International Journal of Sciences: Basic and Applied Research (IJSBAR)

ISSN 2307-4531
(Print & Online)

http://gssrr.org/index.php?journal=JournalOfBasicAndApplied

PIC Microcontroller USB Multi-Channel Data Acquisition and Control System

Abderrahmane Badis*

Electrical and Electronic Communication Engineer from UMBB (Ex.INELEC) Independent Electronics, Aeronautics, Propulsion, Well Logging and Software Design Research Engineer Takerboust, Aghbalou, Bouira 10007, Algeria

Email: Badis.dahmane@gmail.com

Abstract

This paper exposes a microcontroller-based data acquisition and control system that is particularly suitable for educators and students interested in developing lab experiments that do not require high-cost, high-performance data acquisition hardware and yet can benefit from the publically available development resources.

Keywords: PIC Microcontroller; data acquisition; Control; USB; HID Device.

1. Introduction

USB Data acquisition and control boards are becoming very popular these days thanks to the diversity of their application fields. They are essential for interfacing sensors and / or actuators with decision making devices such as Microcontrollers. Thus, data acquisition and control boards are used in monitoring and instrumentation applications involving machinery, process, environment, etc., and in automatic control applications. Many engineers and technicians use the data acquisition, control signals and communication cards to develop smart machines. For hobbyists, such boards offer a source of great passion and pleasure.

---

* Corresponding author.

Data acquisition and control system, as the name implies, is composed of hardware and Software that are used to collect environment information. Data acquisition (DAQ) is the process of measuring the physical phenomenon or physical property of an object under investigation using a dedicated hardware. The physical property or phenomenon could be temperature change, the intensity change of a light source, the pressure of a chamber, the force applied to an object etc [1]. The complete DAQ system consists of sensors, DAQ hardware, and a computer with programmable software.

The sensor performs a conversion of the physical phenomenon into an electrical signal [2]. This signal is further converted into digital numeric values by DAQ hardware (Sampling of the real world signals to generate digital data), which is assisted by a software program developed using various general purpose programming languages (LabVIEW, Visual Basic, C, MATLAB etc.). In addition to the hardware control, such DAQ software usually also includes data analysis, data visualization, and data storage algorithms.

The control function is achieved either through decision making relying on the signals acquired and processed by the acquisition system or simply by sending manual commands. External control signals are sent to manage, command, direct or regulate the behavior of other devices or systems often referred to as peripherals. The control system could be an integrated part of the acquisition system or a completely standalone hardware. Figure 1 illustrates a block diagram of a typical Data acquisition and control System.

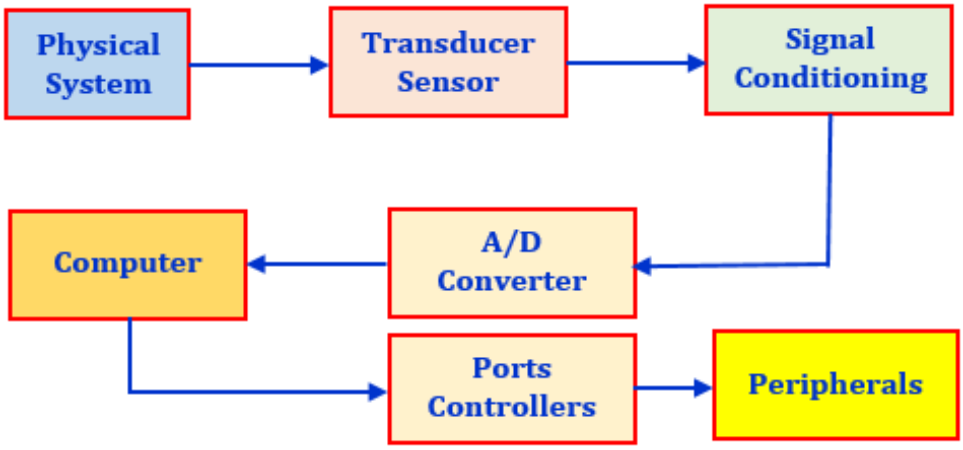

Figure 1: DAQ and Control System Block Diagram

2. The Hardware

This section gives an overview of the DAQ and Control hardware. The section starts with a description of the desired hardware functions, followed by covering the main criterion to select an appropriate microcontroller required to fulfill the needed tasks. Finally, a brief description of the DAQ and Control electronic circuit is covered.

2.1. Hardware Requirements

a. The data exchange between a PC and the hardware must be realized directly using Universal Serial Bus

(USB) without any additional components such as USB to serial converters that will allow computer connection for data exchange.

b. Must have integrated USB 2.0 Stack.
c. Self-power through the USB Port.
d. Eight (08) Analog Inputs, with an Analog to Digital Converter (ADC) with at least 10-bit resolution.
e. Sixteen (16) Digital Ports; eight (08) digital outputs for controls and eight (08) digital inputs for monitoring.
f. Last but not least, it has to be cost-effective.

