DEVELOPMENT OF USB 2.0 COMPLIANT GPIB CONTROLLER

Boštjan Glažar, Marko Jankovec, Marko Topič

University of Ljubljana, Faculty of Electrical Engineering, Ljubljana, Slovenia

Key words: Automated measurement, Computer interfaces, General Purpose Interface Bus

Abstract: This paper describes a development of a custom designed General Purpose Interface Bus (GPIB) controller which is used as the interface between GPIB and PC. The controller is designed as a Universal Serial Bus (USB) 2.0 compliant external device, which provides plug-and-play operation, high speed of data transfer and is powered fully from the USB. In contrast to conventional GPIB controllers in the form of PC cards, such a design extends its usage to notebooks or other computers with no available I/O slots. The FLASH program memory based AT90S8515 microcontroller which is used for data transfer and protocol handling also enables easy firmware upgrade. For simple controller usage the appropriate driver is developed in a graphical programming language LabVIEW, which we use for instrument control software development.

Razvoj GPIB krmilnika združljivega z USB 2.0

Kjučne besede: Avtomatizirane meritve, računalniški vmesniki, GPIB

Izvleček: Članek opisuje razvoj krmilnika za GPIB vodilo (General Purpose Interface Bus), ki se uporablja kot vmesnik med GPIB vodilom in osebnim računalnikom. Krmilnik se na računalnik priključi preko univerzalnega serijskega vodila (USB), ki omogoča enostavno uporabo, visoko hitrost prenosa podatkov in nudi napajanje. Za razliko od običajnih GPIB krmilnikov, ki so osnovani kot vtične kartice, je ta krmilnik primeren tudi za prenosnike in računalnike brez prostih razširitvenih rež. Jedro krmilnika je mikrokrmilnik AT90S8515, ki je vsebuje FLASH spomin in tako omogoča enostavno nadgradnjo programske opreme. Za enostavno uporabo krmilnika smo izdelali tudi gonilnik za programski jezik LabVIEW, ki ga tudi uporabljamo za avtomatizacijo meritev.

1. Introduction

Personal computers became necessary equipment in science laboratories in the last decade /1/. One of reasons is their usage in measurement automation and documentation which is achieved with computer-controlled instruments. A very popular interface for connection between the PC and instruments is General Purpose Interface Bus (GPIB), which is still the most common interface, although several other types of interfaces (Universal Serial Bus (USB), IEEE 1394, Ethernet, etc.) are on the move /2/, /3/. Reasons for this include numerous GPIB instruments available and low latency when compared to otherwise faster standards in parentheses above /4/.

To connect instruments to a PC a suitable GPIB controller is required which is usually in a form of a computer card. Because of relatively simple design and low-cost components we have decided to develop a GPIB controller ourselves. Since we wanted the controller to be applicable also for notebooks, we have chosen USB, which is the most common built-in interface on contemporary computers. Our concept was accepted as Design idea in EDN Journal /5/. In the following, the development of the USB based GPIB controller will be presented in detail and an example of its usage will be given.

2. Development of USB based GPIB controller

GPIB /6/, initially named Hewlett-Packard Interface Bus (HP-IB), later standardized as IEEE-488 and IEC-625, is a parallel bus that uses 24-pin connector to connect devices in a star or a bus configuration. There are three main types of devices on the GPIB bus; controller, talker and one or more listeners. The bus can have only one active controller, which is a bus master and addresses devices to talk or listen by use of GPIB bus commands. A talker transmits data that are received by one or more listeners.

The GPIB controller we developed is designed as a USB device controlled by a host PC, where one port of the controller is connected to the USB and other to a GPIB bus. The controller is built on a double-sided PCB that fits into a box of outer dimensions of 123 mm x 30 mm x 70 mm.

In order to simplify the usage of the GPIB controller a LabVIEW driver was developed, which communicates with the controller by a custom developed protocol through USB using a virtual COM port (VCP) driver. To simplify adaptation of existing programs in LabVIEW, the driver is compatible to the built-in one. In the following sections the hardware, protocol, firmware and the driver are described in detail.

3. Hardware

The controller can be divided into three main parts, where the microcontroller as the central block represents the communication gateway between PC (through USB interface) and GPIB bus (through GPIB line drivers), as shown in Fig. 1.

