computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

REAL TIME DATA ACQUISITION SYSTEM FOR THE ECP-EPP PARALLEL PORT BASED ON PIC16F877 MICROCONTROLLER

John A. Kalomiros

Technical and Educational Institute of Serres, Greece,
Department of Informatics and Communications
P.O. Box 1006, 62110, Serres, Greece
ikalom@de.sch.gr

Abstract: *The design of a simple and low cost 10-bit data acquisition system is presented which makes use of the peripherals of a PIC16F877 microcontroller, interfacing with a personal computer using the extended capabilities of the parallel port. The system is integrated with a visual programming tool based on LabVIEW data acquisition software, which provides design flexibility and real time signal processing capabilities. An optimum assembly code for the PIC microcontroller allows for a free-running mean sampling rate of 100KSps on a Pentium PC running Windows XP OS. This system can be an example of a low cost integrated approach for data acquisition that includes a microcontroller, a personal computer and visual measurement software. The system can be the basis of a A/D interface for many measurement applications and can also be seen as an educational paradigm in itself. An effective and fast DAC solution is also presented in full integration with the microcontroller and the computer parallel port.*

Keywords: *Data acquisition, Microcontrollers, Real-time systems, Computer interfaces, Parallel port, LabView.*

1. INTRODUCTION

Computer based data acquisition systems have become very common during the last fifteen years and prevail in many measurement applications that follow a low cost modular design [1,2]. They usually consist of a PCI or PXI computer card with a number of configurable analog input and output channels, digital inputs and outputs as well as inputs for triggering and timing signals [3]. The system is characterized by the total number of channels, its bit analysis and sampling rate. A basic twelve bits of analysis with a sampling rate of about 150 KSps may rise to a total cost in the range of several hundred Euros. The system is flexible but still its use requires some programming skills and a basic understanding of the architectural details in terms of configurable registers and memory mapping. A particular drawback of such a design is the need of a dedicated computer system and limited portability.

An alternative to the above are devices external to the computer system, connected to the computer through a serial interface, like a COM port or USB port. RS232 devices have a widespread use, especially in educationally oriented applications, in various types of “Computer Based Laboratories” or CBLs. They usually function as data loggers rather than real time data acquisition systems, due to the limited baud rate, but can also monitor slowly varying signals in real time. USB 2.0 devices support high transfer rates and can compete with boards based on PCI bus but they cost twice as much as the later [4]. However, along with IEEE 1394 (firewire) external data acquisition devices they constitute a smart and flexible if expensive solution, able to cover most future needs.

The parallel interface was traditionally used as a printer or scanner port before it was supplanted by the USB port. It is still located in most desktop computer motherboards and provides an easy, fast and well documented interface for data interchange with all kinds of custom devices. The initial printer port evolved into a bi-directional Extended Capabilities Port and supports an enhanced protocol for data transfer that can cover a wide range of needs [5]. It can be addressed in a straightforward manner through globally addressable registers and can even be served by an interrupt request [6]. It presents the easiest way for the design of custom peripherals, since it does not require the use of a controller, like a UART or a USB peripheral controller. Moreover it is supported by driver software designed for all PC based operating systems, so it can be addressed from within most visual programming languages, like Visual Basic, Visual C or LabVIEW.

Inspite of the above flexibility, the parallel port has not been used often in order to support data acquisition devices, because of its legacy use as a dedicated printer port. However it might present an interesting alternative now that USB ports cover most peripheral interfacing needs.

Another aspect of data acquisition systems is measurement software. National Instruments (NI) provides LabVIEW, a software for the development of data acquisition applications and data processing, while Mathworks also provides a toolbox for data acquisition along with Matlab, their software for engineering mathematics and simulation. Both tools are powerful, while NI's software is better suited for easy interfacing and measurement processing, as it incorporates a large number of ready device drivers within a graphic interface. A number of other tools, based on visual programming languages, also present interesting alternatives.

Finally, microcontrollers play today an important role in all kinds of automated control systems and come equipped with many peripherals, like analog to digital converters, UARTS, PIOs, I^2C interfaces or even USB controllers. Although they are limited to executing code following the von-Neumann serial philosophy, they can run at clock speeds that support fast applications, like DSP. It is only inevitable that microcontrollers are used in data acquisition applications and computer interfacing as well.