2.2. Microcontroller Selection

Nowadays, very simple and low cost DAQ and Control devices could be realized using microcontrollers with integrated analogue-to-digital converters (ADC). A microcontroller is a small computer on a single integrated circuit containing a processing unit, memory, and programmable input/output peripherals.

Microcontrollers is the best cost-effective solution to build DAQ and Control hardware. The Selection of the appropriate microcontroller for our hardware is based the previously listed initial requirements. In regard to the stated needs, a Microchip PIC18F4550 microcontroller shown in Figure 2 was selected as our main platform.

Figure 2: Microchip PIC18F4550

This microcontroller is a member of the Microchip 28/40/44-Pin High-Performance and Enhanced Flash USB Microcontrollers with NanoWatt Technology family of microcontrollers that incorporates a fully-featured USB communications module with a built-in transceiver that is compliant with the USB Specification Revision 2.0 [2]. The microcontroller also includes an Enhanced USARTs, two Master Synchronous Serial Port (MSSP) modules capable of both Serial Peripheral Interface (SPI) and I2C (Master and Slave) modes of operation [2], in addition to two comparators.

This family of microcontrollers is ideal for applications requiring cost-effective, low-power USB applications with more code space and peripheral flexibility all set within a small package. The key characteristics of the selected PIC18F4550 microcontroller are summarized in Table 1 [3].

Table 1: Microchip PIC18F4550 Specifications

Parameter Name	Value
Program Memory Type	Flash
Program Memory Size	32 KB
Data RAM Size	2048 B
Data RAM Type	SRAM
Data ROM Size	256 B
Data ROM Type	EEPROM
Data Bus Width	8 bits
Maximum Clock Frequency	48 MHz
Number of I/Os	35 I/O
Timers	4 Timers
Number of ADC Channels	13 Channels
ADC Resolution	10 bits
Interface Types	EUSART, I2C, SPI, SPP and USB (V2.0)
Temperature Range	From - 40 C to + 85 C
Operating Voltage Range	2 V to 5.5 V
Pin Count	40 Pins
Package Type	PDIP-40

2.3. Electronic Circuit

The circuit is very straight forward even for beginners in microcontroller electronics. Figure 3 illustrates an optimized circuit diagram with a minimum configuration for a usable USB Device that will achieve the hardware pre-requirements.

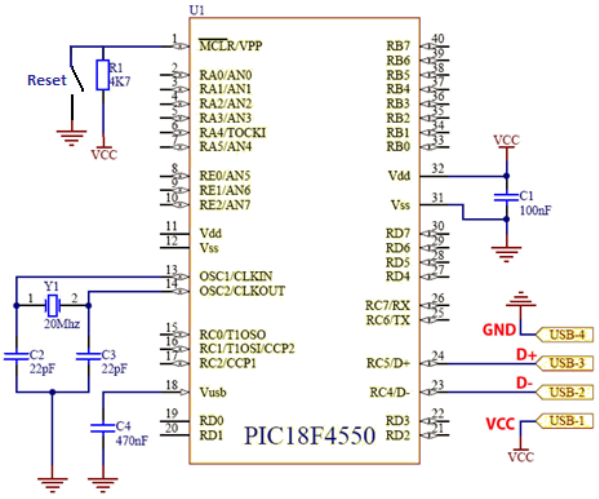

Figure 3: DAQ and Control Circuit Diagram

The PIC18F4550 is 'Bus Powered'; this means that the device is drawing its power from the USB host so no power regulation is required (Less Components). A Female Type-A USB Socket is used, the user is free to use the model that will best fit his needs (e.g. Type-B) the Pinouts for USP Type "A" is presented in Figure 4.

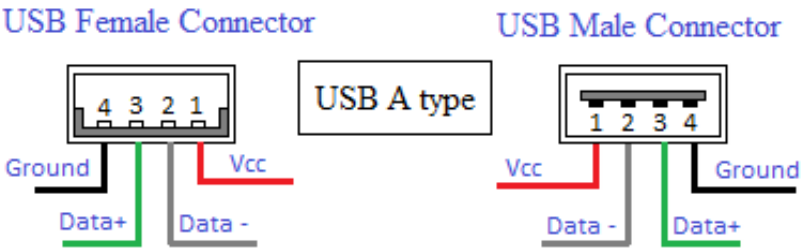

Figure 4: USB Type-A Connectors Pinouts

The 0.47µF capacitor is required to stabilize the internal 3.3V regulator that provides power to the internal transceiver and provide a source for the internal/external pull-ups. In practice any value between 0.22µF and 0.47µF won`t make any odds to the functioning of the circuit, the datasheet recommends a value of 0.22 µF (±20%). An external Crystal is required for the PIC to be able to execute the firmware instructions and to use the on-board USB module. A 20 MHz crystal is used, this clock speed of oscillation will be multiplied internally to the required 48 MHz clock source necessary for the USB full-Speed operation.