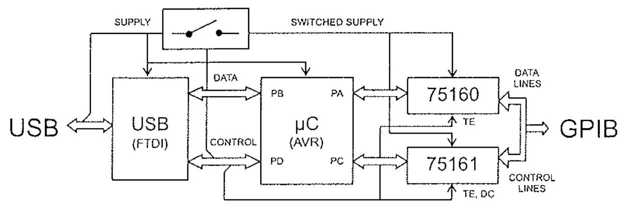

Fig. 1: Block diagram of the USB based GPIB controller

USB interface is based on a FTDI FT245BM integrated circuit /7/ that provides an 8-bit parallel interface to the microcontroller and a virtual COM port on the PC side. Built-in buffer (FIFO) greatly simplifies firmware in the microcontroller. The integrated circuit provides USB 2.0 compatibility and allows data transfer under USB 1.1 specification, i.e. 12 Mb/s. It also provides an output line that goes low when it is enumerated and USB is not in suspend state. External memory chip provides storage for information such as product description and maximum current consumption and can be programmed directly from a host PC.

At GPIB side of the controller appropriate line drivers are required to meet GPIB standard specifications of 48 mA sink current and high impedance state when no power is applied. Standard integrated circuits, 75160 /8/ and 75161 /9/ developed especially for GPIB are used, while sequence control is completely implemented in controller's firmware. The most important control signals for operation for data transfer are Attention (ATN), Data Valid (DAV), Not Ready For Data (NRFD), Not Data Accepted (NDAC) and End Or Identity (EOI). Direction of signals is controlled by TE (Talk Enable) input for data and handshake lines and by DC (Direction Control) for management lines. Both signals are controlled by the firmware.

An AVR microcontroller Atmel AT90S8515 /10/ is used to control all circuitry. It is a 44 pin RISC device with in-circuit programmable FLASH memory and many peripheral units, which are not all utilized in our case, since besides input / output (I/O) ports the only peripheral unit used is timer. In-circuit programming through a 6-pin connector simplifies firmware design and its upgrades. The microcontroller runs at its maximum frequency of 8 MHz to allow the fastest data transfer possible. Both external interrupts are used in conjunction with the USB interface. One triggers when USB goes to suspend state, while the other wakes up the microcontroller on the first received byte.

USB provides power supply of +5 V / 500 mA and thus allows low power devices to be used without additional power supply. In our design USB interface with memory IC and microcontroller are powered directly from USB, while GPIB buffers are powered from USB through a transistor switch. Overall maximum current consumption of the controller is 320 mA during normal operation and is minimized during USB suspend state, by disconnecting the GPIB buffers from the power supply and setting the microcontroller to low-power mode.

Fig. 2 shows the PCB of the controller, while Fig. 3 shows the controller with connected USB and GPIB cables. The PCB was made using milling machine and partially assembled on the manual SMD placer that we use for research & development prototypes and educational purposes. Predominant use of SMD components in the controller makes its fabrication easily automated and lowers the cost of components.

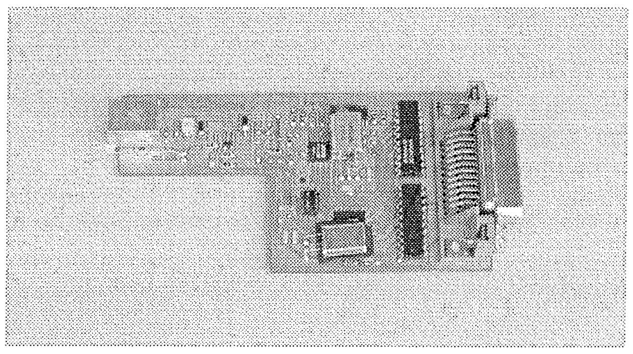

Fig. 2: Top view of printed circuit board of the controller

Fig. 3: The controller with connected cables

4. Protocol

The controller communicates with the PC through a logical serial interface which does not provide control signals and registers like plug-in boards and therefore a certain protocol is necessary to distinguish between data and control bytes.