In this paper we present the design of a data acquisition system that makes use of the peripheral capabilities of a PIC16F877 microcontroller driven by a clock frequency of 20MHz. In this system we integrate the microcontroller with the ECP parallel port and with LabVIEW software. As a result we produce a very low cost A/D computer interface that is capable to cover a wide range of applications in the field of digital measurements. The system can be enhanced by upgrading the microcontroller with a higher member of the Microchip family but performance is limited by the data exchange capabilities of the parallel port in ECP mode. Depending on the computer motherboard the system can provide a mean "free-running" sampling rate of 100KSps on a Pentium III and can support more than 150Ksps for faster systems. The analysis is limited to 10 bits by the PIC16F877 microcontroller A/D module. We also present a DAC module integrated with the microcontroller, able to produce waveforms in the acoustic region of frequencies.

The system serves as an example of simple architecture that makes use of up-to-date off the shelf software solutions and easy to use interfacing in order to achieve acceptable data rates at a fraction of the cost.

2. ARCHITECTURE OF THE ENHANCED PARALLEL PORT

The design that follows is based on the extended capabilities of the enhanced parallel port which is standard equipment for most motherboards of desktop PCs. The extended capabilities are activated through the BIOS settings of the computer, by selecting ECP and EPP1.9 from within the Integrated Peripherals menu. Fig. 1 presents the basic architecture and the memory mapped registers corresponding to each group of pins. Also, it shows the input or output use of each register. In addition to this layout one should also mention the ECR register (Extended Control Register) found in Base address + 402h. Base address in the above architecture usually is address 378h for LPT1 or can be retrieved for all LPTs by reading the BIOS addresses 0000:0408 through 0000:040E.

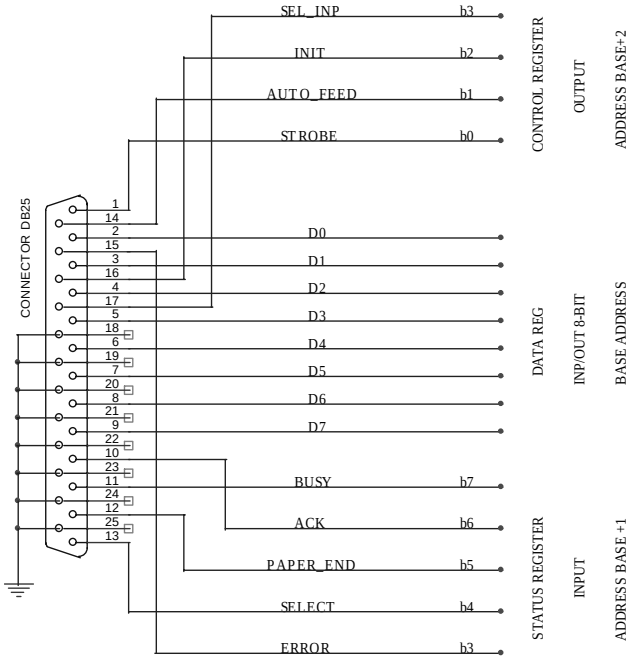


Fig. 1 – Architecture of the parallel port

In the DAQ interface presented in this paper the DATA register is used in bi-directional mode, for data input and output. As discussed in the next paragraph, this register reads the high byte of the A/D conversion as it is transmitted from the PIC port labeled D (PORTD). In reverse, DATA register outputs data that are read from PORTD in input mode and are loaded to programmable PIC registers. The STATUS register of the parallel port is used for input and receives the two least significant bits of the A/D conversion. Bit b3 of the STATUS register also receives handshaking signals from the microcontroller. The CONTROL register of the parallel port is used for output of control signals. These signals include handshaking to PIC using

STROBE pin, and mode select information that controls the execution flow of the PIC code.

When digital-to-analog mode is selected, DATA register is used to transfer a byte to a register in order to control DAC frequency. An alternating analog signal is then produced at the output of a XR2206 integrated circuit, as explained in the following paragraph.