2.4. Important Notes

a. To comply with the hardware requirement; Port-B will be configured as an output port and Port-D and C will be configured as input ports.
b. All the resistors in the circuit diagram are ¼ Watt and all the capacitors are ceramic.
c. The capacitor between VCC and ground is critical for the operation stability of the PIC.

2.5. Summary of the Microcontroller Ports Usage

Table 2 summarizes the microcontroller ports (A through E) usage by pins.

Table 2: 18F4550 Ports usage by Pins

Port / Bit	07	06	05	04	03	02	01	00
PORT-A	-	OSC2	AN5	N/U	AN3	AN2	AN1	AN0
PORT-B	D/O	D/O	D/O	D/O	D/O	D/O	D/O	D/O
PORT-C	D/I	D/I	USB D^+	USB D^-	-	D/I	D/I	D/I
PORT-D	D/I	D/I	D/I	D/I	D/I	D/I	D/I	D/I
PORT-E	N/U	-	-	-	$\overline{MCLR}$	AN7	AN6	AN5

Table 2 Legend:

- OSC2: Oscillator Pin 2.
- ANX: Analog Input "X"
- D/O: Digital Output
- D/I: Digital Input
- USB D +: USB data "-" line
- USB D -: USB data "+" line
- "-": Unimplemented Read as "0"
- $\overline{\text{MCLR}}$: Master Clear Pin

3. The Firmware

The firmware enables the acquisition and generation of Analogue and digital signals, this allows the transfer of data between the chip and the computer using USB serial bus. The source Code for the microcontroller is written in an IDE named as PIC18 Simulator IDE. This is powerful application made by OshonSoft [4] that supplies Microchip microcontroller users with user-friendly graphical development environment for Windows with integrated Simulator, Pic Basic Compiler, Assembler, Disassembler and Debugger.

Before getting into the firmware details, it is worth understanding the USB HID Class (Human Interface Device). The USB protocol divides all peripherals in different classes, according to data transfer requirements and limitations. The HID class, which comprises keyboards and mice, is perfect for interfacing small microcontrollers.

Data is exchanged by means of an Output Report and an Input Report, with direction respectively from and to the PC. Feature Reports are used in both directions as well. In summary, the data exchange is achieved based on four (04) types of reports:

a. Output Reports: used to receive data from the PC to the Microcontroller.
b. Input Reports: used to send data from the Microcontroller to the PC.
c. Output Feature: used to receive configuration information from the PC to the Microcontroller.
d. Input Feature: used to send configuration information from the Microcontroller to the PC.

It is also important to understand the way the USB HID Class are identified when plugged to a computer. The PID (Product ID) together with the VID (Vendor ID) that identifies the manufacturer of the device, are forming a unique identifier for the USB HID. When the device first enumerates, the Operating System will store the VID and PID combination for it.

3.1. Firmware Flowchart

Figure 5 summarizes the workflow and the different processes performed by the firmware.

