Simplifying the Communication with I²C Devices Using LabVIEW and the PC's Parallel Port

E. Lunca[1], C. Damian[1] and F. Mariut[1]
[1]"Gheorghe Asachi" Technical University of Iasi, Faculty of Electrical Engineering, Iasi, Romania

***Abstract*—At the moment, Inter-Integrated Circuit (I²C) is one of the most widely used protocols for serial data communication in embedded systems. The purpose of this paper is to show a simple solution to communicate with I²C devices using the PC's parallel port and LabVIEW graphical programming environment. First, a low-cost 2-wire interface and an associated I²C LabVIEW library are presented. Then, a number of sample applications involving different ICs are described.**

***Index Terms*—I²C interface, LabVIEW library, parallel port adapter, sample applications.**

I. Introduction

The I²C bus is a two-wire bidirectional bus used for low speed, short-distance communication between integrated circuits (ICs). Invented by Philips in the early 1980s [1], it is implemented today in a large variety of ICs, including digital sensors, microcontrollers, real-time clocks, analog-to-digital converters, EEPROM memories, digital potentiometers, etc.

Starting from the necessity to quickly test and configure such ICs, our efforts have been directed to create a simple and yet efficient solution for controlling I²C devices directly from the computer. Because the PC's parallel port (LPT) provides the easiest way to perform direct digital I/O operations with minimal external circuitry [2]-[5], we have developed an LPT-to-I²C adapter based on the 74LS05 hex logic inverter, along with a few passive components. The associated software has been implemented as a set of virtual instruments (a library) for LabVIEW, since this graphical programming environment greatly reduces the development time and offers access to powerful features for data acquisition, processing and presentation. Therefore, being created the I²C LabVIEW library, all efforts can be applied solely to design user applications.

II. Hardware

The schematic diagram of the LPT-to-I²C adapter is shown in Fig. 1. Although there are many interfacing solutions for I²C, the one presented here does not require a microcontroller and firmware. In fact, the only required IC is a 74LS05, which contains six independent gates with open-collector outputs. It is used for converting the TTL signals from the parallel port to I²C signals (the I²C protocol involves two open-drain or open-collector bidirectional lines, SDA and SCL, which need pull-up resistors to achieve high-logic levels), as well as for ensuring some isolation between the parallel port and the downstream circuitry.

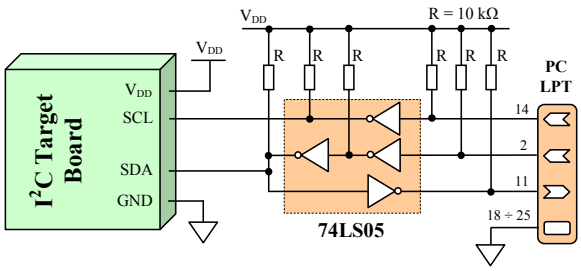

Figure 1. Schematic of the LPT-to-I²C adapter

Three open-collector outputs are used for connecting the bidirectional data signal (SDA) to one dedicated input (pin 11) and one dedicated output (pin 2) of the parallel port, in opposition to the case of using one bidirectional pin of the LPT, which will potentially require to experiment with the PC's BIOS utility [6]. Similarly, an open-collector output is necessary for connecting the SCL signal – generated at the pin 14 of the parallel port – to the I²C device (the clock stretching is not implemented here). Because the signals at the pins 11 and 14 are re-inverted by the LPT through its internal circuitry, we don't have to invert bits in software when transmitting and receiving data over the port. Since only four of the six 74LS05's inverters are required to implement the I²C communication, two inverters remain unused and their inputs are tied to the ground to prevent floating.

A picture of the developed LPT-to-I²C adapter is presented in Fig. 2. It connects to the I²C target system via one of the two 4-pin headers on the board, with the power (+5 V_{DC}) being supplied directly from the target. The LED will indicate a proper power connection. Note that the adapter can only be operated as "single master", at a maximum transfer speed highly dependent on the host PC. Generally, there is no need to operate modifications of the parallel port settings from BIOS.

