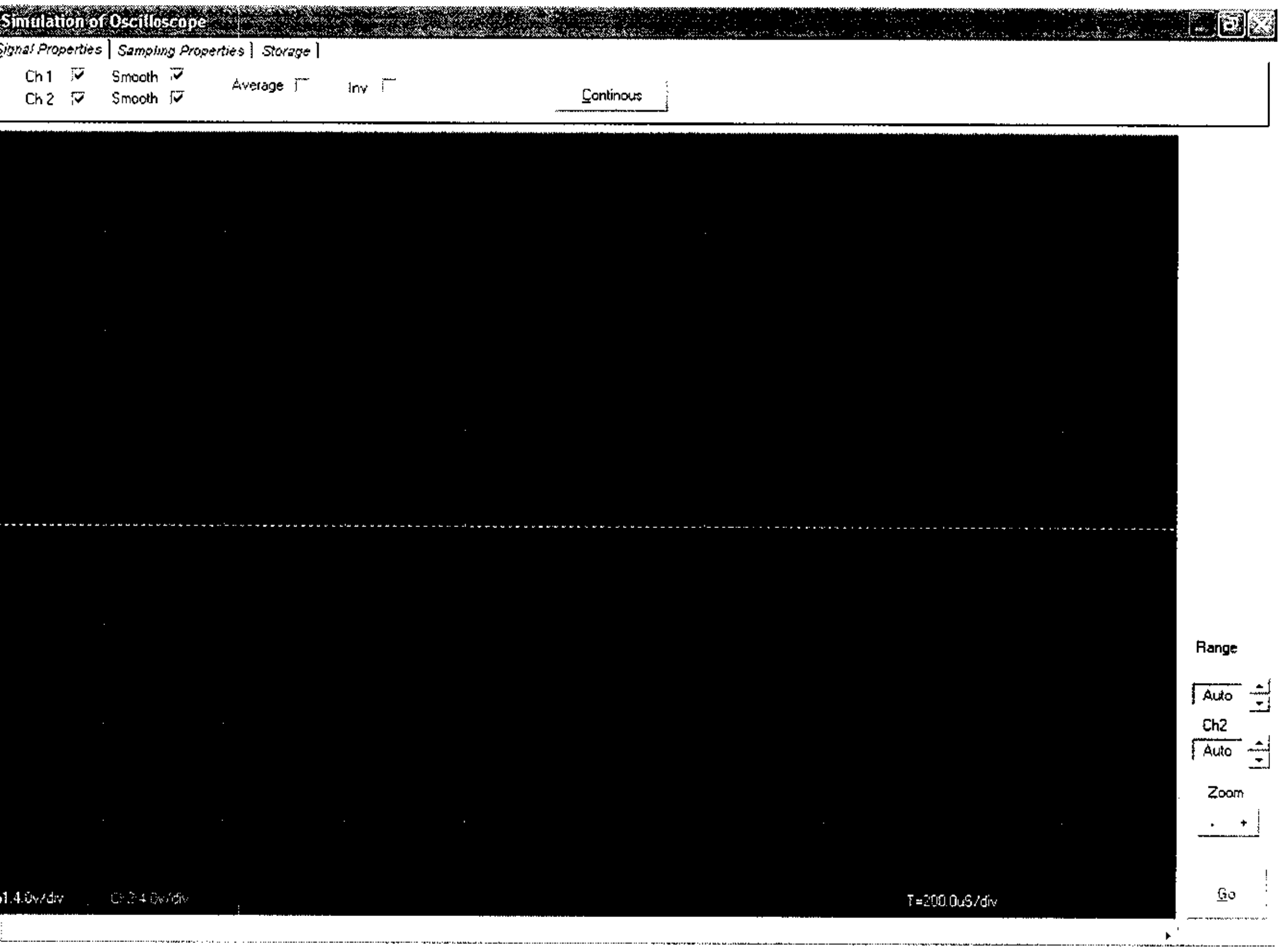
Semantic analyzer takes the output of syntax analyzer and produces another tree. Similarly, intermediate code generator takes a tree as an input produced by semantic analyzer and produces intermediate code

The compiler has a number of phases plus symbol table manager and an error handler illustrated as follows.


Input Source Program
Lexical Analyzer
Syntax Analyzer
Symbol Table Manager
Semantic Analyzer
Error Handler
Intermediate Code Generator
Code Optimizer
Code Generator

2.4.3 USER-INTERFACE DESCRIPTION:

The User-Interface used to plot the graph is Visual Basic 6.0. Visual Basic (VB) is an event driven programming language and associated development environment from Microsoft for its COM programming model. VB has been replaced by Visual Basic .NET. The older version of VB was derived heavily from BASIC and enables the rapid application development (RAD) of graphical user interface (GUI) applications, access to databases using Data Access Objects, Relational Data Objects, or Active-x Data Objects, and creation of ActiveX controls and objects.



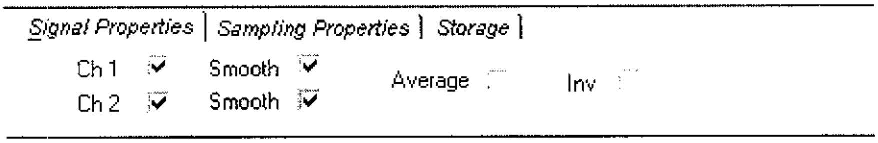


Fig 2.4.2.4 Signal properties Screenshot

The various parameters that could be set to display both the channels or only a single channel could be made possible using the signal properties tab. Using this tab, various properties such as smoothening of the signal, displaying the average value of the channel signals, inversing a particular channel can be made possible. Smoothening provides a smoothened signal when the signal received is much of distorted form.

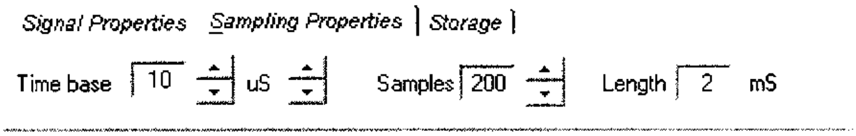
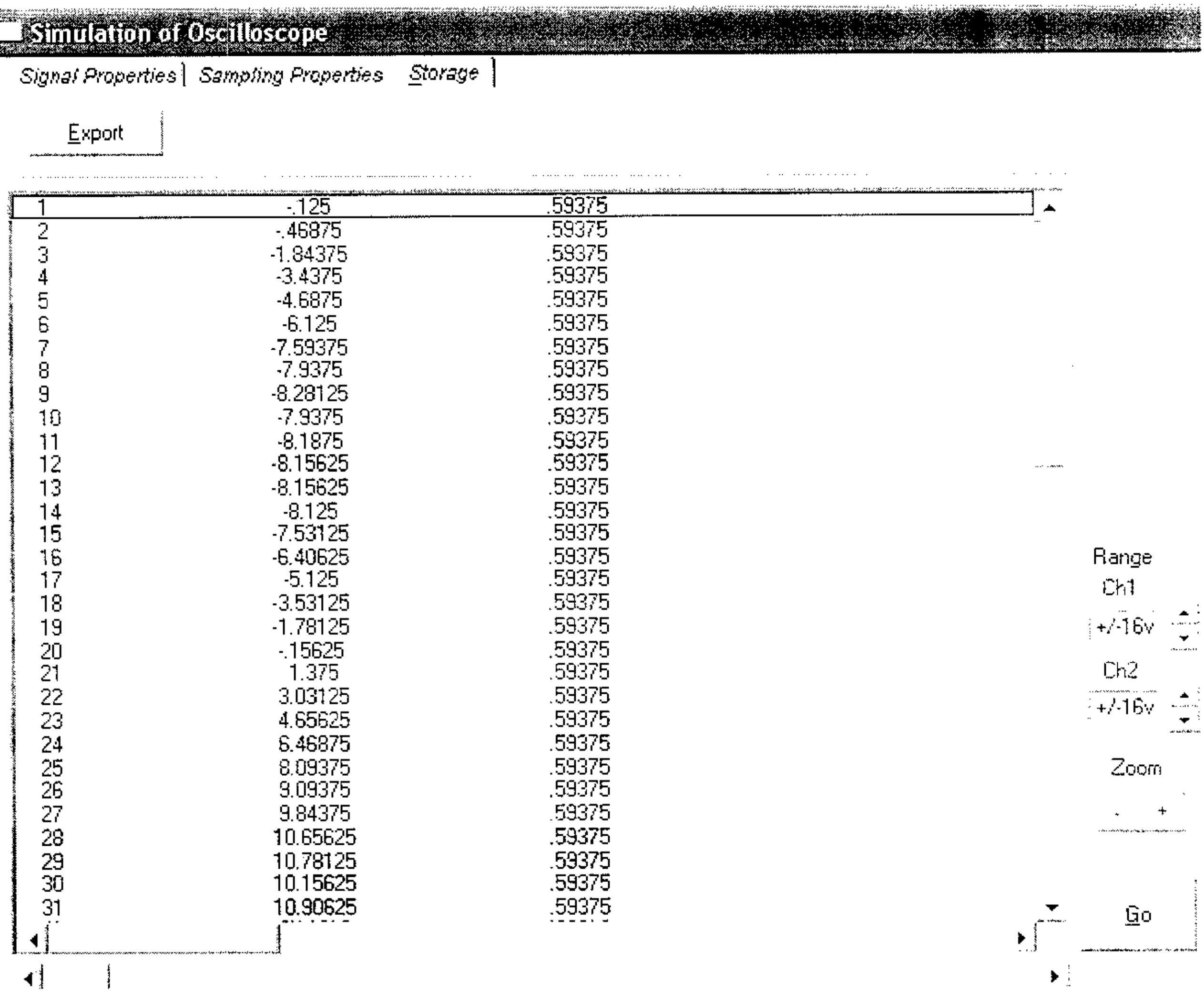


Fig 2.4.2.5 Sampling Properties Screenshot

Sampling properties allows altering the time domain of receiving a signal and the particular number of signals per time value. The time domain can be altered between micro second and milli second. The number of samples could be increased to accommodate more number of samples in the user interface at a same screenshot, but it will take more time to acquire signal if the samples that need to be received are more.