3. HARDWARE DESIGN

Fig. 2 shows the basic hardware design of the A/D interface. Architectural details for this microcontroller can be found in Ref. 7. PORTA of the microcontroller is used mainly for analog data input (pins 2, 3, 7), but it can be configured into any combination of analog or digital I/O. The TOCKI input (pin 6) receives handshaking digital input from the STROBE bit of the parallel port. PORTD serves as the main channel of communication for data exchange with the computer and is connected to the bi-directional DATA register of the parallel port. As mentioned, this channel transmits the high byte of the left-justified A/D conversion result or receives control data from the computer and adjusts the operating mode of the microcontroller by writing to the appropriate PIC registers. PORTB plays a multiple role. Bits RB6 and RB7 transmit the two least significant bits of the 10-bit result of the A/D conversion to the computer. These bits are read by the STATUS register of the parallel port (b4 and b5), which is used for input to the PC. Bits RB1 and RB2 of PORTB receive data from bits b2 and b3 of the CONTROL register of the parallel port. As mentioned, these signals are used for assembly code flow control. Accurate transfer of data from the microcontroller to the computer and vice versa is attained by applying a simple custom handshaking protocol. When the microcontroller has a data set ready for transmission it sets RB3 of PORTB which is polled by b3 of the STATUS register of the computer parallel port. When the computer asks to transfer data to the microcontroller it sends an active low STROBE signal to the TOCKI bit (RA4 or pin 6) of the microcontroller. When expecting data the microcontroller is put in a waiting state and polls the TOCKI bit. Generally, during a bulk transfer of data between the two devices, the one is always polling for a handshake signal and receives data upon exiting the polling state, while the other prepares and transmits the data set and pulls the handshake line in order to end the procedure.