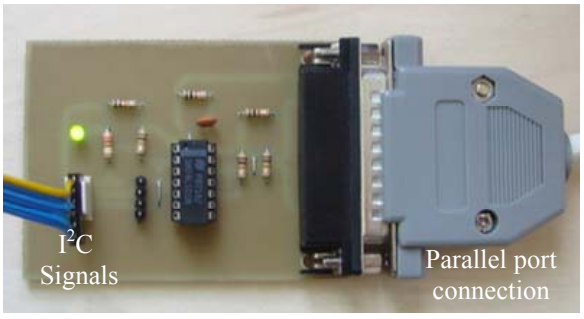

Figure 2. A picture of the LPT-to-I²C adapter

978-1-4673-2542-4/12/$31.00 ©2012 IEEE

III. SOFTWARE

The standard parallel port uses three 8-bit registers, commonly called the Data, Status and Control ports. As illustrated in Table 1, for implementing the I²C communication, we use one bit in each register. The Data register, located at the base address (typically, 378h), contains the bit D0, which is responsible for controlling the pin 2 and thus the SDA input data for the addressed I²C device. Complementary, the SDA output data are obtained by reading the bit 7 (Busy) of the Status register (located at the base address+1), which is responsible for controlling the pin 11 of the LPT. Finally, the bit 1 (Auto Linefeed) of the Control register, at the location base address+2, is responsible for controlling the pin 14 and thus the SCL signal.

In LabVIEW, as in most programming languages, the hardware registers of the pararllel port can easily be accessed using the *In Port* and *Out Port* functions existing in the *Connectivity – Port I/O palette*. These low-level VIs allow reading and writing 8, 16 or 32 bits of data from or to any 16-bit I/O port address and they have been used for creating a LabVIEW library of higher-level VIs that conform to the main events on the I²C bus. As presented in Fig. 3, the developed I²C library contains the following functions:

- *I²C Start.vi* is used to generate the start condition, which is necessary prior to any transaction on the bus. This VI defaults to the LPT address of 378h, so you can let the VI's input unwired. However, if you determine that the software is unable to communicate with the I²C device, then another address should be specified at the LPT Port input (either 278h or 3BCh). The reference output of the *I²C Start.vi* can be passed to other VIs in the I²C library;
- *I²C Send.vi* serves for transmitting messages to the I²C slave device. Typically, the input of this function is an 8-bit array, so it can be used, for instance, to send a 7-bit address followed by the read/write bit, which is added into the array as the LSB. However, the function input can be resized to transmit even a single bit, such as an acknowledge (ACK) or not-acknowledge (NACK);
- *I²C Read ACK.vi* is used for reading the acknowledge bit issued by the addressed I²C device;
- *I²C Read 1 Byte.vi* reads a byte of serial data from the addressed I²Cdevice, MSB first. After receiving the data byte, this VI automatically sends either ACK or NACK, depending on the value written at the correspondent input;
- *I²C Read 1 Byte (without sending ACK NACK).vi* reads a byte of serial data from the addressed I²C device, MSB first, but it doesn't send ACK or NACK. When calling this function, its output must be passed to the *I²C Send* VI, which will be used for sending either an acknowledge or a not-acknowledge bit;
- *I²C Read 2 Bytes.vi* reads two bytes of serial data from the addressed I²C device. The function automatically sends ACK and NACK after receiving the MSB data byte and LSB data byte, respectively;
- *I²C Stop.vi* is used to generate the stop condition, which is necessary after each data transfer.

TABLE I.
PARALLEL PORT REGISTERS

Address		MSB							LSB
	Bit:	7	6	5	4	3	2	1	0
Base (Data port)	Pin:	9	8	7	6	5	4	3	2
Base+1 (Status port)	Pin:	~11	10	12	13	15			
Base+2 (Control port)	Pin:					~17	16	~14	~1

~ indicates a hardware inversion of the bit

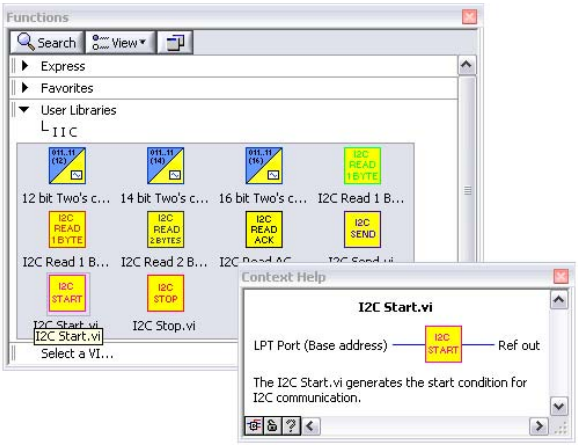

Figure 3. I²C Library for LabVIEW

Other VIs can be developed and added into the I²C library in order to satisfy specific requirements. For instance, the first three VIs in Fig. 3 have been created for converting 12-, 14- and 16-bit 2's complement codes to decimal values (e.g., in the case of an analog-to-digital converter or digital temperature sensor). Because these VIs are not dependent on the hardware, they will work with any LabVIEW program involving such binary manipulation.