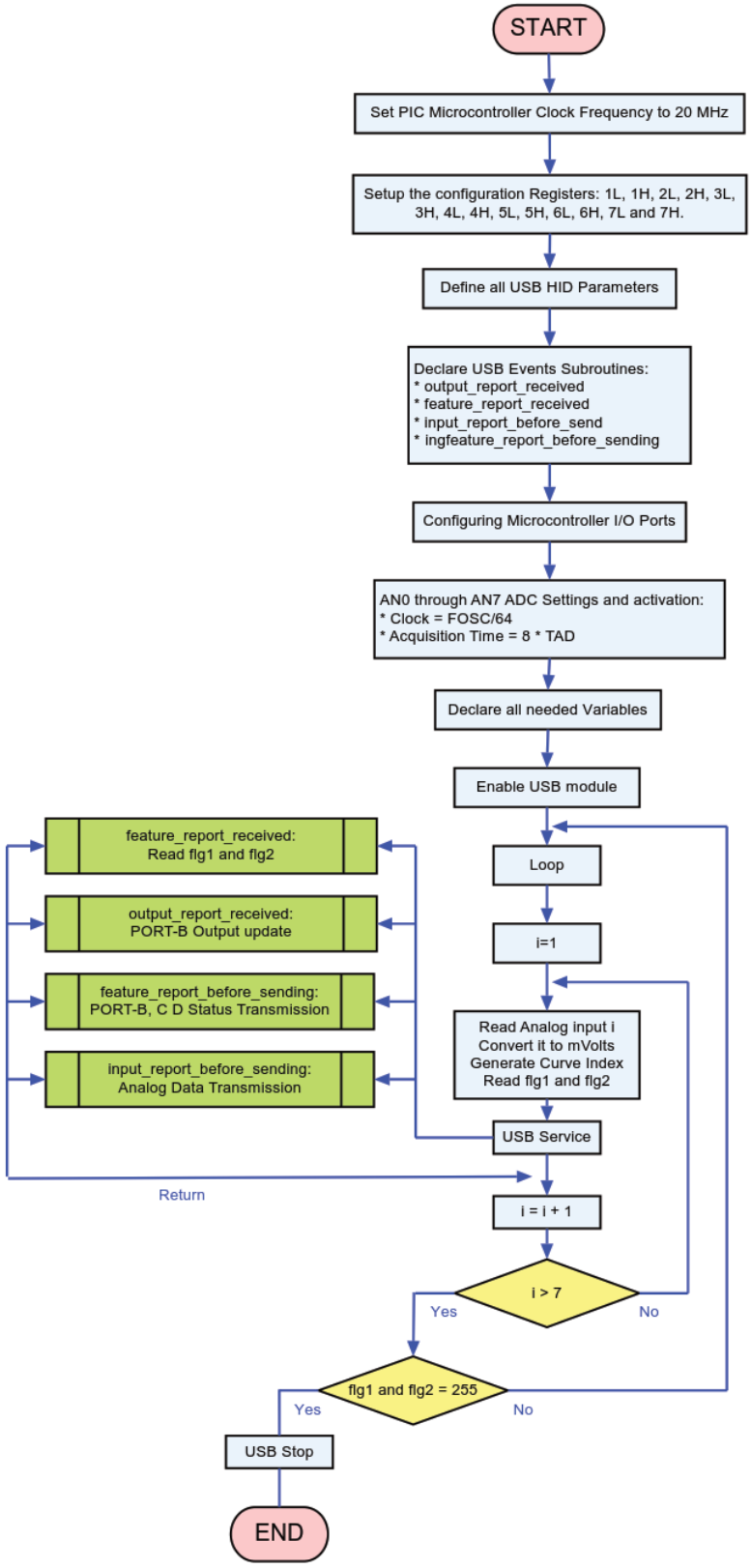

Figure 5: Firmware Flowchart

3.2. The firmware source code reflecting the above flowchart is described below

Setup all configuration parameters at the beginning of the basic program. First, the clock frequency of the Microcontroller is specified as per the circuit diagram:

Define CLOCK_FREQUENCY = 20

Then, there are fourteen (14) configuration registers that need to be set correctly, these override the default setting of the different configuration bits within the PIC microcontroller, it is important to mention that Port-B is configured as a digital port in this case through the 3H register:

Define CONFIG1L = 0x24

Define CONFIG1H = 0x0c

Define CONFIG2L = 0x3e

Define CONFIG2H = 0x00

Define CONFIG3L = 0x00

Define CONFIG3H = 0x81

Define CONFIG4L = 0x80

Define CONFIG4H = 0x00

Define CONFIG5L = 0x0f

Define CONFIG5H = 0xc0

Define CONFIG6L = 0x0f

Define CONFIG6H = 0xe0

Define CONFIG7L = 0x0f

Define CONFIG7H = 0x40

Define all USB HID parameters (numeric and string constants) that will be used to identify the hardware by the host PC:

UsbSetVendorId 0x1234

UsbSetProductId 0x1234

UsbSetVersionNumber 0x1122

UsbSetManufacturerString "Badis Corporation"

UsbSetProductString "A.BADIS"

UsbSetSerialNumberString "Firmware Example"

All Data exchange is implemented for USB Feature, Input and Output reports with packets composed of 8 bytes of data.

Output Report Data sent from the PC will be automatically stored in the system *UsbIoBuffer(0-7*) array. Feature Report Data sent from the PC application will be stored in the *UsbFtBuffer(0-7)* array. The subroutines that will be called by the USB firmware after these two events are specified with *UsbOnIoOutGosub* and *UsbOnFtOutGosub* statements as follow:

UsbOnIoOutGosub output_report_received

UsbOnFtOutGosub feature_report_received

Similarly, two other subroutines specified with *UsbOnIoInGosub* and *UsbOnFtInGosu*b statements are defined; the USB firmware will call them prior to sending Input and Feature reports to the host PC:

UsbOnIoInGosub input_report_before_sending

UsbOnFtInGosub feature_report_before_sending

Microcontroller Ports are configured as inputs or outputs using CONFIGPIN statement:

ConfigPin PORTA = Input

ConfigPin PORTB = Output

ConfigPin PORTC = Input

ConfigPin PORTD = Input

ConfigPin PORTE = Input

The next step is to setup up the different parameters of the Analog to Digital Converter within the PIC microcontroller registers (ADCON0, ADCON1 and ADCON2) depending on the application. The below code will enable AN0 through AN7:

Define ADC_CLOCK = 6 ‘ADC Clock value is set to FOSC/64 (T_{AD} = 64 * 0.05 µSec = 3.2 µSec)

Define ADC_SAMPLEUS = 25 'Set ADC Acquisition Time to 8 * T_{AD}

ADCON1 = 0 'AN0 through AN7 ADC pins activated, the result format is left justified and V_{dd} is the reference voltage.