1	-.125	.59375
2	-.46875	.59375
3	-1.84375	.59375
4	-3.4375	.59375
5	-4.6875	.59375
6	-6.125	.59375
7	-7.59375	.59375
8	-7.9375	.59375
9	-8.28125	.59375
10	-7.9375	.59375
11	-8.1875	.59375
12	-8.15625	.59375
13	-8.15625	.59375
14	-8.125	.59375
15	-7.53125	.59375
16	-6.40625	.59375
17	-5.125	.59375
18	-3.53125	.59375
19	-1.78125	.59375
20	-.15625	.59375
21	1.375	.59375
22	3.03125	.59375
23	4.65625	.59375
24	6.46875	.59375
25	8.09375	.59375
26	9.09375	.59375
27	9.84375	.59375
28	10.65625	.59375
29	10.78125	.59375
30	10.15625	.59375
31	10.90625	.59375

Fig 2.4.2.6 Export signal Screenshot

The User interface also provides an option to store the signals as numeric values as they are received by the acquisition module. The various signal values received are actually stored in a byte array and these values could be exported into a text value and could be shared with people. This feature allows the oscilloscope to act as a Storage Oscilloscope enhancing its functionality.

A programmer can put together an application using the components provided with Visual Basic itself. Programs written in Visual Basic can also use the Windows Application Programming Interface, but doing so requires external function declarations.

Advantages of Visual Basic:

- VB is not only a programming language but primarily an integrated, interactive development environment ("IDE").
- The VB-IDE has been highly optimized to support rapid application development ("RAD"). It is particularly easy to develop graphical user interfaces and to connect them to handler functions provided by the application.
- The graphical user interface of the VB-IDE provides intuitively appealing views for the management of the program structure in the large and the various types of entities (classes, modules, procedures, forms ...).
- VB provides a comprehensive interactive and context-sensitive online help system.
- VB is a component integration language which is attuned to Microsoft's Component Object Model ("COM").
- Interfaces of COM components can be easily called remotely via Distributed COM ("DCOM"), which makes it easy to construct distributed applications.
- COM components can be embedded in / linked to your application's user interface and also in/to stored documents (Object Linking and Embedding "OLE", "Compound Documents").
- There is a wealth of readily available COM components for many different purposes.

SYSTEM FUNCTIONALITY

3. SYSTEM FUNCTIONALITY

3.1 HARDWARE FUNCTIONALITY:

3.1.1 FLASHING MICROCONTROLLER CODE:

1. Write the program in MPLAB IDE using various tools available in IDE.
2. Save the file as *.c. and compile it.
3. After successful compilation of the coding use the .ASM File generated after compilation to flash the Microcontroller.
4. Close the MPLAB IDE.
5. Fix the Controller IC into PIC flashing kit.
6. Using Micro controller PIC Flash Software open the code generated by MPLAB.
7. Before the fusing process, the software will confirm whether the Microcontroller is empty or not and a warning that existing code will be erased.
8. Accept the dialog and proceed to fusing the Microcontroller.
9. A Fuse Settings Dialog Box is displayed which can be used to configure settings for fusing the controller.
10. In the dialog box displayed, Select WDT as Disabled, WRT as Enabled and Oscillator as XT then click on OK.
11. After successful fusing, the application will show the status of the

12. After the status is successful, remove the Controller from the kit and the controller is ready for use based upon the code used to fuse.

13. This completes the PIC fusing process and flashing is complete.

3.1.2 CIRCUIT OPERATION:

The circuit of PC Based Oscilloscope is capable of operating up to 10 KHz with (±16V) input voltage. The PIC 18F2550 Microcontroller chip is powered from the USB power supply acquired from USB port of PC. These controller chips along with MCP6591, LF353 & L7660 IC's are used in the circuit. The project can be run in a PC to display the input status as well as the signal value.

At the heart of this oscilloscope is USB – to – Microcontroller PIC18F2550 transceiver & a CPU running up to 12MBPs. LF353 amplifier is a J-FET amplifier. It is used to convert an input signal range (±16V) to (0-5V) range. It requires negative supply of (-5V) to work. For this purpose, we use a L7660 IC which converts (+5V) supply to (-5V) supply which is actually a monolithic CMOS switched capacitors voltage converter.

MCPS91 amplifier is designed with CMOS input devices. It is a Serial Peripheral Interface allows the PIC to control then through its pins 5, 6, 7. V_{ref} (pin 3) which is an analog input should be at a voltage between V_{ss} & V_{dd}. The voltage at this pin shifts the output voltage. The Serial Peripheral Interface input are chip select (CS), serial input (SI) and serial clock (SCK). These are Schmitt triggered, CMOS logic input.

The USB cable from PC has 4 pin namely VCC, D-, D+ and Ground respectively. It provides DC output of 5V 100mA power supply voltage. This is used to drive the PIC along with OpAmp IC. All the data is transmitter on the D+/D- symmetrical pins using a variable bit rate. The PIC output sends information to PC through USB by the way of different descriptors. Each such descriptor contains a specific kind of information about the device (vender ID, serial number, format and type of data transmitted).

3.2 SOFTWARE FUNCTIONALITY:

The User-Interface used in the system is developed using Visual Basic 6.0 as discussed before. The functions required to read and write from the Microcontroller are provided by the Dynamic Link Library, mpusbapi_vb.dll. Provided along with package, mpusbapi_vb is a dynamic - link library (DLL) that provides a set of public functions for communicating with the Board with the custom driver. Both Visual Basic 6.0 and mpusbapi_vb provide mechanisms to access the Win32 Dynamic Link Library directly. The various sub-systems of the user-interface are discussed as follows.

3.2.1 ACQUIRE VALUES FROM THE CONTROLLER:

The various modules that are used to communicate with the PIC Microcontroller are MPUSBOpen, MPUSBWrite, MPUSBRead and MPUSBClose. The MPUSBOpen module is used to communicate the PC with the Microcontroller using USB as the communicating interface. Along with the function, various parameters such as Instance ID along with the unique VIDPID of the board are programmed and interfaced

The MPUSBWrite does the function of writing data into the USB port back to the microcontroller. This function is useful when the parameters such as calibration factors, channel gain amplitude e.t.c. are to be programmed into the PIC, and the PIC will store it onto its RAM Memory and based upon the specified parameters, the PIC Microcontroller will send data back to the PC. The inverse of this function is done by the MPUSBRead module, which actually helps the application to read values received from the microcontroller and store it in a byte array and these values are used to plot the signal values that are received from the test circuit onto the graph in the user-interface.

After the signals have been received from the Microcontroller and the application needs to be exited, we must close the connection using MPUSBClose sub-function. This ensures proper usage of the system and avoids any data loss of the flash memory in the PIC Microcontroller. This function would return a Boolean value specifying the status of closing the function.

Using these MPUSB Functions, we can communicate with the USB PIC Microcontroller via PC. After the signal values have been received from the Microcontroller, the values need to be multiplied using the gain coefficients that are set during the system calibration and divided by the calibration factors as set by the application developer and the person who uses the application and then the signal transforms into a one that could be plotted onto the application. After signal transformation, the signal values are stored into an array and sent to the graph plotting module.

3.2.2 PLOTTING THE GRAPH:

The first step in plotting the graph is to check whether the signal received from the acquiring function is valid and is capable of being accommodated into the graph of the application, if not an error must be posted saying to change the signal parameters. The graph must be initialized with probably a dull color so that the signal is much more visible as the foreground. Various other parameters such as the graph size are initialized based on the screen resolution received from the user-interface application. The range parameters of the signal are also initialized before plotting the graph from the signal information. The same is also written back to the screen of the user-interface and displayed.

After displaying the signal acquisition information, the other parameters such as calibration and time base are also provided to the user-interface. After initialization, the graph is plotted using the signal value as a multiplier, and the function pset is used to plot the signal value point onto the graph, using a defined fore-color.

During the process of plotting, both the channels of the Microcontroller are checked for valid status and then the graph is plotted based upon the provided parameters.

ADVANTAGES AND LIMITATIONS

4. ADVANTAGES AND LIMITATIONS:

The system "PC BASED OSCILLOSCOPE" that is developed enjoys many benefits due to its connectivity to a PC. The various advantages and limitations are discussed briefly as follows.

4.1 ADVANTAGES:

PC Based Digital Oscilloscope provides many advantages over a traditional oscilloscope. Many of the advantages are discussed as follows.

- By integrating several test instruments into one small unit, PC Oscilloscopes are lighter and more portable units than traditional test equipments. When used with a laptop computer, it is possible to carry a complete electronics lab in the same bag as your PC.

- The display of a traditional oscilloscope is limited by the physical size of the oscilloscope, and may only be a single color. With a PC Oscilloscope, the computer controls the display, so not only a full color display is obtained, but the display can be the size of the computer monitor or display unit.

- PC Oscilloscopes store the signals that are measuring directly on the PC. With the power of today's modern PCs this gives vast storage capabilities. Along with allowing the application to record lengthy signals this also lets you save signals for reviewing at a later date.

- If the signal that has been captured needs to be shared, we can save the waveform and share it as a text document or a picture showing

- PC Oscilloscopes are external devices that are connected to your PC using the Universal Serial Bus (USB). Virtually every laptop or desktop PC has multiple USB ports so using the PC Oscilloscope with either a desktop or a laptop PC is made simple.

- A PC Oscilloscope packages a signal analyzer and data logger. So with PC Oscilloscopes a complete test and measurement lab can be obtained in one cost-effective unit.

4.2 LIMITATIONS:

- The system that is designed to acquire signals from a test circuit cannot receive high frequency of signals in terms of GHz; hence this system cannot be employed in Industrial Applications.

- If the system has a dual channel input, we cannot acquire both AC and DC signals at the same time, hence it is not possible to analyze different set of signals of AC and DC at the same time.

- The hardware unit is very sensitive unit and hence high amplitude signals would eventually damage the protective circuit and would have a possibility of damaging other parts of the circuit.

- High ranging of input signals could disrupt the interfacing between the PC and the project board which in case needs to be re-connected and the drivers re-initialized which consumes much time and effort.

