ISSN 0020-4412, Instruments and Experimental Techniques, 2010, Vol. 53, No. 1, pp. 79–83. © Pleiades Publishing, Ltd., 2010.
Original Russian Text © Yu.V. Tuboltsev, Yu.V. Chichagov, E.M. Khilkevitch, V.D. Simutkin, 2010, published in Pribory i Tekhnika Eksperimenta, 2010, No. 1, pp. 87–91.

A CAMAC–USB Crate Controller

Yu. V. Tuboltsev[a], Yu. V. Chichagov[a], E. M. Khilkevitch[a], and V. D. Simutkin[b]

[a] *Ioffe Physicotechnical Institute, Russian Academy of Sciences, ul. Politekhnicheskaya 26, St. Petersburg, 194021 Russia*
e-mail: tuboltsev@mail.ioffe.ru
[b] *Uppsala University, Marmorvägen 11C, Uppsala, 75244 Sweden*

Received April 13, 2009; in final form, July 16, 2009

Abstract—A controller providing communication between a computer and a CAMAC crate via the USB bus is described. For this purpose, the controller includes a DLP-USB245M module, which allows a programmer to work with the controller through a virtual COM port and, at the same time, provides all the advantages of the USB standard. We consider versions of interactions of the DLP-USB module with controller registers on a programmable logic array and on the microcontroller.

DOI: 10.1134/S0020441210010112

In spite of the fact that the CAMAC standard was designed in 1969, it still finds wide application in automated physical experiments in our country and abroad. Without analyzing all the reasons for this lengthy application, we can point to one of them—the availability of a great number of modules in laboratories acquired and created over many years. Experiments based on these modules are still being performed [1]. The connecting link between the CAMAC crate and a control computer is a controller.

There exist controllers providing communications between the CAMAC crate and a computer via parallel data transmission channels (through adapters of the ISA and PCI buses or through a parallel port of a computer [2–7]. Some of them communicate with the computer via serial data transmission channels, operating through a COM port, Ethernet, or USB. Each controller is intended for one or another purpose and has its own advantages and disadvantages.

Thus, when the controller communicates with the computer via the PCI bus, the maximal data transmission rate is ensured. As a rule, communication is provided via a multicore cable (a loop circuit) with a length of ≤2 m. For this purpose, a special conversion board (an adapter) is inserted into a free slot of the computer. In this case, in contrast to the ISA bus, the controller should be addressed via the adapter of the PCI bus by means of special program drivers, writing of which is a complicated problem [8].

In some cases, operation with standard external ports of the computer is required, e.g., while working with portable computers (lately, this has been often used in field experiments or when the user cannot or does not want to install a controller–computer adapter on the PCI bus). Sometimes it is necessary to collect data from equipment located in the crate far away from the computer, e.g., in the experimental room. In these cases, it is possible to use crate controllers operating with computers via standard serial ports, such as Ethernet or USB. To tell the truth, in the latter case, USB–RS422/485 adapters are required for operation at long distances [9]. The controllers operating via the Ethernet are, as a rule, based on PC/104 inner single-board computers [10] or high-power microprocessors and Ethernet interface cards [11].

In our opinion, the cheapest and simplest method of the hardware and program embodiment is to construct a controller with the USB port.

Figure 1 shows a block diagram of the controller. The controller contains ten 8-bit registers connected to each other by local data bus *D*, via which information is read out of some registers and written into the other registers. Four-bit address bus *BA* is used to address the registers. The registers are used for storing CAMAC commands (*N*, *A*, *F*, *Z*, and *C*), data (*R* and *W*), and status information (*X*, *Q*, and *I*). Signals *L1*–*L23* are read directly from crate modules. There is common signal *L*. The functions of the registers of the controller and their addresses are summarized in the table.

Thus, to prepare a command for writing data into any module, it is required that the module number, subaddress, function, and data be written into registers *N*, *A*, *F*, and *Wh*–*Wm*–*Wl*, respectively. The CAMAC cycle starts operating in response to the *Start* command. Preparation and execution of read commands differs only in the fact that, while executing the cycle, data, instead of being read from registers *W*, will be written in response to strobe *S1* into registers *Rh*, *Rm*, and *Rl* from the CAMAC dataway. The control commands are executed with recourse to *W* and *R* registers. It should be noted that data written into registers are stored in them and do not require re-writing in the case of their repetition.