The set of variables required are declared as follow:

Dim i As Byte 'Variable used in Loop Index.

Dim analogw As Word 'Two byte variable used to store the result after ADC conversion.

Dim analog As Single 'Four bytes single precision floating point numbers in a modified IEEE754 32-bit format.

Dim curve As Byte 'Curve Name sent through USB.

Dim usbstpflg1 As Byte 'This flag is used to disconnect the Hardware from the PC with UsbStop statement.

Dim usbstpflg2 As Byte 'This flag is used to disconnect the Hardware from the PC with UsbStop statement.

Dim portbf As Byte 'This byte variable will be used to read the status of the Port-B.

Dim portcf As Byte 'This byte variable will be used to read the status of the Port-C.

Dim portdf As Byte 'This byte variable will be used to read the status of the Port-D.

PIC chip integrated USB module is enabled with UsbStart statement. After its execution the hardware will be recognized by the Operating System as a generic HID device (enables USB handshaking):

UsbStart

The main body of the firmware will perform the following tasks: read the 7 analog inputs, associate a generated curve name for each, send them through USB, disconnect the Hardware if two 255 commends are received and send port status on request.

loop:

For i = 0 To 7

Adcin i, analogw 'Read the analog port "i" in 10 bit resolution Word.

analog = analogw * 4.8876 'The resolution of 10 bit ADC is 1023 steps, i.e. 5000/1023 = 4.88mv

curve = 49 + i 'Associate an ASCII code for the analog port read CHR(49) = 1…CHR(56) = 8

```
        UsbService ’Call USB service routine to process each of the USB events.
Next i
‘The below is used to disconnect the Hardware from the PC if the PC sent Feature Report Byte-0 and 1 both equal 255.
Select Case usbstpflg1
        Case 255
                If usbstpflg2 = 255 Then UsbStop
        Case Else
                Goto loop
EndSelect
Goto loop
End
```

The two first bytes of the Feature Report received by the microcontroller is used uniquely for the hardware disconnect request, Byte-1 and 2 are stored in flag1 and 2 respectively.

```
'//COMPUTER SENT FEATURE REPORT
feature_report_received:
usbstpflg1 = UsbFtBuffer(0) ‘Received Feature Report Byte-0 is used as a first command for USB STOP
usbstpflg2 = UsbFtBuffer(1) ‘Received Feature Report Byte-1 is used as a first command for USB STOP
Return
```

The Output Report received by the microcontroller is used to control the PORT-B outputs.

```
'//COMPUTER SENT DATA REPORT
output_report_received:
PORTB = UsbIoBuffer (0) ‘Update PORT-B Output
```

```
portbf = UsbIoBuffer (0)  'Save the PORT-B Status

Return
```

The Feature Report received by the computer is used for a real time monitoring of PORT-B, C and D.

```
'//MICROCONTROLER SENT FEATURE REPORT

feature_report_before_sending:

UsbFtBuffer(0) = portbf ‘Send PORT-B Status

portcf =PORTC

UsbFtBuffer(1) = portcf ‘Send PORT-C Status

portdf =PORTD

UsbFtBuffer(2) = portdf ‘Send PORT-D Status

Return
```

The Input Report received by the computer is used to send the seven analog data along with the corresponding curve names.

```
'//MICROCONTROLER SENT DATA REPORT

input_report_before_sending:

UsbIoBuffer(0) = analog.LB ‘Send the First Byte of ADC Single Result

UsbIoBuffer(1) = analog.HB ‘Send the Second Byte of ADC Single Result

UsbIoBuffer(2) = analog.3B  ‘Send the Third Byte of ADC Single Result

UsbIoBuffer(3) = analog.4B ‘Send the Fourth Byte of ADC Single Result

UsbIoBuffer(4) = 65 ‘Send the First Character of the curve Name, in this case CHR(65) = A

UsbIoBuffer(5) = 78 ‘Send the Second Character of the curve Name, in this case CHR(78) = N

UsbIoBuffer(6) = 48 ‘Send the Third Character of the curve Name, in this case CHR(48) = 0

UsbIoBuffer(7) = curve ‘Send the Fourth Character of the curve Name, in this Case CHR(curve)
```

Return

The compiled code for this microcontroller will generate the output in the form of a small hex file. That hex file (firmware) will be loaded into the EPROM of the PIC18F4550 microcontroller using a special hardware called "Microcontroller Programmer".