We have developed a protocol, which is based on the AT protocol for modems; commands begin with letters IB, which are followed by a one character command code and an optional binary or character parameter. Bus commands are sent as binary values, which enables each of 256 possible values to be transferred. The problem of transferring binary data or a problem of data transparency was solved by a protocol similar to Binary Synchronous Communication (BSC) protocol /11/ in which three control characters are used to transfer a binary data block. These are Data Link Escape (DLE), Start Of Text (STX) and End Of Text (ETX). A data block always begins with the pair DLE/STX and ends with DLE/ETX. If a DLE is to be transferred anywhere in the data stream, it is followed by another DLE, denoting it as a part of the data as can be seen in Fig. 4. Bus addressing when sending or receiving data is not implemented in firmware and has to be done by sending bus commands, which is implemented in the driver. The controller commands are listed in Table 1. All responses of the controller are one byte long to allow simple interpretation and include success code or error specific code.

DLE	STX	*data*	DLE	DLE	*data*	DLE	ETX

Fig 4: BSC protocol

5. Firmware

The AVR microcontroller is used to control all lines of the GPIB interface and to communicate with the PC through the USB interface. Main loop of firmware program contains two major successive operations: (a) waiting for IB delimiter and receiving a command and (b) executing the command. If the command is in error (i.e. too long or an invalid command code) it is rejected and a Negative Acknowledge (NAK) is returned to the PC.

Data and bus command transfer have a time limit for each transferred byte. If time-out occurs the microcontroller's program sends an error code to the PC and the loop repeats. When USB goes into suspend state the transfer is also interrupted and is followed by powering off the GPIB drivers and putting the microcontroller into low-power mode. The microcontroller wakes up again when another byte is received. The program was written in C language, compiled by CodeVision AVR and is approximately 1500 words (3.0 KB) long.

The execution speed of the microcontroller mainly determines the maximum data transfer rate which is in our case theoretically limited to 225 KB/s. Actual measured transfer rates using HP54522A digital oscilloscope were 190 and 174 KB/s for talking and listening, respectively.

6. PC driver

We designed a driver in LabVIEW /12/ environment as a collection of VIs, since LabVIEW is widely implemented at our faculty and at the same time it is one of the most popular programs for measurement control and automation. LabVIEW allows fast development of complex and easy-to-use programs. The driver is compatible with the LabVIEW's built-in GPIB driver simplifying adaptation of existing programs to the new interface. Only one additional input is required, i.e. serial port number. VIs are described below.

USB GPIB Initialization opens serial port, powers on the controller and initializes it according to specified parameters (re-addressing required, REN state).

USB GPIB Write transfers data to a device. It also addresses the device if an address is given and the device is not already addressed. After data exchange, the function de-addresses the device if specified at initialization. This addressing logic is the same for all functions.

USB GPIB Read reads data from a device. The VI will stop reading when EOS is detected or specified number of bytes received. This is additional option to the EOI implemented in firmware. Data transfer mode for write and read (EOI, End Of String character (EOS)) can be adjusted with a parameter.

Command	Function
IB	Powers on and initializes the controller.
IBCx, IBcx	Sends byte *x* (binary) as a bus command. ATN line can be released after transfer or not, which enables multi-byte commands to be sent.
IBdata	Sends binary data to GPIB bus. Returns error if there is no listener.
IB?	Receives binary data till EOI is detected, timeout or cancelled.
IBB	Receives one byte (ignores EOI).
IBIx	Sends identification string to the PC.
IBZ	Conducts interface clear.
IBO	Powers off the controller.
IBe	Sets EOI mode for sending data.
IBt, IBT	Sets timeout for handshake and total timeout.
IBm	Controls state of REN line.
IBDE\AA, IBDTx, IBDCx, IBDMx, IBD?x	Debug commands enables each line of the GPIB bus to be independently controlled. They are useful when developing new protocol and for hardware debugging.

Table 1: Controller commands

USB GPIB Clear resets a device or all devices using Selected Device Clear (SDC) or Device Clear (DCL). Which command is send depends on whether an address is given.

USB GPIB Read STB reads status byte using serial poll.

USB GPIB Wait RQS calls USB GPIB Read STB in regular interval waiting for request bit to be set. This bit is an indicator that a device needs attention (ex. measurement is finished).

USB GPIB Close powers off the controller and closes serial port.