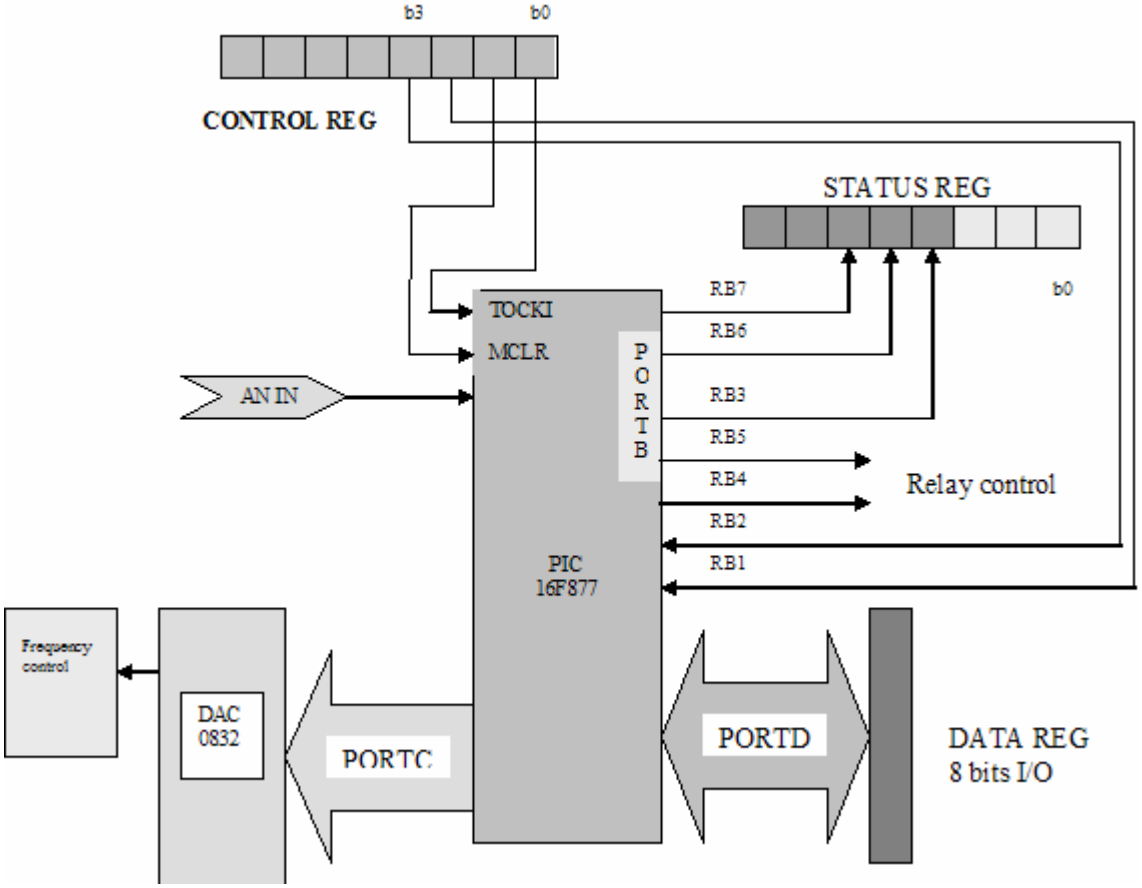


Fig. 2 - Basic hardware design

Fig. 3 shows the design of the digital-to-analog (DAC) module. It is based on a XR2206 monolithic waveform generator, which is capable to produce high accuracy sine, pulse, sawtooth or triangular signals. The frequency range of the circuit can be selected externally, by adjusting the value of an external capacitor connected between pins 5 and 6, while a micrometric variation of the output frequency can be achieved by varying the voltage applied at pin 7. This voltage is produced by a digital to analog DAC0832 R-2R circuit driven by PORTC of the microcontroller.

The frequency control byte is transmitted through the parallel port and is read from PORTD of the microcontroller when appropriate mode is selected through CONTROL register bits b2 and b3 (pins 16 and 17 of the parallel port connector - see Fig. 2). Upon DAC mode selection (see Table 1) an appropriate PIC subroutine is executed and the frequency control byte is read and latched on PORTC.

Capacitor values are chosen through relays (relay1 and 2 in Fig. 3) driven by digital output RB4, RB5 of the microcontroller (pins 37, 38).

RB4 and RB5 combination is produced with respect to mode select values of input bits RB1 and RB2 of PORTB (see Fig. 2). Table 1 shows mode selection data as transmitted by CONTROL register and read by RB1 and RB2.

Table 1. Function mode selection

RB1	RB2	Function	Capacitor selection, driven by RB4, RB5
0	0	ADC	irrelevant
1	0	DAC	100nF (default)
0	1	DAC	10nF
1	1	DAC	1nF

4. DESIGN OF SOFTWARE INTERFACE

The software needed for this application is divided in code written for the PIC microcontroller and code written for the host computer in the form of user interface and device drivers.

