Low Cost Multi-Channel Data Acquisition System with Graphical HMI

Antonio Hernández Zavala, Oscar Camacho Nieto,

Cornelio Yañez Márquez, Osvaldo Espinosa Sosa

Centro de Investigaciones en Computación – Instituto Politécnico Nacional MEXICO Av. Juan de Dios Batiz s/n UP Zacatenco, México DF.
antonioh_z@prodigy.net.mx
oscarc@cic.ipn.mx

Abstract. With the objective of reduce the breach of technology that first world countries have to manage industry's process control, in this paper a low cost multi-channel data acquisition system integrated with a human-machine interface HMI is developed. To validate the efficiency of our MCDAS system with regard to conventional systems, as case of study a tank of water with temperature sensors and a heater are used to supervise and modify its state with multiple parameters to read from an acquisition card developed. Finally it was detected that our hardware system, that has integrated a 10 bit, 16 channel analog to digital converter, a Harvard architecture RISC processor at 20 MHz and RS-232 serial transmission, is a proprietary technological development which is versatile and is realized at low cost, and will be used for more cases of study at the institute.

Keywords. SCADA, Data Acquisition Card, Control, Supervision.

1 Introduction

In real-time processes at industry, low cost supervision systems and reliable data management on an easy of use interface are required. Almost all industries have a computer on their hands, and their users are familiarized with graphical environment because its simplicity to visualize changes on the environment of the questioned variable, and respond to the changes given by the user. A system that allows a direct interaction with real processes by means of a desktop computer is commonly called SCADA (System of Control And Data Acquisition).

SCADA's are not new at industry field, there is too much work done, since small systems dedicated to a single process up to big ones that allow the monitoring of a complete production plant. Among SCADA developers, we have National Instruments with LabView software, which have all the necessary tools for signal conditioning, data acquisition modules and its graphical friendship programming language able to

© L Sánchez, O. Espinosa (Eds.)
Control, Virtual Instrumentation and Digital Systems.
Research in Computing Science 24, 2006, pp. 85-93

interconnect its wide catalog of products. This is a very complete product line which can be adapted to any real process, but software and hardware modules are very expensive and ties users to their specific technologies as happen with the rest of commercial developers.

A good alternative to have an SCADA for monitoring a process, at low price, is to create the complete platform for that specific process; letting options for more cases, by using a robust general purpose object oriented programming language for the creation of an HMI for the process viewing and data transmission. Semiconductors are also required to obtain a binary value that represents that real world variable. Their low price is added to the advantages that represent the creation of this system. The acquisition card uses a general purpose microcontroller, ADC, DAC and signal conditioning that uses solid state circuits

The system and the process of schema on Fig. 1, are a tank of water with six temperature sensors placed on different points, they are signal conditioned prior to analog to digital conversion, and read by a general purpose microcontroller [2] which performs the task of send the sensors measures to the PC, via serial communication. The PC contains the HMI that is dedicated to plot measures and modify the environment of the variable. On Fig. 2 is a picture of the physical system.

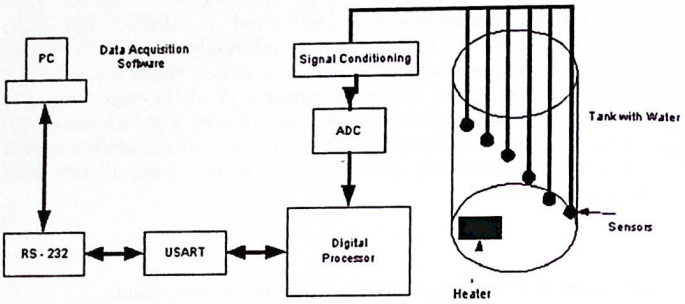

Fig. 1. Block diagram of the system.

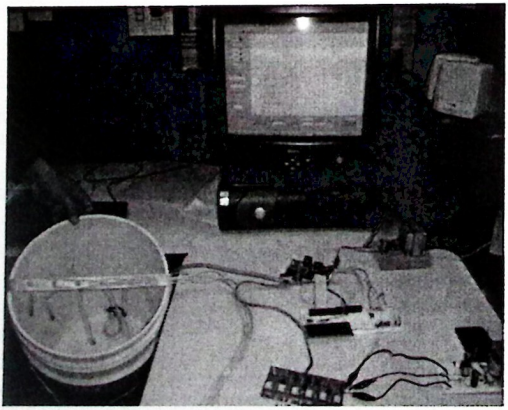

Fig. 2. Physical Prototype System.