To write/read registers of the controllers and execute other commands via the USB bus, we checked

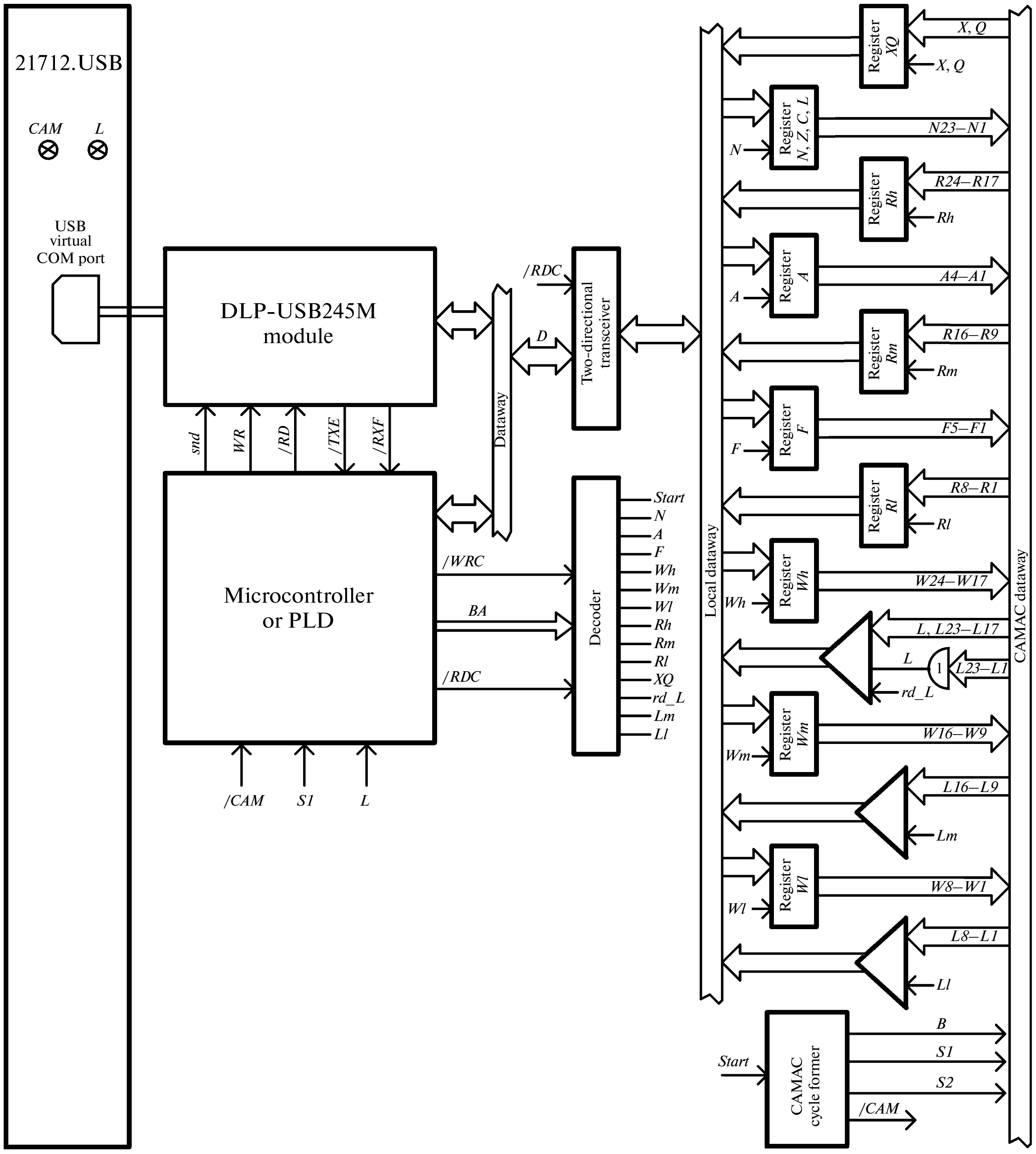

Fig. 1. Block diagram of the CAMAC–USB crate controller.

three simple versions of constructing the interface between the USB port and an 8-bit data bus, as well as with a 4-bit address bus.