PRODUCT TESTING

5. PRODUCT TESTING:

The system "PC BASED OSCILLOSCOPE" that is developed is tested with various test circuits that produce outputs of Sine wave, DC wave, square wave using 555 timers and a DC Rectifier circuit. The various steps on how the testing was performed are listed as follows.

- The circuit is connected to the PC using the USB cable on both the sides and if it is the first time, the drivers need to be specified to ensure proper interfacing between the PC and the PIC.

- The PC Oscilloscope application is opened and various calibration factors are set and the signal value parameters are set based on the input signal.

- During calibration, make sure to remove the channel input from the test circuit, and keep them isolated from connecting them to circuits.

- After calibration is done, the acquisition module is selected so that the Application receives the signals that are sent by the Micro-controller.

- The signal values received by the PIC are initially stored in an Array of bytes and then sent to the plotting function to plot them into the graph as per the signal values.

- The signal values are plotted onto the user-interface based upon the various initialization parameters, and then displayed to the user.

- The user is also specified an option whether to save the signal array to an external text file enabling the user can share the file.

- If a different signal is given as input, the user can change the various parameters available, and can display the signal from the test circuit.

- If any discrepancy occurs in between the user can also calibrate the system back to zero to enable flawless operation.

- When the application is closed, the MPUSBCLOSE module is initiated which closes the connection between the PC and the Microcontroller.

- The testing process is carried out for different types of signals and the output is verified.

- This completes the end of testing phase.

Various inputs are specified to the system and their corresponding graphs are plotted as follows,

5.1 TEST OUTPUTS:

Case 1: (Alternating Current Input)

Input provided:

11V AC Peak-to-Peak

Tested output:

12.5 V AC Peak to Peak

(Tested using Calibrated Multimeter)

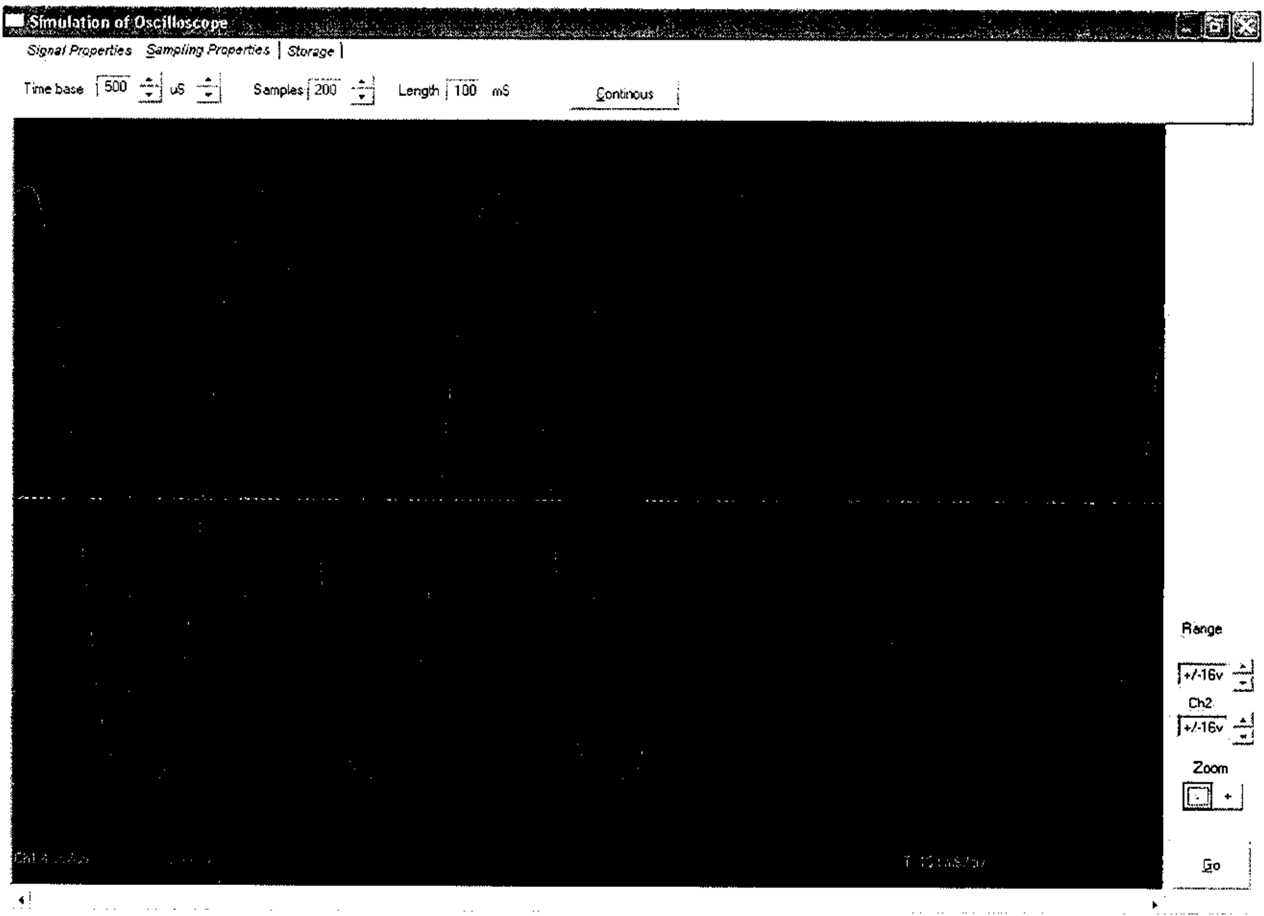


Fig 5.1.1 AC Input Screenshot

Case2: (Direct Current Input)

Input provided:

4 V DC

Tested output:

4.5 V DC

(Tested using Calibrated Multimeter)

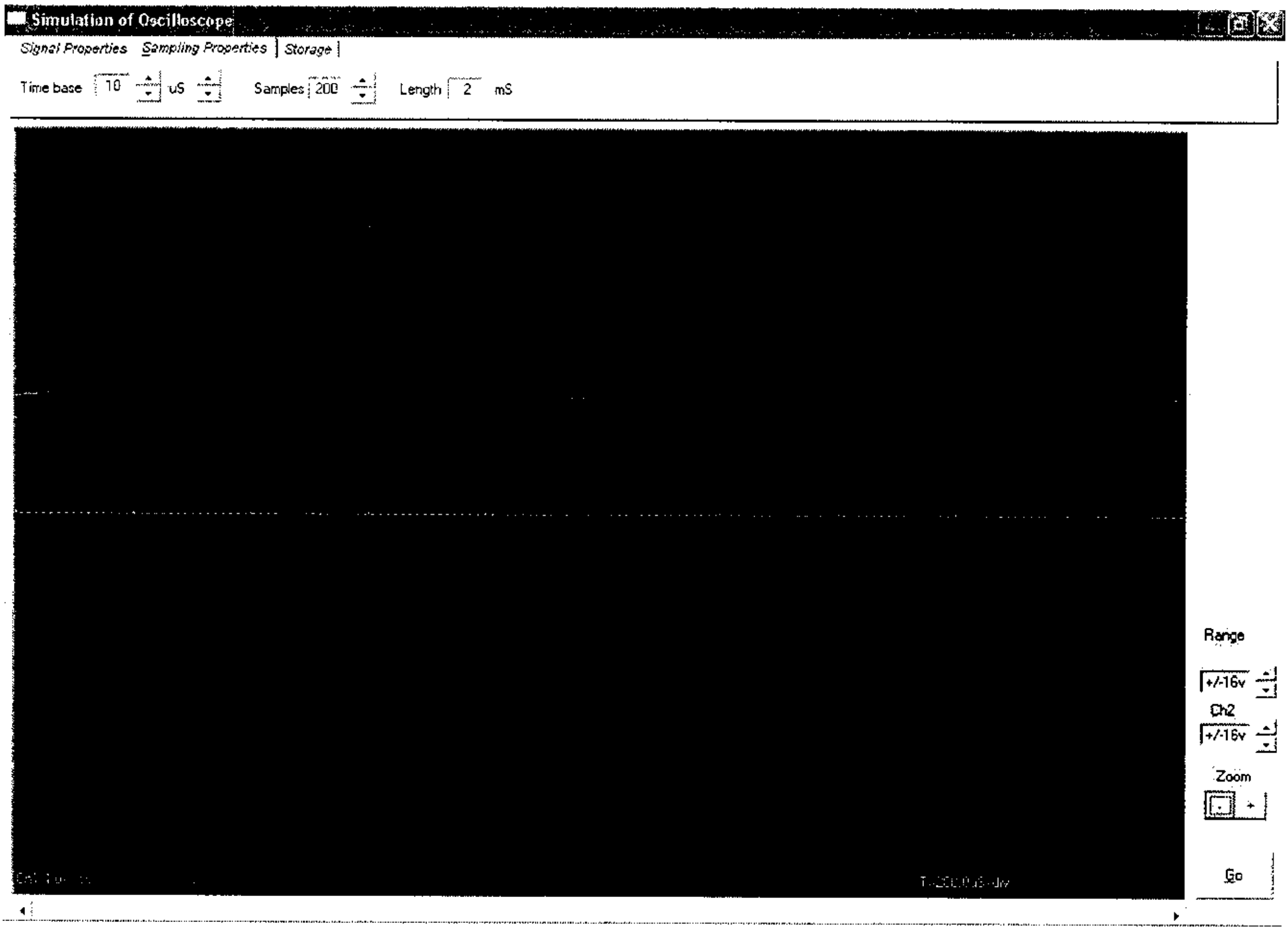


Fig 5.1.2 DC Input Screenshot

Case3: (Square wave Input)

Input provided: Square wave using 555 Timer at Astable mode

T_{on}: 2V T_{off}: 4 V

Tested output:

T_{on}: 1.75 V T_{off}: 3.75 V

(Tested using Calibrated Multimeter)