This paper gives a description about the realization of all the modules of a SCADA system which is intended to be used as measurement and controlling system for different experiments at research centre. The rest of the document is divided in four parts: following part shows how translation of water temperature into a comprehensive voltage or signal conditioning stage, this includes the hardware used for sensing and the 10 bit analog to digital converter. On the next part digital data treatment is discussed. Fourth part is dedicated to the HMI characteristics on a general purpose programming language, in this case C++Builder^{R}. Finally at fifth part results and conclusions are reviewed.

2 Signal Conditioning

The physical variable to be represented is temperature, in order to obtain a voltage level representation at low cost, six LM335 temperature sensors with calibrated output according to manufacturer data [1][5] were used; the resolution of the signal given is 10mV per grade in a linear behavior; calibration is at 2.98V when temperature is 25°C, this values are used as reference ($Vout_{T0}$, T_0) for:

$$Vout_T = Vout_{T_0} \times \frac{T}{T_0} \tag{1}$$

Here $Vout_T$ is the voltage level obtained for a temperature T using the previous reference values. The output of the sensor is connected to an operational amplifier voltage follower [4], which gives us an output for each channel as in Fig. 3.

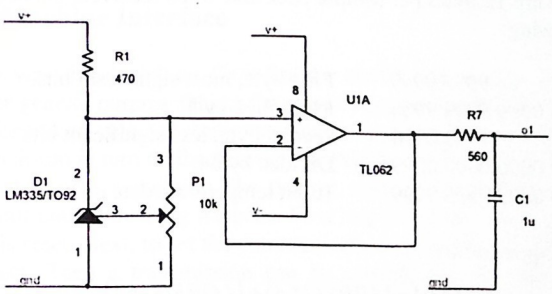

Fig. 3. Signal conditioning for a single temperature sensor.

In order to realize the digitalization of the variable, a 10 bit successive approximations ADC embedded in the COP8 MCU is used. It takes 15 clock cycles to have a sample and the result is available at the ADRSTH and ADRSTL 8-bit registers [2]. Since a 20MHz clock frequency for the microcontroller is used, it takes 0.75us to get one sample. This can be configured to different sampling rates by its internal registers. Depending on the number of channels that are read, sampling time gets slower while number of channels increase.

3 Digital Decoding of Data Packages into Comprehensive Data

This part is the most important because if data is not correctly decoded, there will be great mistakes on readings as on Fig. 4 which shows (a) bad ordered data for ambient temperature and (b) noise from electricity on water. To correct (a), the following algorithm is proposed, for (b), a low pass antialiasing filter is used.

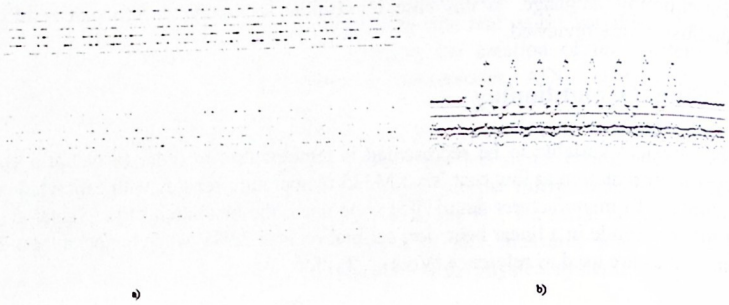

Fig. 4. Disorders at data decoding, a)Disordered data decoding, b)Noise affecting measures

A sample of each channel is 10 bit length so it does not fix on a byte, then two bytes are used as they are read from converter registers ADRSTH and ADRSTL, the first contain the 8 most significant bits, the second contain the 2 least significant bits in the upper 2 bits [6]. Then, a sample is on a single word size, for six samples per channel there are 12 bytes per sample pack and when received, the decoding process is like the following:

9999 9999	First byte, most significant bits.
0000 0099 9999 9900	Multiplied by 4.
9900 0000	Second byte, less significant bits.
0000 0000 0000 0099	Divided by 64.
0000 0099 9999 9999	10 bit temperature data on a single word.