IV. SAMPLE APPLICATIONS

Fig. 4 shows the I²C communication scheme [7]. The master initially sends a start bit, followed by the 7-bit address of the I²C slave device, which is finally followed by a single bit representing whether it wishes to write (0) or to read (1) from the IC. If the addressed slave exists on the bus, it will respond with an ACK bit (active low for acknowledged). The master then continues in either transmitting or receiving mode, while the slave continues in its complementary mode. Every data byte put on the SDA line must be 8-bit long.

To implement such communication schemes, the developed I²C VIs can be called in sequence. All you have to do is appending these functions according to the desired message format – single read, single write or combined messaging – and configuring them with proper information, as specified in the IC's data sheet. In this way, it is possible to quickly create graphical user interfaces (GUIs) that allow testing and validating various I²C-based designs, as illustrating in the following examples.

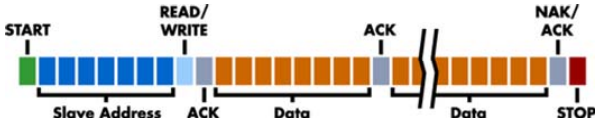

Figure 4. The I²C communication scheme

A. Using I²C digital potentiometers to implement programmable analog functions

The digital potentiometers, sometimes called digital POTs, RDACs or digiPOTs, are mixed signal devices intended as electronic replacements for mechanical potentiometers. Because they are digitally controlled, very often trough an I²C or SPI interface, the digiPOTs can be used, among others, to vary the key parameters of the active circuits and thus to implement programmable analog functions. For instance, as illustrated in Fig. 5, digiPOTs can be combined with operational amplifiers to build programmable active filters. Here, an AD5254 quad digital potentiometer is used for implementing a programmable 4th order high-pass filter, capable of generating popular Butterworth, Bessel and Chebyshev responses over a certain corner frequency range [8].

Fig. 6 presents the front panel and block diagram of the LabVIEW application created for configuring the digiPOT during the filter evaluation. The functionality of this program is limited to writing the content of the four RDAC registers and reading the four resistor tolerances from the nonvolatile memory, but it can be easily adapted to support all other operations specified for the AD5254, such as storing RDAC settings and user-defined data in the memory, quick write commands, etc. Setting the content of a RDAC register requires only pressing the associated radio button and typing the desired value of the R_{WB} resistance. The program automatically determines the closest decimal code and writes the equivalent data byte in the selected RDAC register. Both the decimal code and the predicted R_{WB} are displayed for the user. If there are problems in addressing the slave device or in receiving data from it, an error message will be displayed.

B. PC-based data logging using a 12-bit Data Acquisition System IC with 2-wire serial interface

A slightly different design is shown in Fig. 7. Here, the proposed LPT-to-I²C adapter is tightly integrated with the MAX128, a 12-bit DAS with 2-wire serial interface from Maxim, to provide a simple and cost-effective solution for acquiring and recording data from sensors and other analog outputs. Because the MAX128 is operated from the internal +4.096 voltage reference, each analog input channel can be independently programmed to one of four ranges: 0 to +4.096 V, 0 to +2.048 V, ±4.096 V and ±2.048 V, respectively. Note that the MAX128 provides eight analog inputs, but only four have been used in this application.

The basic software associated with this module is presented in Fig. 8. It allows configuring the control byte of the MAX128 through the "Channel" and "Range" controls and calculates the analog voltage at the selected input by reading the MSB and LSB data bytes from the MAX128. For bipolar input modes, it uses one of the three conversion VIs in the I²C library to convert the 12-bit 2's complement output code of the MAX128 to a decimal value. As for the unipolar input modes, the measured analog voltage is then obtained by multiplying the result by the correspondent LSB value (e.g., 0.002 mV for the range ±4.096 V).

Depending on the application to be developed, it is convenient to encapsulate this program as a sub-VI. Such an example is shown in Fig. 9, which gives the block diagram of a VI that continuously acquires the four analog

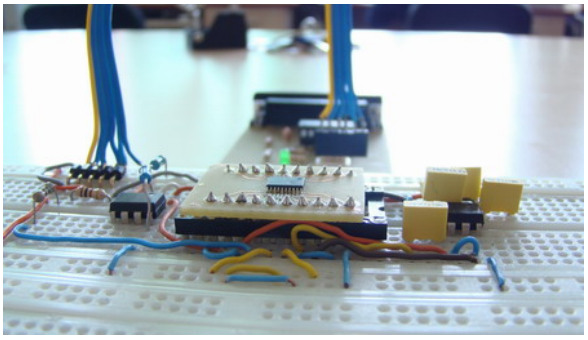

Figure 5. Prototype of a programmable high-pass filter based on the AD5254 digiPOT