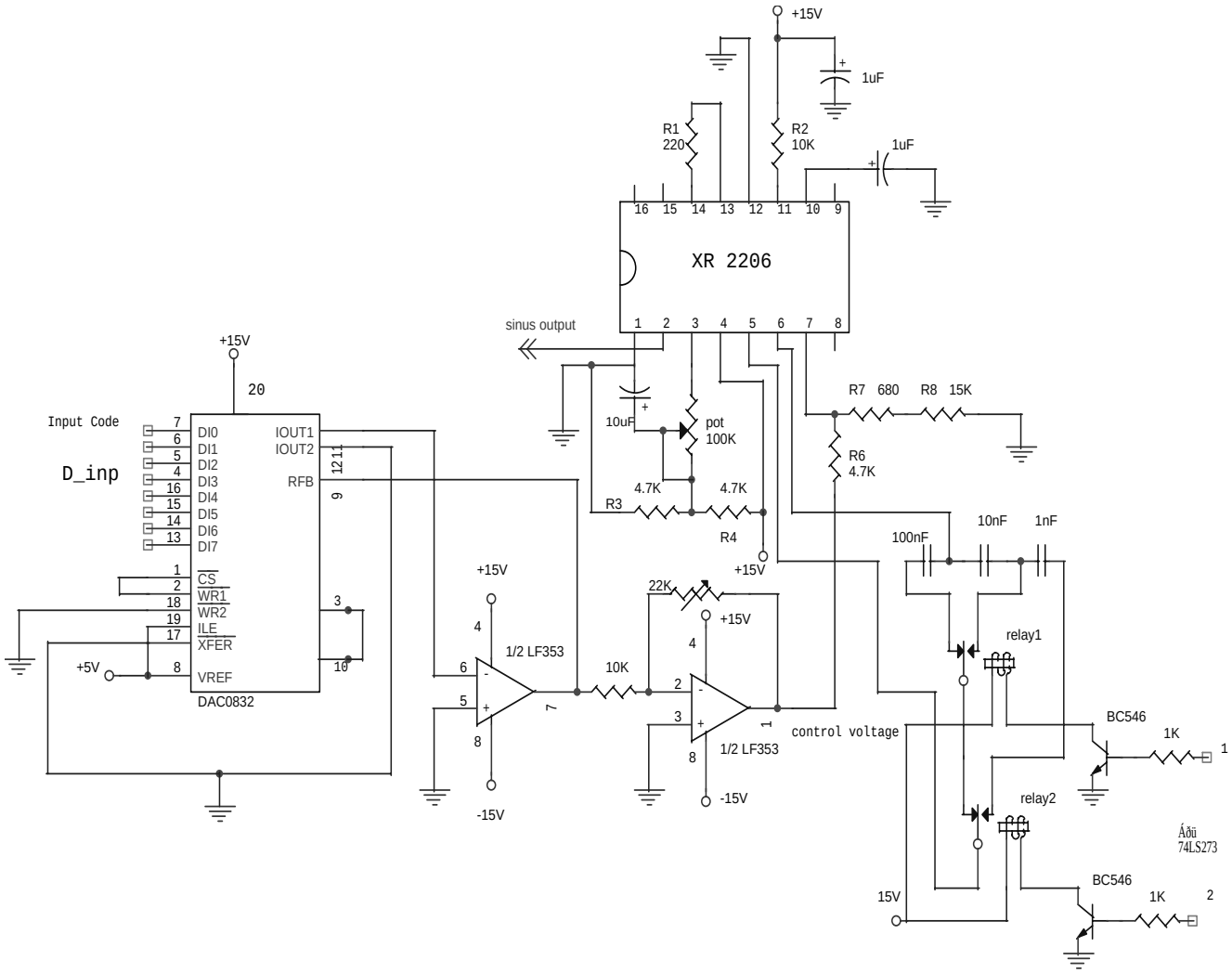


Fig. 3 – DAC module integrated with the microcontroller and the ECP parallel port

Device drivers and user interface are produced using LabVIEW 7.1 development system software, copyrighted by National Instruments Corporation. We used a modular approach and designed a number of hierarchical blocks each one controlling a particular function [9]. In this way one can produce any sequential procedure of the basic functions by simply rearranging the given blocks in due order and adding appropriate processing functions, time delays, graphs and controls wherever needed.

The basic flow chart for device initialization and data acquisition is as follows:

1. We select ADC or DAC mode.
2. We read ECR register found in address "Base+402h" and set b5 in order to select Byte mode for the EPP parallel port (see Ref. 8).
3. We read CONTROL register of the parallel port in location 37Ah and we clear b5 in order to set DATA register of the bi-directional parallel port to output mode.
4. We program ADCON0 register of the PIC16F877 microcontroller by transmitting the byte "01000001" through the DATA register of the parallel port. This code turns on the A/D module of the microcontroller and sets the conversion speed to its maximum value (in this way we actually drive the PIC by overclocking, but it works fine).
5. We reset the microcontroller by sending a zero bit to MCLR. For this purpose we pull down AUTOFEED (b1 of the CONTROL register), as shown in Fig. 3.
6. We pull down SROBE bit of the parallel port CONTROL register, initiating handshaking. The microcontroller reads PORTD and writes ADCON0.
7. We set b5 of CONTROL register in order to set DATA register of the parallel port into input mode.

There follows the flow chart for bulk data acquisition. In this subroutine a large set of sampled data is transferred to the computer following A/D conversions. The following code is included into a FOR loop:

1. We set STROBE bit.
2. We poll b3 of microcontroller PORTB in a WHILE loop, waiting for the end of A/D conversion.
3. Upon finishing the conversion the microcontroller places the high byte of the 10 bit conversion on PORTD. The two lower bits are placed on RB6 and RB7 of PORTB. The microcontroller code sets b3 of PORTB and is put into a waiting state, polling the TOCKI bit.
4. When the WHILE loop ends, the high byte of the A/D conversion is read through DATA register, while the two lower bits are read through STATUS register (see also paragraph 3).
5. When the host computer finishes reading the conversion bytes it pulls down the STROBE bit of the CONTROL register.
6. The microcontroller exits the waiting state and starts a new conversion.
7. The host computer executes next turn of the FOR loop. It waits in the WHILE loop polling RB3 (b3 of PORTB).
8. The procedure goes on until all scheduled conversions are transferred.