The microcontroller performs only simple tasks while higher level operations are implemented in the driver. For example: the controller cannot address a device, send data to it and de-address using a single command from the PC. This must be implemented in three steps, i.e. with three controller commands. This separation enables many features to be implemented or updated without changing the controller's firmware and enables faster and simpler updates since the reprogramming of the microcontroller is not required.

Since many other programs (Visual Designer, FieldPoint and HPVee and other programs like MATLAB, C++ or Visual Basic) support virtual serial port, our USB based GPIB controller needs no hardware or firmware changes, but only specific drivers to be implemented in these programs.

7. Discussion and conclusions

The USB GPIB controller was developed as a compact plug-and-play device with maximum transfer rate of 225 KB/s. Predominant use of SMD components in the controller makes its fabrication easily automated and lowers the cost of components.

Firmware of the microcontroller can be upgraded to support other GPIB features like operation as a device. Upgrading can be very simplified by using newer version of microcontroller, ATmega8515, which features self-programming that enables firmware to be upgraded directly from a PC (without opening the controller). This would enable users to download new firmware from the Internet and install it themselves. At the same time, ATmega8515 with its 16 MHz clock frequency would double maximum transfer rate, since the microcontroller is the limiting factor. The speed can be further increased by using special GPIB circuits.

Many features can be added in the driver and therefore performed in the field, without using a microcontroller programmer. Nevertheless, drivers can be upgraded easily in LabVIEW programming environment as well as easily adapted to other programming environments that support serial port. VISA drivers would also be beneficial and would additionally ease conversion of existing programs. Nonetheless, USB interface enables the controller to be used with both, desktop computers and notebooks. In this way, researchers may use their notebooks to run self-built virtual instruments with the USB GPIB controller in a laboratory. Furthermore, non-standard features can be also implemented. GPIB printer emulation was already successfully applied and used for digital oscilloscope's (HP 54600B) waveform printout. The controller is comparable (concerning speed and dimensions) to commercial controllers like National Instruments' GPIB USB B.

Acknowledgments

Franc Smole and Janko Drnovšek are acknowledged for helpful discussions.

References

/1/ W. Winiecki, Methodology for teaching measuring systems, Measurement, Vol. 18 (4), 1996, pp. 237-244

/2/ B. Murovec, S. Kocijancic, A USB-based Data Acquisition System Designed for Educational Purposes, The International Journal of Engineering Education, Vol. 20, 2004, pp. 24-30

/3/ M. Smith, Bridging the future of GPIB, R&D Magazine, Vol. 43, 2001, p. 10, Available: http://www.rdmag.com/features/0112gpib.asp

/4/ Choosing the Right Bus Technology: LAN, USB, GPIB, PCI/PXI, Automated test summit 2005, National Instruments' seminar, 2005

/5/ B. Glažar, M. Jankovec, M. Topič, USB based GPIB controller, EDN (Electronic Design News), 2005, in-print

/6/ M. Colloms, Computer controlled testing and instrumentation: an introduction to the IEC-625: IEEE-488 bus, Pentech Press, 1983

/7/ FT245BM USB FIFO (USB-Parallel) I.C., /FTDI datasheet/, Available: http://www.ftdi.co.uk, 2003

/8/ SN75160B, Octal General-Purpose Interface Bus Transceiver, /Texas Instruments datasheet/, 1995

/9/ SN75161B, SN75162B, Octal General-Purpose Interface Bus Transceivers, /Texas Instruments datasheet/, 1995

/10/ AT90S8515, 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash, /ATMEL datasheet/, /Online/, Available: http://www.atmel.com, 2001

/11/ B. A. Forouzan, Introduction to data communications and networking, WCB/McGraw-Hill, 1998

/12/ R. H. Bishop, LabVIEW student edition 6i, Prentice Hall, Upper Saddle River, 2001

Boštjan Glažar, univ. dipl. ing. el.
Dr. Marko Jankovec, univ. dipl. ing. el.
Prof. dr. Marko Topič, univ. dipl. ing. el.

University of Ljubljana,
Faculty of Electrical Engineering
Laboratory of Semiconductor Devices
Tržaška cesta 25, SI-1000 Ljubljana, Slovenia
Tel.: +386 (0)1 4768 723, Fax: +386 (0)1 4264 630
E-mail: bostjan.glazar@fe.uni lj.si

Prispelo (Arrived): 01.06.2005 Sprejeto (Accepted): 30.09.2005