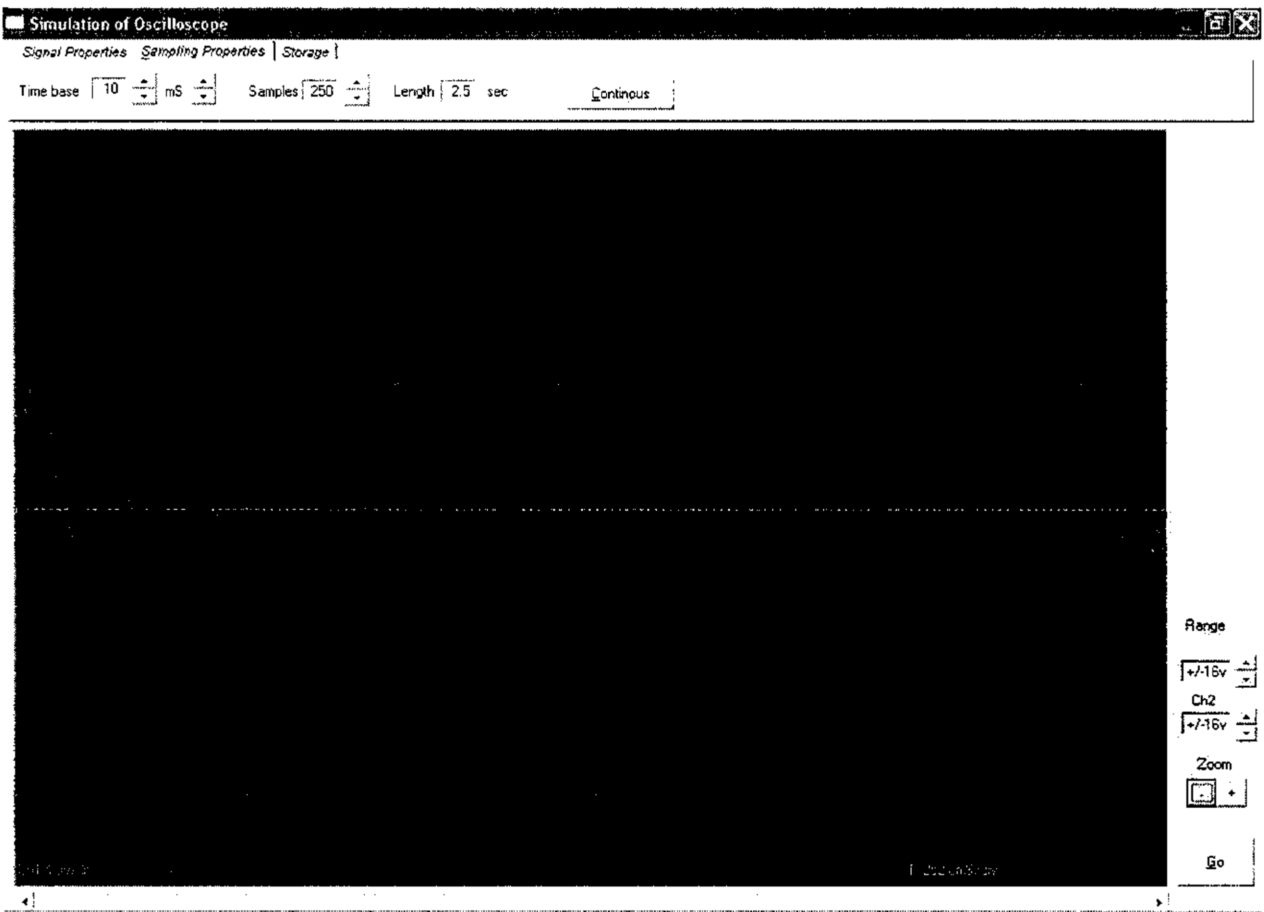


Fig 5.1.3 Square Wave Input Screenshot

FUTURE ENHANCEMENTS

6. FUTURE ENHANCEMENTS:

Various future enhancements that could be made possible to the system are listed down.

- The circuit's acquisition is capable of interfacing only 2 channels simultaneously. This could be further increased to 4 or 8 channels.

- Further signal noise that is caused due to circuit can be reduced by using external filter mechanisms.

- The frequency range of input signals to the acquisition board can be further improvised so that the circuit is capable of receiving signals of high frequency in terms of Giga-Hertz

- The signal values that could be exported using the export option in the user-interface could be made to re-plot back into the application as if the test circuit is directly connected to the system.

- The USB-PC interface could be made un-interrupted while changing the test circuit.

- Both AC and DC can be made to plot simultaneously without any errors in signal plotting.