4. The Software

4.1. Software Key Operation

Unlike serial and parallel interfaces, USB needs a complicated enumeration process in order establish a communication channel. The so called USB frameworks are code libraries that hide the details of the protocol, so that is possible to use the interface very quickly to communicate with a PC.

One of the most used comes from Microchip, but has a closed license and a somewhat complicated structure; in addition to it more or less all the compiler vendors give a framework with their highly priced products.

The PIC18 Simulator IDE comes with a USB support add-on that consists of the USB ActiveX control (or DLL library) for the development of the PC application that will communicate with the hardware. With this library the designed application will be able to send Feature Report, to request Feature Report, to send Output Report and to request Input Report from the hardware.

An application program on the PC was developed using Visual Basic [5]. The GUI is designed to show all the functionality of the hardware. A master timer function is used to detect the HID Device identified by its corresponding PID and VID. As soon as the correct HID Device is detected, it enables the rest of the program sections: acquisition of the analog data, enables buttons, or visual effects...etc. The reader is encourage to read more about the different functions used by the ActiveX Control or library used. Personalized libraries could be created as well but it is not the scope of the present paper.

The most important task performed by the software that needs particular attention is the correct interpretation of the data formats received or sent. This is performed through data conversion processes. All characters are sent as bytes containing ASCII codes, whereas numerical values are single precision floating point numbers, 7 digits precision in a modified IEEE 754 standard illustrated in Figure 6 [6].

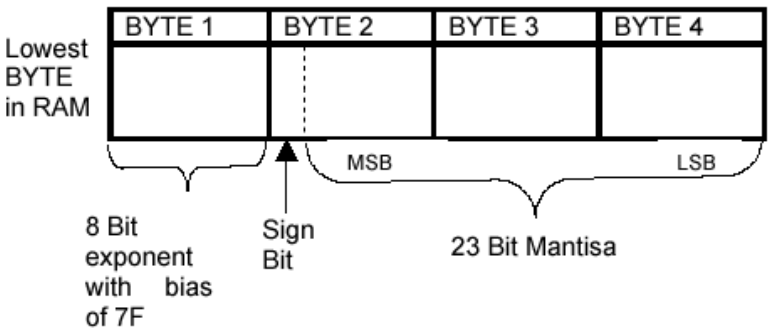

Figure 5: Modified IEEE 754 Standard

To perform the required format conversions, a module is created containing set of functions with specific tasks as summarized below, these latter are called from the master code or within the module:

Function-1: ASCII Codes to Character Conversion

```
Public Function CurveName (ByVal C0 As Byte, ByVal C1 As Byte, ByVal C2 As Byte, ByVal C3 As Byte) As String

‘Generates a Curve Name from a 4 Bytes Word

‘Usage:   CN = CurveName (C0, C1, C2, C3)

Let CurveName = UCase$(Chr$(C0) + Chr$(C1) + Chr$(C2) + Chr$(C3))

End Function
```

Function-2: Integer to Binary Conversion

```
Public Function BinConv (ByVal x As Long) As String

'Convert Long integers to binary

'Usage:   F$ = BinConv (Number)

Dim Index As Integer

Dim Temp As String

Let Temp = ""

'Start translation to binary

Do

'Check whether it is 1 bit or 0 bit

If x Mod 2 Then

    Let Temp = "1" + Temp

Else

    Let Temp = "0" + Temp
```

```
End If

'Normal division     7/2 = 3.5

'Integer division     7\2 = 3

    Let x = x \ 2

If x < 1 Then Exit Do

Loop

'Define the Function final result

If Len(Temp) >= 8 Then

    Let BinConv = Temp

Else

    For Index = 1 To 8 - Len (Temp)

       Let Temp = "0" + Temp

    Next Index

    Let BinConv = Temp

End If

End Function
```

Function-3: Binary to Integer Conversion

```
Public Function BinConvInt (ByVal BinStr As String) As Integer

'Convert Binary to Integer

'Usage:   S = BinConvInt (Binary String)

Dim Index As Integer

Dim BV As String
```

```
'Exponent Calculation
Let BinConvInt = 0
For Index = 8 To 1 Step -1
   Let BV = Mid(BinStr, Index, 1)
   Let BinConvInt = BinConvInt + (Val(BV) * 2 ^ (8 - Index))
Next Index
End Function
```