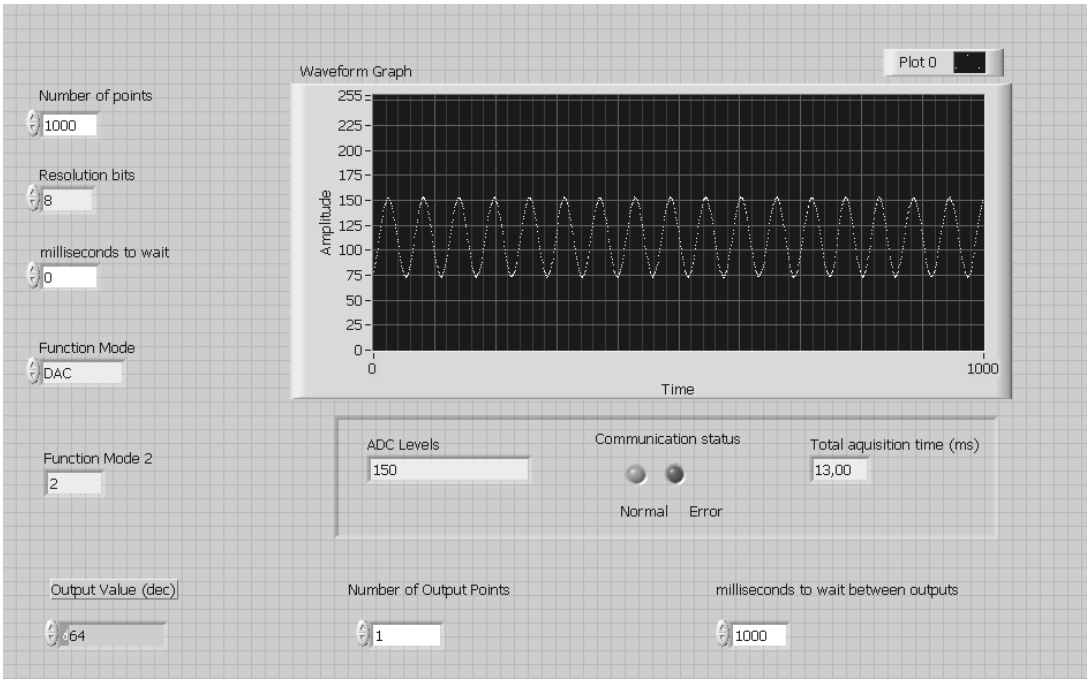


Fig. 4 – User interface made with LabVIEW software

From the microcontroller side, we first initialize the ports and set them appropriately for input or for output. Bits b0, b1, b2 of PORTB are set as inputs, while b3, b4, b5, b6, b7 are set as outputs.

We then read bits RB1 and RB2 and set the function mode accordingly. We either execute a "acquisition" subroutine, where we setup a conversion and output data according to step 3 of the above algorithm or execute a "analog_out" subroutine where the microcontroller receives a frequency control byte and outputs data according to paragraph 3. We switch the function of PORTD to input or output according to I/O needs. Register ADCON0 is configured with data transmitted from the host computer and so can be configured ADCON1 if we need to have control over channel selection.

Fig. 4 shows the front panel of a virtual instrument designed with LabVIEW software for data acquisition. The corresponding block diagram with graphic code is shown in Fig. 5. In the same manner one can add to this modular design by adding hierarchical blocks for particular needs. Such a simple and straightforward programming approach is also suitable for the classroom.