APPENDIX

7. APPENDIX:

7.1 MICROCONTROLLER CODE:

```
#include "p18f2550.h"

#include "typedefs.h"
#include "usb.h"
#include "io_cfg.h"          // I/O pin mapping
#include "user.h"


DATA_PACKET databuff;



int send1_pts, send2_pts, s_shift;
int send1H_pts, send2H_pts, sendoffs;


unsigned char acqcyc, timeout;
unsigned char ordre;
unsigned char vH1, vH2, nptsH, nptsL;
unsigned char testH, testL;
unsigned char tt1, tt2, tt3;
unsigned char savFSR1L, savFSR1H,savFSR2L, savFSR2H;





/** P R I V A T E   P R O T O T Y P E S
***************************************/
void CopyData(unsigned int addr);
unsigned char RdEEPROM(unsigned char ad);
void WrEEPROM(unsigned char ad, unsigned char dat);

/** D E C L A R A T I O N S
************************************************/

/************************************************************
*****************
```

```
*

****************************************************************
***************/
void UserInit(void) {


 ADCON0 = 0x01;              adc on
 ADCON1 = 0b00001101;
 ADCON2 = 0b10001100;

 PORTA = 0b11110011;
 TRISA = 0b11100011;
 TRISB = 0xff;
 PORTC = 0b11111100;
 TRISC = 0b11111000;

 send1_pts = send2_pts = 0;
 send1H_pts = send2H_pts = 0;
 sendoffs=0;
}

/***************************************************************
*****************
 * Function:

****************************************************************
***************/
unsigned char RdEEPROM(unsigned char ad) {

        _asm
         bcf   EECON1,7,0
         bcf   EECON1,6,0
         bsf   EECON1,0,0
        _endasm
        return(EEDATA);
        }

void WrEEPROM(unsigned char ad, unsigned char dat) {
         EEADR=ad;
         EEDATA=dat;
          asm
```

```
        bcf    EECON1,6,0
        bsf    EECON1,2,0

        movlw       0x55
        movwf       EECON2,0
        movlw       0xAA
        movwf       EECON2,0

        bsf    EECON1,1,0
       _endasm

   while (EECON1bits.WR!=0) { ; }

       _asm
        bcf    EECON1,2,0
       _endasm
       }


void SetGain0(unsigned char gain)
 {
unsigned char nn, q;

 CS_CH0=0;

 nn=0b01000000;
 for (q=0;q<8;q++) {
  if (nn&128) SI_CH0=1; else SI_CH0=0;
  SCK_CH0=1;
  nn<<=1;
  SCK_CH0=0;
 }

 nn=gain;
 for (q=0;q<8;q++) {
  if (nn&128) SI_CH0=1; else SI_CH0=0;
  SCK_CH0=1;
  nn<<=1;
  SCK_CH0=0;
 }
 CS_CH0=1;
}
```

```
void SetGain1(unsigned char gain)
{
unsigned char nn, q;

 CS_CH1=0;

 nn=0b01000000;
 for (q=0;q<8;q++) {
  if (nn&128) SI_CH1=1; else SI_CH1=0;
  SCK_CH1=1;
  nn<<=1;
  SCK_CH1=0;
 }

 nn=gain;
 for (q=0;q<8;q++) {
  if (nn&128) SI_CH1=1; else SI_CH1=0;
  SCK_CH1=1;
  nn<<=1;
  SCK_CH1=0;
 }
 CS_CH1=1;
}
void CopyData(unsigned int addr) {
        _asm
                decf   FSR1L,1,0
                decf   FSR1L,1,0
                movf   INDF1,0,0
                movwf  FSR0H,0
                decf   FSR1L,1,0
                movf   INDF1,0,0
                movwf  FSR0L,0


                movf   FSR2H,0,0
                movwf  vH2,1
                movf   FSR2L,0,0
                movwf  tt2,1

                movlw  databuff
                movwf  FSR2L,0
```

```
                movwf       FSR2H,0
                movlw       64
                movwf       vH1,1


        movf POSTINC0,0,0
                movwf       POSTINC2,0
                decfszvH1,1,1


                incf    FSR1L,1,0
                incf    FSR1L,1,0
                incf    FSR1L,1,0

                movf vH2,0,1
                movwf       FSR2H,0
                movf tt2,0,1
                movwf       FSR2L,0
        _endasm
}


unsigned int doADC(unsigned char voie)
{
    if ADCON0=5; else ADCON0=1;
    _asm

        bsf         ADCON0,1,0
wait:  btfsc  ADCON0,1,0


    _endasm
    return(ADRES);
    }

void doADC0(void)
 {
    ADCON0=5;
    _asm
        bsf         ADCON0,1,0
waitv0:     btfsc  ADCON0,1,0
        bra         waitv0
    _endasm
```

```
void doADC1(void) {
        ADCON0=1;
        _asm

                bsf             ADCON0,1,0
waitv1:         btfsc   ADCON0,1,0

        _endasm
        }

void ProcessIO(void) {
unsigned char ordrelen, pt;
unsigned int level;
unsigned long int sum32;

  // User Application USB tasks
  if((usb_device_state                                          <
CONFIGURED_STATE)||(UCONbits.SUSPND==1)) return;

  ordrelen=(USBGenRead((byte*)&databuff,16));



  if (ordre==0x80)
  {

        pt=0xff;
        WrEEPROM(2,pt);
        WrEEPROM(4,pt);

        sendoffs=64;

  }


  if (ordre==0x81) {
        pt=databuff._byte[1];
        WrEEPROM(9,pt);
        //Voie 2
        pt=databuff._byte[2];
        WrEEPROM(10,pt);
```

```
  ordre=0;
 }

if (ordre==0x83)
 {
   SetGain0(0);
  SetGain1(0);

  sum32=0;

  for (level=0;level<0x100;level++) {
   sum32+=doADC(0);
   for (tt1=0;tt1<64;tt1++) { ; }
   }

  pt=sum32>>8;
       WrEEPROM(1,pt);
       pt=sum32>>16;
       WrEEPROM(2,pt);


       sum32=0;
  for (level=0;level<0x100;level++) {
   sum32+=doADC(1);
   for (tt1=0;tt1<64;tt1++) { ; }
   }

  pt=sum32>>8;
       WrEEPROM(3,pt);
       pt=sum32>>16;
       WrEEPROM(4,pt);


  sum32=0;
  for (level=0;level<0x100;level++) {
   sum32+=doADC(0);
   for (tt1=0;tt1<64;tt1++) { ; }   }

        pt=sum32>>8;
        WrEEPROM(5,pt);
        pt=sum32>>16;
        WrEEPROM(6,pt);
```

```
    sum32=0;
  for (level=0;level<0x100;level++) {
   sum32+=doADC(1);
   for (tt1=0;tt1<64;tt1++) { ; }   }

  pt=sum32>>8;
    WrEEP


 if (ordre==0x85)
{


      SetGain0(databuff._byte[8]);
      SetGain1(databuff._byte[9]);

            t_basethh=databuff._byte[3];
      t_baseth=databuff._byte[4];
      t_basetl=databuff._byte[5];


      nptsH =  databuff._byte[6];
      nptsL =  databuff._byte[7];
      t_nbpts= nptsL + 256*nptsH;

      _asm
           movlw        1
           movwf        FSR0H,0
           clrf    FSR0L,0
      clrf    INDF0,0
           decfszFSR0L,1,0

           incf    FSR0H,1,0
           clrf    FSR0L,0
      clrf    INDF0,0
           decfszFSR0L,1,0



           incf    FSR0H,1,0
           movlw        127
           movwf        FSR0L,0
      movlw        0xAA
```

```
        decfszFSR0L,1,0

        movwf       INDF0,0

        movlw       5
        movwf       FSR0H,0
        clrf   FSR0L,0
  clrf  INDF0,0
        decfszFSR0L,1,0

        incf   FSR0H,1,0
        clrf   FSR0L,0
clrf    INDF0,0
        decfszFSR0L,1,0


        incf   FSR0H,1,0
        movlw       127
        movwf       FSR0L,0
  movlw       0xAA
        movwf       INDF0,0
        decfszFSR0L,1,0

        movwf       INDF0,0



        incf   nptsH,1,1

        movwf       savFSR1H,1
        movf FSR1L,0,0
        movwf       savFSR1L,1

        movf FSR2H,0,0
        movwf       savFSR2H,1
        movf FSR2L,0,0
        movwf       savFSR2L,1

        movlw       0x1
        movwf       FSR0H,0
        movlw       0x5
        movwf       FSR2H,0
```

```
            clrf    FSR2L,0

            movlw       0x3
            movwf       FSR1H,0
            clrf    FSR1L,0

            movlw       4
            movwf       acqcyc,1

            clrf    vH1,1
            clrf    vH2,1
      _endasm

         synchro

                    b5 =1 :
                    b4 =1 :
                    b3 =1 :



{


  T0CON=0b10010110;
  TMR0H= 256-183;
  TMR0L= 0;
  INTCON=0b10100000;
  timeout=0;


   {


{
    doADC0();
    while (ADRES>=level)
      { doADC0();  if (timeout==15) break; }
         while (ADRES<level)
      { doADC0();  if (timeout==15) break; }  }
        else  {
```

```
        while (ADRES<level)
      { doADC0();  if (timeout==15) break; }
        while (ADRES>=level)
      { doADC0();  if (timeout==15) break; }  }
   }

  else  {

    doADC1();
        while (ADRES>=level)
      { doADC1();  if (timeout==15) break; }
        while (ADRES<level)
      { doADC1();  if (timeout==15) break; }  }
       else {

    doADC1();
        while (ADRES<level)
      { doADC1();  if (timeout==15) break; }
        while (ADRES>=level)
      { doADC1();  if (timeout==15) break; }
}
   }

  INTCON=0;
  }


 {
      _asm
            movlw       0x01
            movwf       ADCON0,0
            bsf         ADCON0,1,0
            nop
            nop

loopv0:     movff ADRESL,POSTINC0

            bcf         STATUS,0,0
            rlcf   vH1,1,1
            rlcf   vH1,0,1
            iorwf  ADRESH,0,0
```

```
            decfszacqcyc,1,1



            movf vH1,0,1
            movwf       POSTINC1,0
            clrf   vH1,1
            movlw       4
            movwf       acqcyc,1



        movlw       1
        decfszWREG,1,0

            nop



         bsf
        ADCON0,1,0



            movfft_basethh, tt3
          movff     t_baseth, tt2
          movff     t_basetl, tt1
tpo1:  decfsztt1,1,1

            decfsztt2,1,1

            decfsztt3,1,1




            decfsznptsL,1,1

            decfsznptsH,1,1

         _endasm
    }

  {
         asm
```

```
            movwf       FSR1H,0

      movlw 0x05
            movwf       ADCON0,0
            bsf         ADCON0,1,0
            nop

loopv1:     movffADRESL,POSTINC2

            bcf         STATUS,0,0
            rlcf   vH1,1,1
            rlcf   vH1,0,1
            iorwf ADRESH,0,0
            movwf       vH1,1

            decfszacqcyc,1,1



            movf vH1,0,1
            movwf       POSTINC1,0
            clrf   vH1,1
            movlw       4
            movwf       acqcyc,1

      movlw       1
      decfszWREG,1,0

            nop


bsf         ADCON0,1,0

            movfft_basethh, tt3
         movff      t_baseth, tt2
         movff      t_basetl, tt1
      decfsztt1,1,1

            decfsztt2,1,1

            decfsztt3,1,1
```

```
        decfsznptsL,1,1

        decfsznptsH,1,1

      _endasm
  }

{
     _asm

     movlw       0x05
        movwf       ADCON0,0
        bsf         ADCON0,1,0
        nop
        nop
        movffADRESL,POSTINC0

        bcf         STATUS,0,0
        rlcf   vH1,1,1
        rlcf   vH1,0,1
        iorwf ADRESH,0,0
        movwf       vH1,1


        decfszacqcyc,0,1


        movf vH1,0,1
        movwf       INDF1,0
        clrf   vH1,1

     movlw       1
     decfszWREG,1,0

        nop

     movlw       0x01
        movwf       ADCON0,0
        bsf         ADCON0,1,0
        nop
        nop
        nop
```

```
        bcf           STATUS,0,0
        rlcf    vH2,1,1
        rlcf    vH2,0,1
        iorwf ADRESH,0,0
        movwf         vH2,1


        decfszacqcyc,1,1
        bra           stoh3

        movlw         0x4
        addwfFSR1H,1,0
        movf vH2,0,1
        movwf         POSTINC1,0
        movlw         0x4
        subwfFSR1H,1,0
        clrf    vH2,1
        movlw         4
        movwf         acqcyc,1


    movlw       3
    decfszWREG,1,0
        bra
        nop
movfft_basethh, tt3
      movff      t_baseth, tt2
      movff      t_basetl, tt1
    decfsztt1,1,1
        bra
        decfsztt2,1,1

        decfsztt3,1,1



        decfsznptsL,1,1

        decfsznptsH,1,1
     _endasm
    }
```

```
        _asm
            movf savFSR1H,0,1
            movwf       FSR1H,0
            movf savFSR1L,0,1
            movwf       FSR1L,0

            movf savFSR2H,0,1
            movwf       FSR2H,0
            movf savFSR2L,0,1
            movwf       FSR2L,0
        _endasm



        send1_pts = t_nbpts;
        s_shift = 0x100;
        send1H_pts = send2_pts =send2H_pts = sendoffs=0;
  }



      }
    }
  }
 }
}//end ProcessIO
/****************************************************/
```