Function-4: Binary to a Modified 32-Bit IEEE-754 Conversion

```
Public Function BinConvSing(ByVal BinStr As String) As Single
'Convert Binary to Single Precision
'Usage:   S = BinConvSing(Binary String)
'Binary String is a Modified Single Precision Floating Point IEEE-754 Format
Dim Index As Integer
Dim BV As String
'Fraction calculation
Dim Fraction As Single
Let Fraction = 0
For Index = 1 To 23 Step 1
Let BV = Mid(BinStr, Index + 9, 1)
Let Fraction = Fraction + (Val(BV) * 2 ^ (-Index))
Next Index
Let Fraction = 1 + Fraction
```

```
'Exponent Calculation

Dim Exponent As Single

Let Exponent = 0

For Index = 8 To 1 Step -1

Let BV = Mid(BinStr, Index, 1)

Let Exponent = Exponent + (Val(BV) * 2 ^ (8 - Index))

Next Index

Let Exponent = 2 ^ (Exponent - 127)

'Calculation of the final Result

If Mid(BinStr, 9, 1) = "0" Then

Let BinConvSing = Fraction * Exponent

Else

  Let BinConvSing = -Fraction * Exponent

End If

End Function
```

Function-5: 04 Bytes to a Modified 32-Bit IEEE-754 Conversion

```
Public Function GetSingle(ByVal B0 As Byte, ByVal B1 As Byte, ByVal B2 As Byte, ByVal B3 As Byte) As Single

'In Basic Compiler they must be assigned:  4H    3H    HB    LB

'Bear in mind that Byte-04 is 0x7F Biased.

'Usage:  Sng = GetSingle(B0,B1,B2,B3)

Dim SB0 As String

Dim SB1 As String
```

```
Dim SB2 As String

Dim SB3 As String

Dim BinString As String

'Conversion to Hexadecimal

Let SB0 = "&H" + Hex(B0)

Let SB1 = "&H" + Hex(B1)

Let SB2 = "&H" + Hex(B2)

Let SB3 = "&H" + Hex(B3) + "-" + "&H7F"

'Generate the Binary String

Let BinString = BinConv(Val(SB3)) + BinConv(Val(SB2)) + BinConv(Val(SB1)) + BinConv(Val(SB0)) 'Calls Function-2

'Calculate the Single Value

Let GetSingle = Round(BinConvSing(BinString), 7) 'Calls Function-4

End Function
```

4.2. GUI Application Designed

Figure 6 demonstrates the designed GUI, this represents a basic application that could be re-designed to fit a particular application as illustrated in figure 7.

Example of a Practical Application:

The modification of the previously designed HID-Device with additional electronic circuits, Sensors and the programming of a powerful application illustrated in fig.7, extended the capabilities of the design to perform the below tasks:

- Real time monitoring of an air tank pressure and Temperature.
- Software Control of Three Solenoid Valves, this could be performed either manually by clicking the buttons or automatically by programming the sequence of opening and closing events.
- Real time display of both Pressure and Temperature.
- Record of all the data in a binary file.

- Real time monitoring of the hardware ports status.
- Possibility of saving the Graphs at any time in BMP or PDF files.