One of the solutions in creating the "USB–controller registers" interface is to use a microcontroller with the USB port. For this purpose, an AT89C5131A 8-bit microcontroller with an MSC-51 architecture is used [12]. It has a 48-MHz clock frequency, 256-byte cache memory, and 1-Kbyte built-in extended RAM. A 32-Kbyte flash memory with a possibility of programming via the USB is provided for programs.

To operate with the microcontroller via the USB bus, it is possible to use the computer cross-platform library with *libusb* open initial code, avoiding needing to create a driver for it.

The drawback of using the AT89C5131A microcontroller is the need to realize the protocol of opera-

Table

Address	*BD7*	*BD6*	*BD5*	*BD4*	*BD3*	*BD2*	*BD1*	*BD0*	Note*
0	*W24*	*W23*	*W22*	*W21*	*W20*	*W19*	*W18*	*W17*	*W*(*H*)
1	*W16*	*W15*	*W14*	*W13*	*W12*	*W11*	*W10*	*W9*	*W*(*M*)
2	*W8*	*W7*	*W6*	*W5*	*W4*	*W3*	*W2*	*W1*	*W*(*L*)
3	–	–	–	–	*A8*	*A4*	*A2*	*A1*	*A*
4	–	–	–	*F16*	*F8*	*F4*	*F2*	*F1*	*F*
5	*I*	*C*	*Z*	*N16*	*N8*	*N4*	*N2*	*N1*	*I*, *C*, *Z*, *N*
6	–	–	–	–	–	–	–	–	Reserve
7	–	–	–	–	–	–	–	–	*Start*
8	*LF*	*LE*	*LD*	*LC*	*LB*	*LA*	*X*	*Q*	*CAM*, *LAM*, *X*, *Q*
9	*R24*	*R23*	*R22*	*R21*	*R20*	*R19*	*R18*	*R17*	*R*(*H*)
A	*R16*	*R15*	*R14*	*R13*	*R12*	*R11*	*R10*	*R9*	*R*(*M*)
B	*R8*	*R7*	*R6*	*R5*	*R4*	*R3*	*R2*	*R1*	*R*(*L*)
C	–	–	–	–	–	–	–	–	Reserve
D	*L*	*L23*	*L22*	*L21*	*L20*	*L19*	*L18*	*L17*	*L*, *L23–L17*
E	*L16*	*L15*	*L14*	*L13*	*L12*	*L11*	*L10*	*L9*	*L16–L9*
F	*L8*	*L7*	*L6*	*L5*	*L4*	*L3*	*L2*	*L1*	*L8–L1*
24	–	–	–	–	–	–	–	–	*Send*

* *W*(*H*) is high-order bit of the written CAMAC word; *W*(*M*) is mean-order bit of the written CAMAC word; *W*(*L*) is low-order bit of the written CAMAC word; *R*(*H*) is high-order bit of the read CAMAC word; *R*(*M*) is mean-order bit of the read CAMAC word; *R*(*L*) is low-order bit of the read CAMAC word; *A* is subaddress of the CAMAC module; *F* is CAMAC executed function; *N* is number of the addressed CAMAC module; *Z* is initialization of all modules of the crate; *C* is reset (clear).

tion via the USB bus at its program level [13], and this, obviously, creates a substantial limitation for the operation execution time. For example, the execution time of the read command of the 24-bit number from the CAMAC module is 60 μs.

We have decided in favor of other versions of constructing the "USB–controller registers" interface. They are related to the use of the DLP-USB245M module produced by the DLP Design Inc. [14]. The control functions of transmitting data from the DLP-USB module to controller registers and, conversely, are placed on the programmable logical device (*PLD*) in one version and, in the other version, nevertheless, on microcontroller *MC*. While using the DLP-USB module, the data transmission protocol is directly executed by this module. The computer communicates with the DLP-USB module via the virtual COM port by using drivers of the FTDI company [15]. In contrast to employing the real RS-232 interface, this decision, on the one hand, allows one to use well-known methods of operation with the COM port, and, on the other hand, supports such USB advantages as the reliability and the high data transmission rate.

The DLP-USB245M module is the adapter between the USB port and bidirectional parallel port. The module contains an FIFO receiver (128 byte) and FIFO transmitter (384 byte) of data transmitted via the USB port. Data from the FIFO receiver are read in response to signal/*RD* and written into the FIFO transmitter in response to signal *WR* through a bidirectional 8-bit parallel port. When at least1 byte is available in the FIFO receiver, signal/*RXF* is produced, and, when the FIFO transmitter is overfilled, the signal *TXE* is produced. Data from the FIFO transmitter are transmitted upon its complete filling or by the *snd* signal. The module is made in the DIP-24 housing.