7.2 USER - INTERFACE CODE:

```
Option Explicit

'default Vidpid of the board
Const VIDPID = "vid_04d8&pid_000c"

'vars for measure corrections
Dim delta0_0, delta0_1, delta1_0, delta1_1, ec0, ec1
'var for data display
Dim mk_x, mk_y          'pour deplacement souris
Dim pt_deb%, pt_fin%    'deb/fin courbe
Dim zoomchange%, avertis%, timout%

'aquisition parameters
Dim tbase_unit$(4)
Dim t_sample As Single
Dim t_dure As Single     'durée totale dans l'unité choisie
Dim t_durems As Single   'durée totale en ms
Dim t_pts As Long
Dim t_unit%              'unité de temps de t_dure

Dim fgain0 As Single
Dim fgain1 As Single  'coeff gain = 1 +/- n/1000

Dim p0, p1               'valeur des 2 pointeurs

Dim indexmoy1 As Byte, indexmoy2 As Byte
```

```
'data array for usb transmit
Dim sd(64) As Byte          '64 emitted bytes
Dim rec(64) As Byte         'received
Dim MSB(128) As Byte        '4 msb bits
Dim shf(64) As Byte

Dim inpipe, outpipe As Long

Dim Voie0(1 To 500) As Integer  'all samples
Dim Voie1(1 To 500) As Integer

Dim Moy0(1 To 500, 8) As Integer    'averaged samples
Dim Moy1(1 To 500, 0 To 7) As Integer



Private Sub Majtime()
'Displays time parameters up to date after modification
 t_unit = unitScr.Value
 Tunittxt.Text = tbase_unit(t_unit)
 t_sample = Val(timetxt.Text)
 t_pts = Val(nbptstxt.Text)

 'reset zoom 1:1
 pt_deb = 1
 pt_fin = t_pts

 t_dure = (t_pts * t_sample)
```

```
'sample time unit in uS
If t_unit = 1 Then t_sample = t_sample * 1000
If t_unit = 2 Then t_sample = t_sample * 1000000

'check if too small interval
If t_sample < passcr.Max Then
  gocmd.Enabled = False
 Else
  gocmd.Enabled = True
End If

'calc total time in mSec
t_durems = t_dure
If t_unit = 0 Then t_durems = t_durems / 1000
If t_unit = 2 Then t_durems = t_durems * 1000
If t_durems < 0.05 Then t_durems = 1

If (t_dure >= 1000) And (t_unit < 2) Then
  t_unit = t_unit + 1
  t_dure = t_dure / 1000
End If

duretxt.Text = Str$(t_dure)
'display time unit
durunittit.Text = tbase_unit(t_unit)
End Sub
```

```
Private Sub Drawcourb()
Dim w%, h2%, q%, larg As Long
Dim x, y

 pix.Cls
 'Plot backscreen and reticule
 pix.BackColor = RGB(0, 0, 84)
 pix.ForeColor = &H8000000E
 pix.DrawStyle = 0
 h2 = -30 + pix.Height / 2
 w = pix.Width

 'Plot axis
 pix.DrawStyle = 2
 pix.Line (50, h2 - 2)-(-10 + w, h2 - 2)    'horyzont

If xyChk Then
 If w >= 0 Then
  pix.Line (w / 2, 0)-(w / 2, pix.Height)          'vertical
 End If
End If

For y = 0 To pix.Height Step (pix.Height / 8)
 For x = 0 To w Step (w / 10)
  pix.PSet (x, y - 9)
Next x, y

For x = 0 To w Step (w / 10)
```

```
Next x

larg = pt_fin - pt_deb
pix.DrawStyle = 0

'///////////// Y(t) N O R M A L  M O D E //////////////
'Plot channel 0 (if valid)
If v1Chk.Value Then
 'Write calibre1 on screen
 pix.CurrentX = 10
 pix.CurrentY = pix.Height - 350
 pix.ForeColor = RGB(10, 220, 10)
 If cal0 > 4 Then
  pix.Print "Ch1:" + FormatNumber(4000 / cal0, 0) + "mv/div"
 Else
  pix.Print "Ch1:" + FormatNumber(4 / cal0, 1) + "v/div"
 End If

 'Plot channel 0
 For q = pt_deb To pt_fin
  'Voie 0 - Y pixel
  y = h2 - 20 + (pix.Height / 1024) * Voie0(q)
  'Voie 0 - X pixel
  If q <= pt_deb + 1 Then
    pix.PSet (0, y), &HFF00&
   Else
    pix.Line -((w / larg) * (q - pt_deb), y), &HFF00&
  End If
```

```
End If

'Plot channel 1 (if valid)
If v2Chk.Value Then
 'Write calibre2 on screen
 pix.CurrentX = 1500
 pix.CurrentY = pix.Height - 350
 pix.ForeColor = RGB(250, 30, 30)
 If cal1 > 4 Then
  pix.Print "Ch2:" + FormatNumber(4000 / cal1, 0) + "mv/div"
 Else
  pix.Print "Ch2:" + FormatNumber(4 / cal1, 1) + "v/div"
 End If

 'Plot samples
 For q = pt_deb To pt_fin
  'Voie 1
  y = h2 - 20 + (pix.Height / 1024) * Voie1(q)
  If q <= pt_deb + 1 Then
   pix.PSet (0, y), pix.ForeColor
  Else
   pix.Line -((w / larg) * (q - pt_deb), y), pix.ForeColor
  End If
 Next q
End If

'Write time-base
pix.CurrentX = w * 0.77
```

```
 pix.ForeColor = RGB(210, 210, 210)
 y = ((1 + pt_fin - pt_deb) * t_durems) / (10 * t_pts)
 If y < 1 Then
  pix.Print "T=" + FormatNumber(y * 1000, 1) + "uS/div"
 Else
  pix.Print "T=" + FormatNumber(y, 1) + "mS/div"
 End If
End If  'fin de si xy
End Sub


Private Sub cal0Scr_Change()
Select Case cal0Scr.Value
 Case 0: cal0txt.Text = "Auto"
        cal0Scr.Tag = 0
        cal0 = 1
 Case 1: cal0txt.Text = "+/-16v"
        cal0Scr.Tag = 0
        cal0 = 1
 Case 2: cal0txt.Text = "+/-8v"
        cal0Scr.Tag = 1
        cal0 = 2
 Case 3: cal0txt.Text = "+/-4v"
        cal0Scr.Tag = 2
        cal0 = 4
 Case 4: cal0txt.Text = "+/-2v"
        cal0Scr.Tag = 4
        cal0 = 8
```

```
        cal0Scr.Tag = 6
        cal0 = 16
  Case 6: cal0txt.Text = "+/-.5v"
        cal0Scr.Tag = 7
        cal0 = 32
 End Select
End Sub

Private Sub cal1Scr_Change()
Select Case cal1Scr.Value
  Case 0: cal1txt.Text = "Auto"
        cal1Scr.Tag = 0
        cal1 = 1
  Case 1: cal1txt.Text = "+/-16v"
        cal1 = 1
        cal1Scr.Tag = 0
  Case 2: cal1txt.Text = "+/-8v"
        cal1Scr.Tag = 1
        cal1 = 2
  Case 3: cal1txt.Text = "+/-4v"
        cal1Scr.Tag = 2
        cal1 = 4
  Case 4: cal1txt.Text = "+/-2v"
        cal1Scr.Tag = 4
        cal1 = 8
  Case 5: cal1txt.Text = "+/-1v"
        cal1Scr.Tag = 6
        cal1 = 16
```

```
        cal1Scr.Tag = 7
        cal1 = 32
 End Select
End Sub



Private Sub calCmd_Click()
Dim sended, recv, z As Long

outpipe = MPUSBOpen(0, VIDPID, "\MCHP_EP1", 0, 0)
inpipe = MPUSBOpen(0, VIDPID, "\MCHP_EP1", 1, 0)

 If MsgBox("You are about to clear actual calibration factors. Continue
?", vbOKCancel, "Calibration") = vbOK Then
  'Envoie commande de Raz !
  sd(0) = &H80
  If MPUSBWrite(outpipe, sd(0), 16, sended, 1000) = 0 Then GoTo
usberror1
  'Recup offset comme accuse de reception
  If MPUSBRead(inpipe, rec(0), 64, recv, 2000) = 0 Then GoTo
usberror
 Else
  Exit Sub
 End If   'fin gains

 If MsgBox("Zero calibration - Unplug both CH1 & CH2 input signals,
and press OK !", _
     vbOKCancel, "Calibration") = vbOK Then
```

```
  sd(0) = &H83
  If MPUSBWrite(outpipe, sd(0), 16, sended, 1000) = 0 Then GoTo usberror1
  If MPUSBRead(inpipe, rec(0), 64, recv, 2000) = 0 Then GoTo usberror
  'teste accusé de reception = ok
  MsgBox "Zero calibration done.", vbInformation, "Calibration"
 End If

 If MsgBox("Set new gain factors ?", vbOKCancel, "Calibration") = vbOK Then
  ec0 = Val(InputBox("True CH1 peak to peak amplitude ?", "Channel 1 Gain"))
  ec1 = Val(InputBox("OscilloPIC measured CH1 peak to peak amplitude ?", "Channel 1 Gain"))
  If (ec0 <> 0) And (ec1 <> 0) Then
   fgain0 = fgain0 * (ec0 / ec1)
   If Abs(256 * (fgain0 - 1)) > 128 Then
    MsgBox "Incorrect !"
    Exit Sub
   Else
    MsgBox "Ok (G1= " + Str$(fgain0) + ")"
   End If

   ec0 = Val(InputBox("True CH2 peak to peak amplitude ?", "Channel 2 Gain"))
   ec1 = Val(InputBox("OscilloPIC measured CH1 peak to peak amplitude ?", "Channel 2 Gain"))
```

```
      fgain1 = fgain1 * (ec0 / ec1)
      If Abs(256 * (fgain1 - 1)) > 128 Then
       MsgBox "Incorrect !"
       Exit Sub
      Else
       MsgBox "Ok (G2= " + Str$(fgain1) + ")"
      End If
     'Envoie coeff de gain !
     sd(0) = &H81
     sd(1) = Round(128 + 256 * (fgain0 - 1))
     sd(2) = Round(128 + 256 * (fgain1 - 1))
     If MPUSBWrite(outpipe, sd(0), 16, sended, 1000) = 0 Then GoTo
usberror1
     'Recup offset comme accuse de reception
     If MPUSBRead(inpipe, rec(0), 64, recv, 2000) = 0 Then GoTo
usberror
     MsgBox "Gain calibration done.", vbInformation, "Calibration"
    End If
   End If
 End If   'fin gains

z = MPUSBClose(inpipe)
z = MPUSBClose(outpipe)

GoTo finaquis

usberror1:
 MsgBox "Unplug & plug USB Cable !", vbCritical, "Signal
```

```
 GoTo finaquis
usberror:
 MsgBox "Unplug & plug USB Cable !", vbCritical, "Signal
Interrupted"
finaquis:
End Sub

Private Sub Cmdzm_Click()
Dim l%, c%
Dim xx, xe

 If (1 + pt_fin - pt_deb) < t_pts Then
  zoomchange = 1
  c = (pt_fin + pt_deb) / 2
  l = (pt_fin - pt_deb) * 1.3
  l = 10 * Round(l / 10)

  If l >= t_pts Then l = t_pts
  pt_fin = c + (l / 2)
  pt_deb = c - (l / 2)

  If pt_fin > t_pts Then
   pt_fin = t_pts
   pt_deb = pt_fin - l
  End If
  If pt_deb <= 0 Then pt_deb = 1

  posScr.Max = t_pts - (pt_fin - pt_deb)
```

```
If vertChk Then
  xe = (pt_fin - pt_deb + 1) * (t_durems / t_pts) 'duree de l'ecran
  xx = pt_deb * (t_durems / t_pts)              'temps debut ecran
  cursTxt.Text = FormatNumber(xx + xe * (p0 / pix.Width), 2) +
durunittit.Text
  xx = xe * (Abs(p1 - p0) / pix.Width)
  deltaTxt.Text = FormatNumber(xx, 2) + durunittit.Text
  If xx > 1 Then
    freqTxt.Text = FormatNumber(1000 / xx, 0) + "Hz"
   Else
    freqTxt.Text = FormatNumber(1 / xx, 2) + "kHz"
  End If
 End If

 Drawcourb
 zoomchange = 0
 End If
End Sub


Private Sub Cmdzp_Click()
Dim l%, c%
Dim xx, xe

 zoomchange = 1

 c = (pt_fin + pt_deb) / 2
 l = (pt_fin - pt_deb)
```

```
 l = 10 * Round(l / 10)

 pt_fin = c + (l / 2)
 pt_deb = c - (l / 2)

 posScr.Max = t_pts - (pt_fin - pt_deb)

If vertChk Then
  xe = (pt_fin - pt_deb + 1) * (t_durems / t_pts)
  xx = pt_deb * (t_durems / t_pts)
  cursTxt.Text = FormatNumber(xx + xe * (p0 / pix.Width), 2) +
durunittit.Text
  xx = xe * (Abs(p1 - p0) / pix.Width)
  deltaTxt.Text = FormatNumber(xx, 2) + durunittit.Text
  If xx > 1 Then
   freqTxt.Text = FormatNumber(1000 / xx, 0) + "Hz"
   Else
   freqTxt.Text = FormatNumber(1 / xx, 2) + "kHz"
  End If
 End If

 Drawcourb
 zoomchange = 0
End Sub


Private Sub Command1_Click()
If Command1.Caption = "&Continous" Then
```

```
Command1.Caption = "&Pause"
ElseIf Command1.Caption = "&Pause" Then
Timer1.Enabled = False
Command1.Caption = "&Continous"
End If
End Sub

Private Sub duretxt_KeyUp(KeyCode As Integer, Shift As Integer)
Dim nd%, dur As Single

 nd = Val(duretxt.Text)
 If nd <> 0 Then
  gocmd.Enabled = True
  dur = nd / t_pts
 Else
  gocmd.Enabled = False
 End If
End Sub



Private Sub aqu_voies()
Dim sended, recv, s0, s1 As Long
Dim q%, p%, b%, bloc%, z%
Dim shf0_16%, shf1_16%, shf0_1%, shf1_1%
Dim t_s As Single
Dim v0$, v1$

outpipe = MPUSBOpen(0, VIDPID, "\MCHP_EP1", 0, 0)
```

```
t_pts = 4 * Int(t_pts / 4)

'Lance l'ordre avec les sd(...) deja regles
'o0 = &h85
'o1-2 = seuil
'o3-4-5 = base temps
'o6-7 = nb points
'o8-9 = calibres 0 & 1
 sd(0) = &H85

' t_seuilh - b7  =1 : synchro
'           b6  =0:voie0, =1:voie1
'           b5  =1:montant, =0:descendant
'           b4 : voie 1 mesurée
'           b3 :  "   0 mesurée (si une seule 5uS !!!)
' t_seuilh 1:0 +seuill = niveau vu par ADC direct.

If MPUSBWrite(outpipe, sd(0), 16, sended, 1000) = 0 Then GoTo usberror1