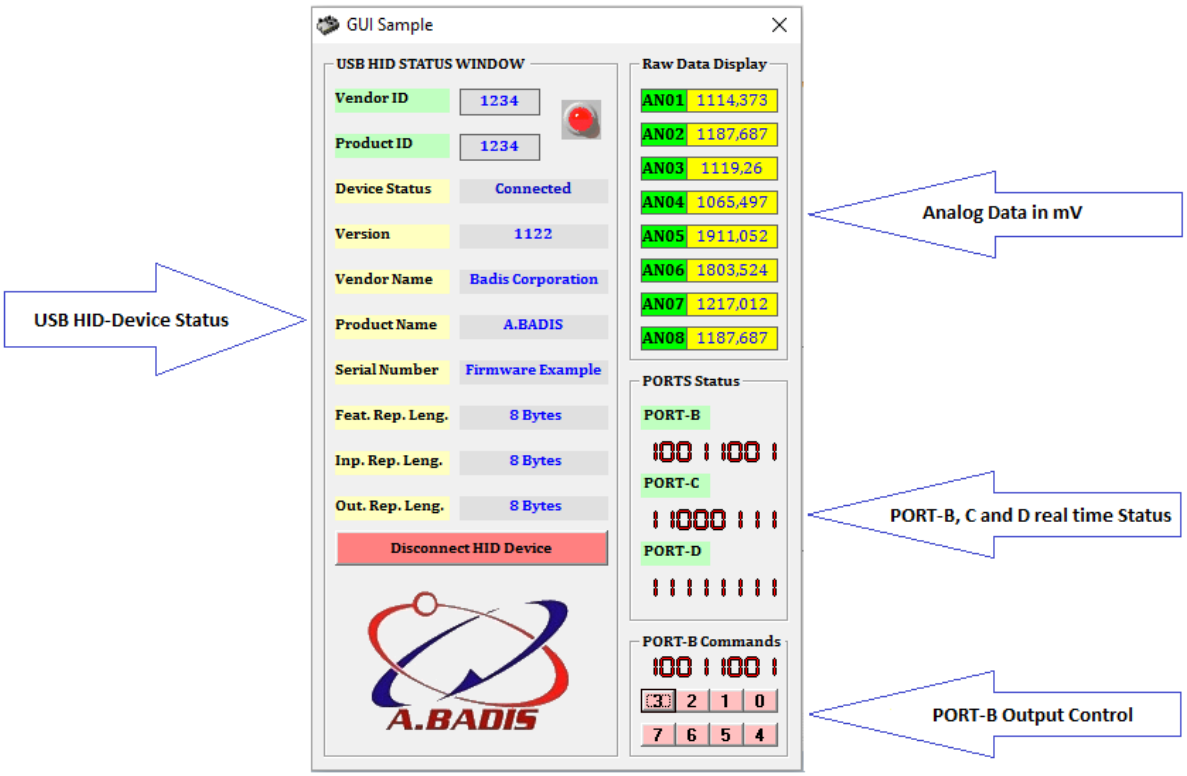

Figure 6: GUI Sample

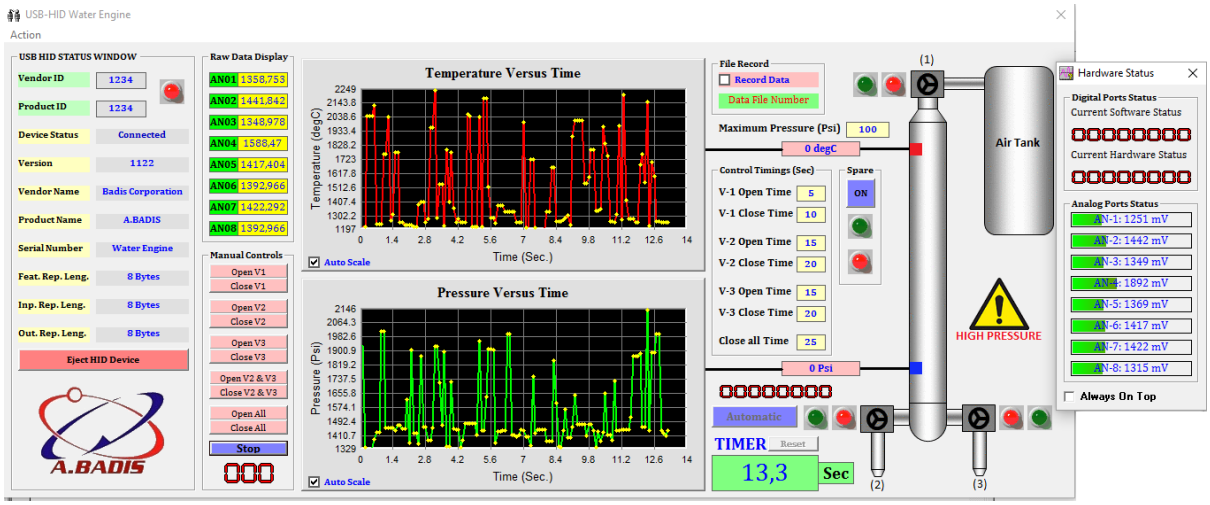

Figure 7: Designed Application for a Practical Application

5. Conclusion

This paper provided an overview of a low-cost data acquisition and control system that consists of Hardware build around the 18F4550 PIC Microcontroller, a firmware developed using the PIC18 Simulator IDE and a

Windows GUI developed using Visual Basic. The Serial communication capabilities of the PIC microcontroller and the Visual Basic Library allowed the exchange of sensory data and status signals between the PIC Microcontroller and Computer. The capabilities of this low-cost data acquisition and control system were illustrated through a practical example that incorporates real time Pressure and Temperature Display, Solenoid Valves control and Port Status Monitoring.

References

[1] Design Of A USB-Based Data Acquisition System by Samiran Maiti (IJRET).

[2] A Flexible Microcontroller-Based Data Acquisition Device by Darko Hercog and Bojan Gergič.

[3] Microchip PIC18F2455/2550/4455/4550 Data Sheet.

[4] PIC18 Simulator IDE BASIC Compiler Reference Manual, Licensed to Badis Abderrahmane.

[5] Design and Development of Low Cost Multi-Channel USB Data Acquisition System for the Measurement of Physical Parameters by Nungleppam Monoranjan Singh, Kanak Chandra Sarma and Nungleppam Gopil Singh

[6] website, http://www.ccsinfo.com/faq.php?page=mchp_float_format, last date accessed 1\11\2016.