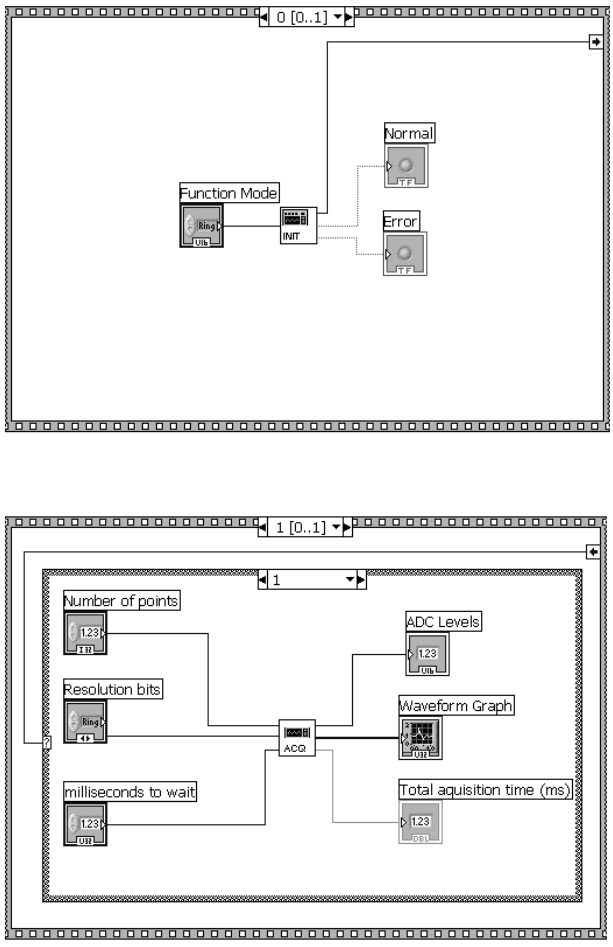


Fig. 5 - Sequential block diagram constructed with hierarchical blocks

5. CONCLUSIONS

In this article a very compact and low-cost DAQ architecture was presented in the form of an external device interfacing with the computer by means of the ECP parallel port. This system is an example of an effective integration between microcontroller, computer interface and software design. The microcontroller runs at 20MHz and has a command execution cycle of 250 ns. By allowing overclocking of the A/D module of the microcontroller we attain a 10-bit conversion time of approximately 8μs. Sampling frequency was measured to be up to 150 KHz on a Pentium IV platform and it maintains a mean "free-running" value of about 100KHz on slower systems. Free-running frequency is defined as the full speed frequency that is allowed by the custom protocol described in paragraph 3 of this article. Sampling frequency can be controlled by executing delay subroutines between samples.

User interface and system drivers were designed using LabVIEW software that allows addressing the registers of the enhanced parallel port from within Windows XP OS. Graphical programming in LabVIEW can easily be condensed into hierarchical blocks that only need to be connected with appropriate input controls and output graphs or indicators. In this way we construct a very flexible platform for configuring the device and programming input and output procedures. This programming tool can also be used for educational purposes where a low cost solution is often appreciated.

By integrating a fast digital to analog module into the above architecture we allow for flexibility in automated measurements with alternating signals. The DAC module is based on XR2206 waveform generator and can be adjusted with external capacitors to produce a maximum frequency of 50KHz, keeping excitation signals within the limits of the Niquist theorem. A typical frequency response can thus be measured automatically.

A number of variations of the above architecture are possible. A more powerful microcontroller with higher RAM capacity can be used to log samples into RAM registers before transmitting them to the computer. A USB interface to the computer can be supported by controllers like PIC16C745/6 (Ref. 10, 11), but a handler to the USB interface is not widely available and has to be designed from scratch. Visual languages with access to the Windows API interface can offer some solution (Ref. 12), however graphic languages like LabVIEW are short of ready drivers for USB interfacing.

6. REFERENCES

[1] Tran Tien Lang, *Electronique des systemes des measures*, Masson, 1997.

[2] Kevin James, *PC Interfacing and Data Acquisition*, Newnes, 2000.

[3] http://www.ni.com/dataacquisition/

[4] http://www.ni.com/dataacquisition/usb/

[5] Jan Axelson, *Parallel Port Complete*, Lakeview Research, Madison, 2000.

[6] Paul Bergsman, Controlling the world with your PC, Hightext Publications, 1994.

[7] Microchip PIC16F87x Data Sheets, Microchip Technology Inc. 1999.

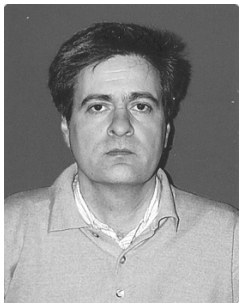
[8] Craig Peacock, *Interfacing the Standard parallel port*, in: http://www.beyondlogic.org/spp/parallel.htm, 2000.

[9] See for example: John Essick, Advanced LabVIEW Labs, Prentice Hall, 1999.

[10] Microchip PIC16C745/6 Data Sheets, Microchip Technology Inc. 2001.

[11] John Hyde, USB design by example, Intel, Wiley, 1999.

[12] Jan Axelson, USB Complete, Lakeview Reasearch, 1999.



John A. Kalomiros *received the degree of Physics at the Aristotle University of Thessaloniki in 1984 and a Masters degree in Electronics in 1987. His doctoral thesis is on characterization of materials for microelectronic devices. His recent research interests include microcontrollers and digital systems design with applications in Robotics. He is also working on systems for digital measurements and instrumentation. He teaches electronics and related subjects at the Technical and Educational Institute of Serres, Greece.*