That can be expressed as:

$$swDATA = (HIREG * 4) + (LOREG / 64) \tag{2}$$

This is the task that will be applied to every word in the package received and will loop until no data received. On Fig. 5a and 5b the flow diagram of the timer interrupt is shown divided in two parts: the first part, is the beginning, where some possible fails on data reception are handled in a way that let us work; on part *b* a loop until no data available is executed for data decoding with (2); plotting of the points and unit visualization are realized with (3), where temperature from a single word data is obtained at °C and visualized on the graphics screen of HMI.

$$T°C = \frac{swDATA * 298}{800} - 273 \quad (3)$$

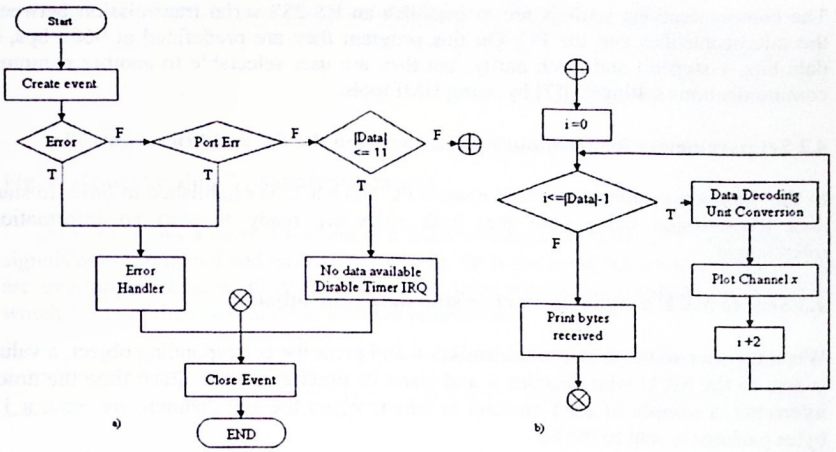

Fig. 5. a)Failure detection, b)Data decoding until no data available

4. Human Machine Interface

HMI Software was developed on C++ Builder 5.0™, which is a robust graphical environment for general purpose Object Oriented Programming. In this language, when something is done to an object at runtime, an event is generated and calls its associated subroutine, which can in turn do the desired task [8][9].

The program works something like this: at start up it initializes all parameters required and wait until something happens. First step is for the user, who must ensure that the MCU is reset. Next, to set the communications parameters double click on the respective object. Then a transmission can be started, and the plot of the curves representing the actual state of the variables is seen on screen graphics. Transmission will loop until the user stops it.

While a transmission is running, the user can save on a file, previously created with the HMI, the measures obtained on a given time period. The software is able to do the following tasks that are detailed later:

1. Set parameters for communications between the PC and microcontroller.
2. Send to MCU a signal to start or stop data transmission.
3. Decoding of data packages into comprehensive data.
4. Changes number of samples per second, from 3 to 50 per channel.

5. Saves on file a desired segment of signals.
6. Reproduces signals previously recorded for analysis

4.1 Communications Settings

The communications settings are to establish an RS-232 serial transmission between the microcontroller and the PC. On this program they are predefined at 9600 bps, 8 data bits, 1 stop bit and Even parity, but they are user selectable to another common communications settings[2][7] by using HMI tools.

4.2 Set parameters for Communications between the PC and Microcontroller

In this function a connection line between PC and MCU is established in order to start data transmission being sure that both sides are ready to start an information interchange.

4.3 Sent to MCU a signal to start or stop data transmission

When the user wants to start a transmission and press the corresponding object, a value is sent to the MCU who decodes it and starts its timer execution. Each time the timer interrupts, a sample of each channel is taken. When the six channels are read, a 12 bytes package is sent to the PC.

On the PC, a timer which generates an interrupt every time period given is activated and makes transmission a loop of port buffer readings until the user stops it or no more data available.

When the user stops transmission another byte is sent to MCU and indicates that the timer must be stopped and values reset.

4.5 Saves on file a desired segment of signals

The user can choose to record some signals on a time interval by selecting *Record signal* on the toolbar; a file must be first created with HMI [8][9]. This procedure can be stopped when the user wants it.

4.6 Reproduces signals previously recorded for analysis

A signal which has been recorded can be reproduced and viewed later, with no connection to the acquisition card, or some signal processing algorithm can be applied to test signal state without affecting real system [8][9].

5. Results and Tests

Several tests have been realized since the beginning of the project, we encountered many fails and correct them. All measure plots presented here were performed by this HMI software, note that some plots are reproduced from file and its measures have an offset, which is intended for analysis and visualization but it does not seen on real time plots, if they have the same measure, they are overlapped.

On Fig. 6 there is a failure, this occurs when the heater is turned on, and it reduces voltage on the whole circuit. Therefore it was supplied independent from the digital circuitry.

Fig. 6. Heater Stealing Voltage (reproduced).

Fig. 7a shows a correct reading at a uniform temperature of 22°C, note that these signals are reproduced and have an offset. Fig. 7b is the same but on real time, all lines are overlapped because all of them have the same temperature, except for lower line which is out of the water and has ambient temperature.

a) b)

Fig. 7. Ambient temperature measures. a) Reproduced, b) Real time.