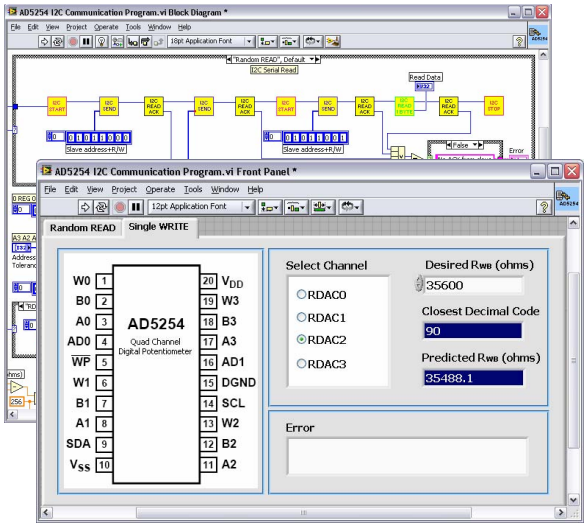

Figure 6. Simple LabVIEW program for configuring the resistor values of the AD5254 digiPOT

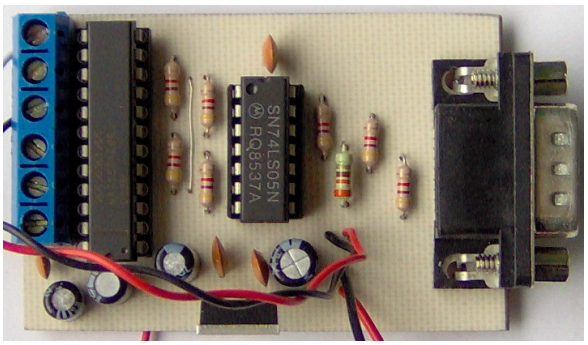

Figure 7. A simple data logging module based on the MAX128 12-bit DAS, for the PC's parallel port

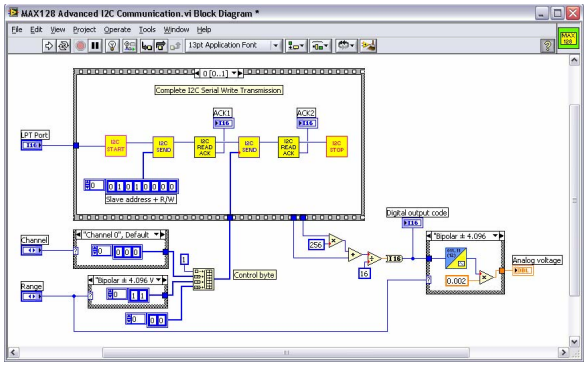

Figure 8. LabVIEW program for communicating with the MAX128 DAS via I²C protocol

inputs and displays four digital readouts on its front panel. Any further LabVIEW manipulation of the acquired data is rather trivial.

C. *Temperature monitoring using a TMP75 digital sensor with 2-wire serial interface*

There is a wide variety of temperature sensor ICs that combine analog sensing circuitry with digital I/O and control registers, to provide standard connectivity solutions for microcontroller-based systems. Since many of them are I^{2}C-bus compatible devices, our interfacing approach allows performing PC-based temperature acquisition with virtually no effort.

A LabVIEW program for temperature monitoring over the parallel port is shown in Fig. 10. It is based on the TMP75 digital sensor with 2-wire interface from Texas Instruments, which requires no external components and is capable of measuring temperatures with a resolution of 12 bits (0.0625 °C). In this application, a single sensor has been used (all three address pins, A2, A1 and A0, have been grounded), but the number of sensors connected to the bus can be extended up to eight.

The VI's operation is straightforward. First, 9, 10, 11 or 12 bits of resolution can be obtained through the "Select resolution" control (by addressing the Configuration Register of the TMP75 and setting the resolution bits accordingly). Then, the 2's complement temperature data are continuously read from the Temperature Register, converted to Celsius degrees and displayed in both digital and graph format. Additional data processing is used for determining the max and min values from the start time, as well as for recording the measurement results. The period between records is user-programmable, but the displayed temperatures can be saved at any moment.

V. Conclusions

Using LabVIEW and the PC's parallel port, it is possible to quickly communicate with I^{2}C devices. The proposed LPT-to-I^{2}C interface and the associated I^{2}C LabVIEW library allow developing intuitive GUIs, providing programmability and control in a large variety of applications.