'fait une pause
If t_durems > 2000 Then
  timout = 0
  pausetim.Enabled = True
  Do
   DoEvents
  Loop Until (timout = 1)
```

```
End If

For bloc = 0 To Int((t_pts - 4) / 64)
 'Recupere blocs de 64 octets de Voie 0
 If MPUSBRead(inpipe, rec(0), 64, recv, 16000) = 0 Then GoTo
usberror
 DoEvents
 'stocke les valeurs
  For b = 1 To 64
   If (b + 64 * bloc) < 500 Then Voie0(b + 64 * bloc) = rec(b - 1)
  Next b
Next bloc

'Recupere les 2 blocs de 64 octets poids forts Voie 0
If MPUSBRead(inpipe, MSB(0), 64, recv, 1000) = 0 Then GoTo
usberror
DoEvents
If MPUSBRead(inpipe, MSB(64), 64, recv, 1000) = 0 Then GoTo
usberror
DoEvents

'Rajoute les 2 bits de poids fort aux mesures
q = 0
b = 0
Do
  b = b + 1
  p = 4 * (192 And MSB(q))
  Voie0(b) = Voie0(b) + p    'ptr+1
```

```
  b = b + 1
  p = 16 * (48 And MSB(q))
  Voie0(b) = Voie0(b) + p     'ptr+2
  If (b = t_pts) Then Exit Do
  b = b + 1
  p = 64 * (12 And MSB(q))
  Voie0(b) = Voie0(b) + p     'ptr+3
  If (b = t_pts) Then Exit Do
  b = b + 1
  p = 256 * (3 And MSB(q))
  Voie0(b) = Voie0(b) + p     'ptr+4
  q = q + 1
Loop Until (b = t_pts)

For bloc = 0 To Int((t_pts - 4) / 64)
If MPUSBRead(inpipe, rec(0), 64, recv, 1000) = 0 Then GoTo usberror
 DoEvents

  For b = 1 To 64
   If (b + 64 * bloc) < 500 Then Voie1(b + 64 * bloc) = rec(b - 1)
  Next b
Next bloc

If MPUSBRead(inpipe, MSB(0), 64, recv, 1000) = 0 Then GoTo
usberror
DoEvents
If MPUSBRead(inpipe, MSB(64), 64, recv, 1000) = 0 Then GoTo
usberror
```

```
If MPUSBRead(inpipe, shf(0), 64, recv, 1000) = 0 Then GoTo usberror
DoEvents

If (shf(1) > 3) Or (shf(5) > 3) Or (shf(7) > 3) Then
  If avertis = 0 Then MsgBox ("Check calibration factors !")
  shf0_16 = 0
  shf1_16 = 0
  shf0_1 = 0
  shf1_1 = 0
  avertis = 1
 'Met des gains de 1 par defaut
 fgain0 = 1
 fgain1 = 1
Else
 shf0_16 = 512 - shf(0) - 256 * shf(1)
 shf1_16 = 512 - shf(2) - 256 * shf(3)
 shf0_1 = 512 - shf(4) - 256 * shf(5)
 shf1_1 = 512 - shf(6) - 256 * shf(7)
 'Prend les coeff de gain
 fgain0 = 1 + (shf(8) - 128) / 256
 fgain1 = 1 + (shf(9) - 128) / 256
End If

'Rajoute les 2 bits de poids fort aux mesures
q = 0
b = 0
Do
```

```
 p = 4 * (192 And MSB(q))
 Voie1(b) = Voie1(b) + p     'ptr+1
 If (b = t_pts) Then Exit Do
 b = b + 1
 p = 16 * (48 And MSB(q))
 Voie1(b) = Voie1(b) + p     'ptr+2
 If (b = t_pts) Then Exit Do
 b = b + 1
 p = 64 * (12 And MSB(q))
 Voie1(b) = Voie1(b) + p     'ptr+3
 If (b = t_pts) Then Exit Do
 b = b + 1
 p = 256 * (3 And MSB(q))
 Voie1(b) = Voie1(b) + p     'ptr+4
 q = q + 1          'index du tableau des msb
Loop Until (b = t_pts)

delta1_0 = (shf0_1 - shf0_16) / 15          'inversé????
delta1_1 = (16 * shf0_16 - shf0_1) / 15
delta0_0 = (shf1_1 - shf1_16) / 15          'inversé????
delta0_1 = (16 * shf1_16 - shf1_1) / 15

ec0 = delta0_1 + (cal0 * delta0_0)
ec1 = delta1_1 + (cal1 * delta1_0)

Listval.Clear

For b = 1 To t_pts
```

```
 If invchk(0) Then Voie0(b) = -Voie0(b)
 v0$ = Str$(Voie0(b) / (32 * cal0))
 v0$ = Space$(35 - Len(v0$)) + v0$

 Voie1(b) = fgain1 * (-512 + Voie1(b) + ec1)
 If invchk(1) Then Voie1(b) = -Voie1(b)
 v1$ = Str$(Voie1(b) / (32 * cal1))
 v1$ = Space$(35 - Len(v1$)) + v1$

 Listval.AddItem (Str$(b) + Space$(5) + Chr$(9) + v0$ + Chr$(9) +
v1$)
Next b

z = MPUSBClose(inpipe)
z = MPUSBClose(outpipe)

GoTo finaquis

usberror1:
 MsgBox "Unplug & plug USB Cable !", vbCritical, "Signal
Interrupted"
 GoTo finaquis
usberror:
 MsgBox "Unplug & plug USB Cable !", vbCritical, "Signal
Interrupted"
finaquis:
End Sub
```

```
Private Sub exporcmd_Click()
Dim b%, v$

 ComDlg.ShowSave
 If ComDlg.FileName <> "" Then
  Open ComDlg.FileName For Output As #1
  For b = 1 To t_pts
   v$ = Str$(b) + Chr$(9) + Str$(Voie0(b) / (32 * cal0))
   v$ = v$ + Chr$(9) + Str$(Voie1(b) / (32 * cal1))
   'Rajoute dans la liste
   Print #1, v$
 Next b
 Close #1

 MsgBox "Signal Values Export done"
 End If
End Sub

Private Sub Form_Load()
 tbase_unit(0) = "uS"
 tbase_unit(1) = "mS"
 tbase_unit(2) = "sec"

SSTab1.TabVisible(2) = False
SSTab1.TabVisible(3) = False
SSTab1.TabVisible(5) = False
```

```
 call = 1
 fgain0 = 1
 fgain1 = 1

 avertis = 0
 timout = 0

Majtime

 'raz bornes courbe
 pt_deb = 1
 pt_fin = t_pts

posScr.Max = t_pts - (pt_fin - pt_deb)

 Drawcourb

End Sub

Private Sub Form_Resize()
Dim w%, h%
  h = Me.Height
  pix.Height = h - 1880
  Listval.Height = h - 1880
  gocmd.Top = h - 1410
  Cmdzm.Top = h - 2130
  Cmdzp.Top = h - 2130
  Label2.Top = h - 2415
```

```
Label4.Top = h - 3885
cal1txt.Top = h - 2955
cal0txt.Top = h - 3615
cal0Scr.Top = h - 3675
cal1Scr.Top = h - 3000
Label1.Top = h - 4140
posScr.Top = h - 730
waittxt.Top = (h - 1880) / 2 - 200

w = Me.Width
pix.Width = w - 1345
Listval.Width = w - 1345
posScr.Width = w - 1350
SSTab1.Width = w - 275
gocmd.Left = w - 1160
Cmdzm.Left = w - 1040
Cmdzp.Left = w - 680
Label1.Left = w - 1050
Label2.Left = w - 905
Label3.Left = w - 980
Label4.Left = w - 980
cal0Scr.Left = w - 440
cal1Scr.Left = w - 440
cal0txt.Left = w - 1100
cal1txt.Left = w - 1100
waittxt.Left = (w / 2) - 1500
Listval.ColumnWidths = pix.Width / 3
```

```
 hcur2.X2 = pix.Width - 50
 vcur1.Y2 = pix.Height - 50
 vcur2.Y2 = pix.Height - 50
 Drawcourb
End Sub


Private Sub gocmd_Click()
Dim s0, s1 As Long
Dim q%, p%, b%, redo%, bloc%
Dim maxi, t_s As Single

If MPUSBGetDeviceCount(VIDPID) <> 1 Then
  MsgBox ("Connect USB Cable and click ok !")
  Exit Sub
End If

gocmd.Enabled = False
DoEvents

'Set the usb acquisition command sentence
'o0 = &h85
'o1-2 = raw level 16 bits
'o3-4-5 = time base 24 bits
'o6-7 = nb samples
'o8-9 = ranges 0 & 1

'Triggering Mode & level
```

```
'         b6  =0: ch1, =1: ch2
'         b5  =1:rising, =0:falling
'         b4 : ch 2 measured
'         b3 :  " 1 measured
' t_seuilh 1:0 +seuill = raw level on the CAN.
t_seuilh = 0
If srccomb.Text = "Ch 1" Then
  t_seuilh = 128
  q = 512 - 51.2 * seuilScr.Value / (16 / cal0)
  If froncmb.Text <> "Rising edge" Then t_seuilh = t_seuilh + 32
 Else
 If srccomb.Text = "Ch 2" Then
   t_seuilh = 128 + 64
   q = 512 - 51.2 * seuilScr.Value / (16 / cal1)
   If froncmb.Text <> "Rising edge" Then t_seuilh = t_seuilh + 32
 End If
End If

'Add selected chann bits
If v2Chk.Value Then t_seuilh = t_seuilh + 16
If v1Chk.Value Then t_seuilh = t_seuilh + 8

t_seuilh = t_seuilh + Int(q / 256)
t_seuill = q Mod 256

sd(1) = t_seuilh
sd(2) = t_seuill
```

```
'T=86? + 3.Tl + 770.Th + 197122.Thh (cycles)
'T en uS = NCycles / 6
t_s = 6 * (t_sample - passcr.Max)
sd(3) = Int(t_s / 197122)
t_s = t_s Mod 197122
sd(4) = Int(t_s / 770)
t_s = t_s Mod 770
sd(5) = t_s / 3

sd(3) = sd(3) + 1
sd(4) = sd(4) + 1
sd(5) = sd(5) + 1

'Samples Number
sd(6) = Int(t_pts / 256)
sd(7) = t_pts Mod 256

'Add ranges with
If cal0txt.Text = "Auto" Then cal0Scr.Tag = 0
sd(8) = cal0Scr.Tag
If cal1txt.Text = "Auto" Then cal1Scr.Tag = 0
sd(9) = cal1Scr.Tag

 indexmoy1 = (indexmoy1 + 1) And 7
For b = 1 To t_pts
  Moy0(b, indexmoy1) = Voie0(b)
  s0 = 0
  For bloc = 0 To 7
```

```
  Next bloc
  Voie0(b) = Round(s0 / 8) + ec0
 Next b
End If

If moychk(1) Then
indexmoy2 = (indexmoy2 + 1) And 7
For b = 1 To t_pts
  Moy1(b, indexmoy2) = Voie1(b)
  s1 = 0
  For bloc = 0 To 7
    s1 = s1 + Moy1(b, bloc)
  Next bloc
  Voie1(b) = Round(s1 / 8) + ec1
 Next b
End If
redo = 0
If cal0txt.Text = "Auto" Then
  maxi = 0.1
  For b = 2 To t_pts - 2
   If Abs(Voie0(b)) > maxi Then maxi = Abs(Voie0(b))
  Next b
  maxi = 512 / maxi
  If maxi > 30 Then
    redo = 1
    cal0 = 32
    cal0Scr.Tag = 7
  Else
```

```
     redo = 1
     cal0 = 16
     cal0Scr.Tag = 6
    Else
     If maxi > 8 Then
      redo = 1
      cal0 = 8
      cal0Scr.Tag = 4
     Else
      If maxi > 4 Then
       redo = 1
       cal0 = 4
       cal0Scr.Tag = 2
      Else
       If maxi > 2 Then
        redo = 1
        cal0 = 2
        cal0Scr.Tag = 1
       End If
      End If
     End If
    End If
  End If
End If

If cal1txt.Text = "Auto" Then
  maxi = 0.1
  For b = 2 To t_pts - 2
```

```
Next b
'fait le rapport au max d'echelle
maxi = 512 / maxi
If maxi > 30 Then
 redo = 1
 cal1 = 32
 cal1Scr.Tag = 7
Else
 If maxi > 16 Then
 redo = 1
 cal1 = 16
 cal1Scr.Tag = 6
 Else
  If maxi > 8 Then
   redo = 1
   cal1 = 8
   cal1Scr.Tag = 4
  Else
   If maxi > 4 Then
    redo = 1
    cal1 = 4
    cal1Scr.Tag = 2
   Else
    If maxi > 2 Then
     redo = 1
     cal1 = 2
     cal1Scr.Tag = 1
    End If
```

```
   End If
  End If
 End If
End If

If redo <> 0 Then
 sd(8) = cal0Scr.Tag
 sd(9) = cal1Scr.Tag
 aqu_voies
End If

If lisschk(0) Then
 For b = 3 To t_pts - 3
  s0 = Voie0(b - 2) + Voie0(b - 1) + Voie0(b) + Voie0(b + 1) +
Voie0(b + 2)
  Voie0(b) = s0 / 5
 Next b
End If

If lisschk(1) Then
 For b = 3 To t_pts - 3
  s0 = Voie1(b - 2) + Voie1(b - 1) + Voie1(b) + Voie1(b + 1) +
Voie1(b + 2)
  Voie1(b) = s0 / 5
 Next b
End If

desscourb:
```

```
Drawcourb

waittxt.Visible = False

fingocmd:
gocmd.Enabled = True
End Sub

Private Sub nbptsScr_Change()
 nbptstxt.Text = Str$(nbptsScr.Value)
 'Remet a jour la duree
 Majtime
End Sub


Private Sub pix_MouseDown(Button As Integer, Shift As Integer, x As
Single, y As Single)
Dim fact, xe, xx

 'Bouton droit appuyé pour curseurs
 If Button = 1 Then

  'curseurs verticaux
  If vertChk Then
   p0 = x
   If Abs(x - vcur1.X1) < Abs(x - vcur2.X1) Then
    vcur2.BorderStyle = 2
    vcur1.BorderStyle = 1
```

```
    vcur1.X2 = x
    p1 = vcur2.X1
   Else
    vcur1.BorderStyle = 2
    vcur2.BorderStyle = 1
    vcur2.X1 = x
    vcur2.X2 = x
    p1 = vcur1.X1
   End If
   xe = (pt_fin - pt_deb + 1) * (t_durems / t_pts) 'duree de l'ecran
   xx = pt_deb * (t_durems / t_pts)              'temps debut ecran
   cursTxt.Text = FormatNumber(xx + xe * (p0 / pix.Width), 2) +
durunittit.Text
   xx = xe * (Abs(p1 - p0) / pix.Width)
   deltaTxt.Text = FormatNumber(xx, 2) + durunittit.Text
   If xx > 1 Then
     freqTxt.Text = FormatNumber(1000 / xx, 0) + "Hz"
    Else
     freqTxt.Text = FormatNumber(1 / xx, 2) + "kHz"
    End If
  End If

  If horizChk Then
   If Abs(y - hcur1.Y1) < Abs(y - hcur2.Y1) Then
    hcur2.BorderStyle = 2
    hcur1.BorderStyle = 1
    hcur1.Y1 = y
    hcur1.Y2 = y
```

```
    Else
     hcur1.BorderStyle = 2
     hcur2.BorderStyle = 1
     hcur2.Y1 = y
     hcur2.Y2 = y
     p1 = hcur1.Y1
    End If
    If v1Chk Then
      fact = (2 * 16 / cal0)
    Else
      fact = (2 * 16 / cal1)
    End If
    p0 = fact * (0.5 - (y / pix.Height))
    p1 = fact * (0.5 - (p1 / pix.Height))
    cursTxt.Text = FormatNumber(p0, 2) + "v"
    deltaTxt.Text = FormatNumber(Abs(p1 - p0), 2) + "v"
    freqTxt.Text = ""
  End If
 End If


End Sub
```