The algorithm of interaction between the computer and the controller may differ, being determined, in particular, by the possibilities of the *MC* or *PLD*. The user can determine the algorithm and "sew" appropriate programs into these units.

In our case, the simplest algorithm is used, in accordance to which a sequence of alternate address/data bytes is sent from the computer via the virtual COM port, which are received by the FIFO receiver. In accordance with this algorithm, the *MC* or *PLD* should write the first byte into its register as an address, analyze it, and carry out one of four operations:

(i) rewriting of the second byte into one of the registers of the controller at the address indicated by the first byte;

(ii) rewriting of the content of the controller register indicated as a first address byte in the FIFO transmitter;

(iii) execution of the CAMAC cycle; and

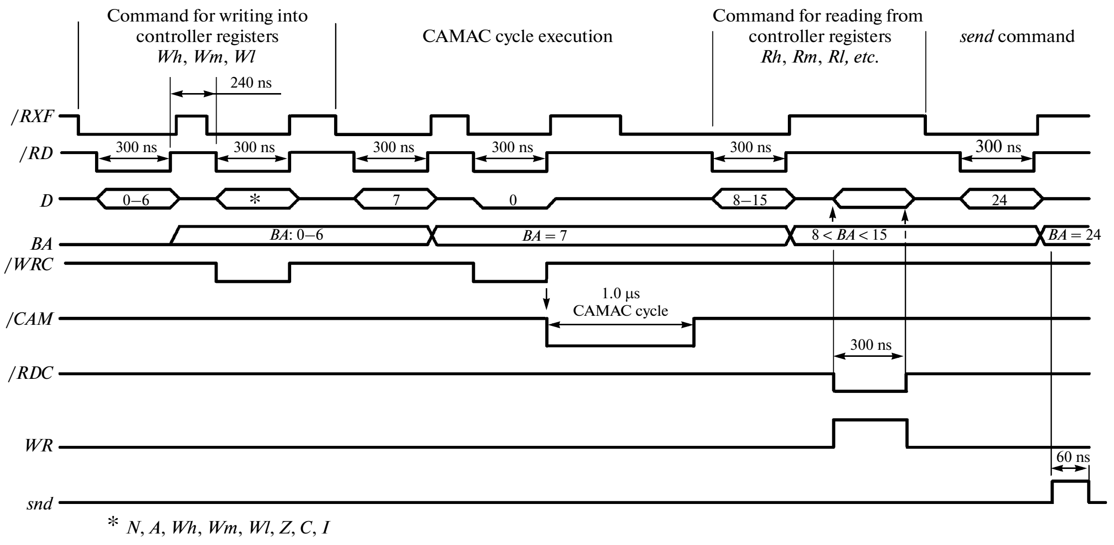

Fig. 2. Timing diagram of the PLD-based control unit.

(iv) execution of the data transmission command from the FIFO transmitter (*snd* signal).

In accordance with this algorithm, we designed two versions of control of data transmission from the DLP-USB module to registers of the controllers, namely, on an AT89C5131A microcontroller and EPM3064ATC-44 PLD [16]. The version of using the PLD is the most advantageous in speed of operation.

Figure 2 shows a timing diagram of the control unit based on the PLD. The control unit operates as follows. When signal/*RXF* appears, the/*RD* signal for reading the FIFO transmitter is produced. The first byte is stored in the address register. Later, depending on the received address (see table), one or another operation is executed. If this address is an address for writing into the controller register, the second signal/*RD* is produced. In response to it, the next byte is placed on the dataway of the controller and written into the register corresponding to the address. Writing is carried out in response to the trailing edge of signal/*WRC*.

When address 7 appears, the CAMAC cycle is triggered in response to signal/*WRC*.

The signal/*RDC* is produced when addresses correspond to read commands from registers of the controller, and, in response to this signal, data from the corresponding register are placed on the local dataway and written into the FIFO transmitter in response to the trailing edge of signal *WR*. Regardless of the number of data bytes written into it, their transmission time is ~1 ms. Therefore, to increase the data transmission rate, it is expedient to transmit them in batches, i.e., to completely fill the FIFO transmitter.