Acknowledgment

This paper was supported by the project PERFORM-ERA "Postdoctoral Performance for Integration in the European Research Area" (ID-57649), financed by the European Social Fund and the Romanian Government.

References

[1] Philips Semiconductor, *The I^{2}C-Bus Specification*, Version 2.1, January 2000.

[2] E. Lunca, A. Salceanu, M. Cretu, "Implementing the I^{2}C Communication Protocol in LabVIEW", 15th IMEKO TC4 International Symposium on Novelties in Electrical Measurements and Instrumentation, Iasi, Romania, 2007, pp. 514–517.

[3] J. Axelson, "Parallel Port Complete: Programming, Interfacing, and Using the PC's Parallel Printer Port", Lakeview Research, 2000.

[4] P.H. Anderson, "Use of a PC Printer Port for Control and Data Acquisition", *Technology Interface Journal*, Fall 1996 [Online: http://technologyinterface.nmsu.edu/fall96/].

[5] D.V. Gadre, "Programming the Parallel Port: Interfacing the PC for Data Acquisition and Process Control", CMP Books, 1998.

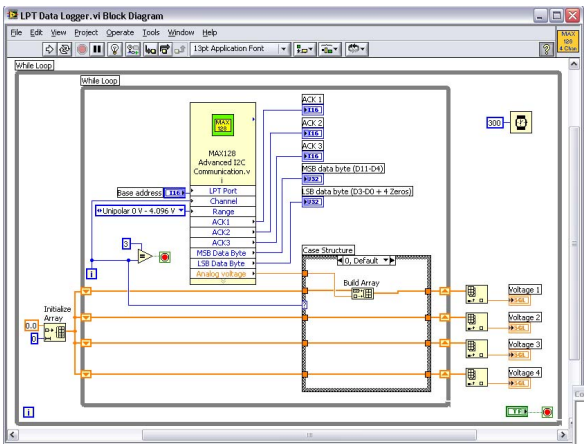

Figure 9. LabVIEW program for continuously reading the first four channels of a MAX128 DAS

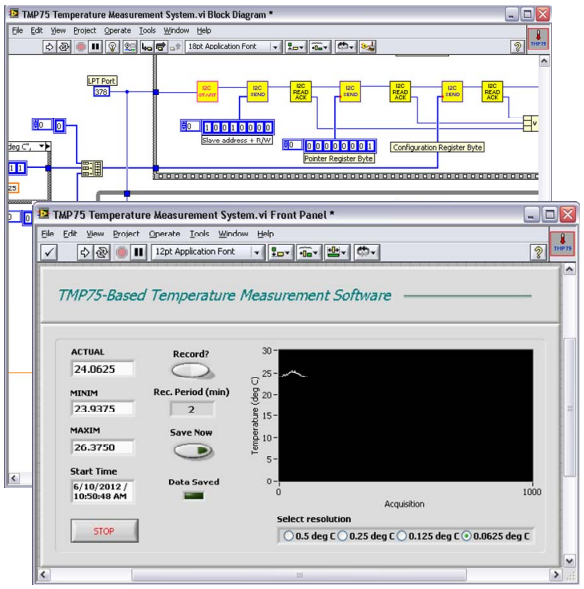

Figure 10. Temperature acquisition software based on the TMP75 digital sensor with 2-wire serial interface

[6] B. Vasquez, "How to Use a PC's Parallel Port to Communicate with 2-Wire Devices", Application Note 3230, Maxim Integrated Products, June 15, 2004.

[7] www.totalphase.com/docs/aardvark_datasheet/sect001/#s1.1.

[8] E. Lunca, C. Damian, D. Petrisor, "Programmable Active Filters Based on Digital Potentiometers", unpublished.

Authors

E. Lunca is with the "Gheorghe Asachi" Technical University of Iasi, Faculty of Electrical Engineering, 21-23 Professor Dimitrie Mangeron Street, 700050 Iasi, Romania (e-mail: elunca@ee.tuiasi.ro).

C. Damian is with the "Gheorghe Asachi" Technical University of Iasi, Faculty of Electrical Engineering, 21-23 Professor Dimitrie Mangeron Street, 700050 Iasi, Romania (e-mail: cdamian@ee.tuiasi.ro).

F. Mariut is with the "Gheorghe Asachi" Technical University of Iasi, Faculty of Electrical Engineering, 21-23 Professor Dimitrie Mangeron Street, 700050 Iasi, Romania (e-mail: fmariut@ee.tuiasi.ro).