REFERENCES

8. REFERENCES:

1. Chandan Bhunia, Saikat Giri, Samrat Kar, Sudarshan Haldar, and Prithwiraj Purkait,, 'A Low-Cost PC-Based Virtual Oscilloscope,' - IEEE Trans. Educ., Vol. 47, NO. 2, MAY 2004

2. Sulaiman, N., Mahmud, N.A., 'Designing the PC-Based 4-Channel Digital Storage Oscilloscope by using DSP Techniques', - 5th Student Conference on Research and Development, Pages: 1–7, Dec. 2007

3. Pereira, J.M.D., 'The history and technology of oscilloscopes', - IEEE Instrumentation & Measurement Magazine, Pages: 27 – 35 ,2006

4. Michael Predko, Myke Predko(2007), "Programming & Customizing the PIC Microcontroller", McGraw-Hill Professional Ltd., NewDelhi, April 2007.

5. Kirk Zurell(2006), 'C Programming for Embedded Systems', R&D Books, December 2006.

6. Circuit Ideas (2007), "DUAL CHANNEL PC – BASED DIGITAL OSCILLOSCOPE", Electronics for You, Jan 2007,pp- 60-67.

Online References

1. www.microchip.com (05-JAN-2009)
2. http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf (10-MAR -2009)
3. http://en.wikipedia.org/wiki/PIC_microcontroller (05-JAN-2009)
4. http://www.howstuffworks.com (22-JAN-2009)
5. http://www.picotech.com (17-FEB-2009)
6. http://books.google.co.in (14-FEB-2009)