When the heater is on, water temperature is increasing its temperature from the effect of the heater on it, which can be seen on Fig. 8, note that the higher temperature correspond to the sensor which is closer to the heater, the lower line comes from the sensor which is more distant from heater.

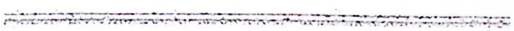

Fig. 8. Measures when heater is on.

6. Conclusions

With the development and construction of the MCDAS system (Fig. 1) with HMI, proprietary technology was developed at low cost. About the programming language, this option is still expensive, but it is very versatile and every program made can be scaled up by using all the tools that language can provide, besides this, the

communications protocol can be proposed as simpler as user wants or application requires. By the hardware side, great cost savings are reached, this is because the whole platform was realized since the beginning using low cost parts, but we are still looking for a cheaper and robust microcontroller and we have in mind that for great volumes, cost gets reduced.

With the correct configuration of used semiconductors, as a case of study we used a tank with water on which temperature was monitored using sensors on different places at real time. This is able of change water temperature by a heater. We found that the system effectively keep track of the variable behavior and can affect the state of the system easily.

With this project, we have a monitoring tool that can be used with other industrial processes previously signal conditioned to operate from 0 to 5 Vcc, reducing considerably the hardware acquisition cost. In fact, this tool will help us to have industrial equipment as on first world countries but at lower price and totally designed by us.

This project is still on growing development, we are looking for the addition of new tools that allow an easier use and understanding of the referenced process.

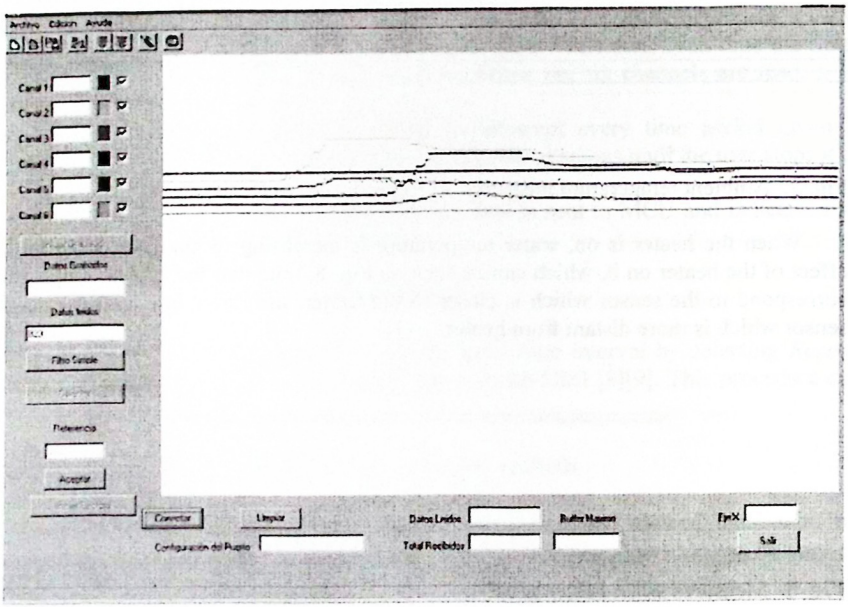

Fig. 9. Graphical interface for MCDAS.

References

1. LM135/LM235/LM335, LM135A/LM235A/LM335A Precision Temperature Sensors. National Semiconductor Datasheet. DS005698 (2000)

2. COP8CBR9/COP8CCR9/COP8CDR9 8-Bit CMOS Flash Microcontroller with 32k Memory, Virtual EEPROM, 10-Bit A/D and Brownout. User Manual National Semiconductor. DS101374 (2002)
3. Hernández A., Algoritmo de Lógica Difusa para Corrección de Error en Aplicaciones de Tiempo Real, MC Tesis, Centro de Investigaciones en Computación IPN, México, 2004.
4. Boylestad, N.: Electrónica: Teoría de Circuitos y Dispositivos electrónicos. Prentice-Hall. 8th. Edition. (2003)
5. Maloney, T. J.: Electrónica Industrial Moderna. Prentice-Hall. 5ª Edicion. (2006)
6. Tocci, R. J.: Sistemas Digitales, principios y aplicaciones. Prentice-Hall. 6th Edition. (1996)
7. Denver, A.: Serial Communications in Win32. Microsoft Windows Developer Support. (1995)
8. Hollingworth, J., Butterfield, D., Swart, B., Alsop, J.: Borland C++ Builder 5.0 Developers guide. SAMS Publishing. (2001)
9. Schildt, H.: Borland C++ Builder: The Complete Reference. Mc Graw Hill. (2001)