The same is valid for the FIFO receiver. It is also expedient that this receiver be completely filled with data from the computer, which should be taken into account while writing user programs.

The *snd* signal is produced at address 24, and this signal provides data transmission from the DLP-USB module to the computer.

As a rule, the highest operating speed of the controller is required in experiments when reading data arrays from measuring modules. For this purpose, a circuit for executing a more integrated command (execution of CAMAC cycles with rewriting of registers *Rh*, *Rm*, and *Rl* into the FIFO transmitter) is "sewn" into the PLD. The rewriting of the registers starts during the CAMAC cycle after signal *S1* (exactly "after," since, in response to signal *S1*, data have already been written into registers *Rh*, *Rm*, and *Rl* during the CAMAC cycle). Thus, the execution time of the command for reading a 24-bit number is only 2.8 μs; in this case, it is possible to ensure a 3.8-μs command repetition rate, while the similar operation time is 5.2 μs if the commands sent via the virtual COM port to the DLP-USB module are executed in series.

In an embodiment of the AT89C5131A microcontroller-based control unit for transmitting data from the DLP-USB module, it operates with two FIFO modules and registers of the controller via the two-dimensional parallel port as with an external data memory. In this case, the same timing diagram is kept as in the case in which this unit is based on a PLD, the only difference being that the command execution time becomes four times as long. In spite of this, some users prefer to use a version of the microcontroller-based control unit in experiments that do not require a special high speed of operation, because the programming language (C or Macroassembler) is more understandable. In this case, there is a possibility of executing some of the data acquisition, storage, and preprocessing programs in the crate at a microcontroller level. This allows one to optimally distribute program resources.

Libraries in Delphi and C languages are created for operation of applied programs on the computer. They provide programmers with a simple interface of operation with the controller, regardless of the selected version of its design.

The USB interface for data and addresses of the DLP-USB245M-based controller registers allows one to create new controllers operating with the standard port. In addition, by embedding a small additional card containing the interface on it, we managed to upgrade 21712F controllers that have already been operating in various experiments [7].

At present, CAMAC–USB crate controllers are used for automating physical experiments at the Ioffe Physicotechnical Institute (St. Petersburg), Khlopin Radium Institute (St. Petersburg), Leipunskii Institute of Physics and Power Engineering (Obninsk, Kaluga oblast), and Uppsala University (Sweden). The possibility of using the controller for communications via the virtual COM port allowed programmers to use well-known methods of operation with the COM port, in addition, having kept such advantages of the USB as the reliability and a high data transmission rate.

The USB interface with a virtual COM port can also be useful for other applications.

REFERENCES

1. Murin, Yu., Babain, Yu., Chubarov, M., et al., *Nucl. Instrum. Methods Phys. Res. A*, 2007, vol. 578, no. 2, p. 385.
2. Struck Innovative System GmbH // <http://www.struck.de/sis5100.htm/>
3. WIENER, Plein & Baus GmbH // <http://www.wiener-d. com/products/17/33.html>
4. Highland Technology // <http://www.zpeng.com/jenet>
5. Yasu, Y., Inoue, E., Fujii, H., et al., *Proc. XIII IEEE-NPSS Real Time Conf.*, Montreal, Canada, 2003, pp. 18–23.
6. Computer Methods // <http://www.computer-methods.com/>
7. 21712F CAMAC Controller // <http://www.technoexan.ru/>
8. Oney, W., *Programming the Microsoft Windows Driver Model*, Microsoft Press, 2003. Translated under the title *Ispol'zovanie Microsoft Windows Driver Model*, St. Petersburg: Piter, 2007.
9. USconverters.com // <http://www.usconverters. com/>.
10. Highland Technology // <http://www.highlandtechnology.com/DSS/M250DS.html/>
11. Hytec Electronics LTD <http://www.hytec-electronics.co.uk/1365.html/>
12. Atmel Corporation // <http://www.atmel.com/>
13. Agurov, P.V., *Interfeis USB. Praktika ispol'zovaniya i programmirovaniya* (USB Interface: Practice of Application and Programming), St. Petersburg: BKhV, 2005.
14. DLP Design, Inc. // <http://www.dlpdesign.com/>.
15. Future Technology Devices International Ltd. // <http://www.ftdichip.com/>
16. Altera Corporation // <http://www.altera.com/>