FIELDBUS COMMUNICATION IN CONTROL SYSTEM OF FLNR

V. Aleinikov, S. Paschenko, JINR, Dubna, Russian Federation

Abstract

The existing computer Control System of FLNR is being renewed to fulfil the increasing requirements of the accelerator operations. The purpose of improvement is to provide the physicists with ion beams of higher stability and quality by means of higher reliability and flexibility of parameters of the accelerator. The communication is an essential component of the control system. It is a guarantee of real time data acquisition and processing. The joint work of units is possible only by use of the appropriate standards of communication between these parts. For low levels (field level), i.e. levels of industrial controllers, sensors, actuators and transducers the standard fieldbus system does not exist yet. This area develops now due to efforts of the separate companies or their groups, and yet it is not clear which of the systems will be the standard. In studying the various fieldbus choices it's apparent that the benefits are not specific to a particular protocol. The Control System Fieldbus Protocol (CtrlBUS) was developed in FLNR in 1997. It is a high-speed, multi-master device- and process-level protocol based on distributed intelligence and master-slave communication. This paper considers the concept of the CtrlBUS protocol and gives a brief description of the CtrlBUS modules based on Intel 8051 class microcontrollers.

1 INTRODUCTION

The analysis of modern embedded systems and operating experience showed that the most promising technology for control and acquisition of distributed data is a network of 'smart' field devices, such as controllers, transducers, actuators and sensors, i.e. fieldbus. However there is a question: «How should all the components of the control system be joined electrically and which communication protocol should be used?»

As a physical interface, the following electrical standards have gained widespread acceptance: RS-232, RS-422, RS-485, IEEE-802, MIL-STD 1553. Our experience showed standard RS-485 to be the most promising. One of its advantages is good stability to interference. All the leading companies of the electronic industry use that standard.

There are many standard communication protocols used in industrial electronics: MODBUS[1], PROFIBUS [2, 3], CANbus [4], WorldFIP [5], Fieldbus Foundation [6], etc. At present Fieldbus Foundation is attempting to develop the common standard. Ref. [7, 8] help to understand the differences between the various fieldbuses. Each of them has been invented at a different time, by a different company and for different purposes. Let us specify the requirements that a communicating protocol for the field level should meet.

The first requirement is that the system should have highly fast response to enable a real-time mode of operation. The efficiency of the protocol depends on two factors:

- the hardware fast response;
- the message frame efficiency.

The first component is defined by the quality and length of the cable, the network topology as well as by the characteristics of the hardware that provides the transmission of electric signals. At present the communication facilities for industrial networks have an transfer rate of up to 12 Mbit/sec (Siemens, Motorola, Texas Instruments).

The second component depends on the message frame format, data coding and error detection. Unlike traditional networks, such as Ethernet, where performance is measured in throughput when transferring large data blocks, fieldbuses are optimised for the exchange of short point to point status and command messages.

The second requirement is that fieldbus should provide multi-user bus access. It allows to have multiple operator consoles and supervise process control from researcher computers.

The third requirement is that the network must be independent of host-computer reliability as well as stable to single node malfunction.

The fourth requirement is that the network must be handy to administer.

2 BASE CONCEPT

To make the protocol efficient, one must use the peculiarities of the communicating at the field level.

The analysis of embedded systems showed that 99% percent of the messages in fieldbus are read operations. The typical node of data acquisition and the control system in FLNR is multipurpose microprocessor module controlling over 20-50 physical signals. On the average, to control one physical unit the module checks six parameters: two analog and four digital signals. One module serves 8 peripheral facilities that means we have to make 8x6=48 readings in order to get full information about physical objects. In every reading data is supplemented with about six bytes of housekeeping information (source address, destination address, command code, checksum). Moreover to get each parameter it is necessary to send request command. It is

obvious that information can be gained at much higher rate if all the parameters of the units are **read in block** from the module in one access and placed into the memory of the computer. Then that «copy» of the module is to be used to get the parameters separately. In doing so, the information can be updated in significantly (5-10 times) less time because the total amount of housekeeping information, the synchronisation time and the frame processing time decrease.

We have already noted of a need that exists for the bus to be controlled from several consoles, that is, a need for a **multi-master** mode. What goes on in a bus with several masters? In the supervise mode, upon authorisation and connection to the bus every master makes esquires about its controlled units. Because, as a rule, all the host computers control similar sets of controlled units, or the sets are almost similar, the same read instructions will be directed to the same modules. The more the network has masters, the longer every of them waits for its turn to make inquiry about the same information that has been required by its predecessor. In these circumstances it is reasonable to do as follows: one master makes inquiries about its controlled facilities, the others **intercept** the answers of the modules and write the data in the memory of their host computers. In such a manner the information is updated at a higher rate, because the masters with no right to access the bus use the right of the active master. The interception denies repetitive information to be transmitted through the bus allowing the response of the system to an emergency situation to be made faster. In our opinion the effective exchange rate will be 5-30 times higher depending on the system complexity.

If block data transmission and interception are used, then the information flow in the control system bus will consist of repetitive inquiries sent in turn to every module. The replies to those inquiries are equally interesting to every master. Thus, information about the parameters of all the controlled facilities is continuously available in the line. Is it possible that the processor of the bus controller should make interrogation of all the modules and write the data in the memory of the PC, which would **relieve** the processor of the host computer of that routine job? It turns out to be possible. The effectiveness of such an approach is extremely high. Because the bus controller will take the trouble of making inquiries to the peripheral units, the computer processor will be given additional resources, which can to be used for other tasks, for example, for redisplaying the images on the screen at higher rate. In doing so, the information on the screen and the copies of the modules in the memory of PC can be updated asynchronously and independently of each other.

Such an algorithm is applicable only in supervisor mode. Control commands that are generated by the operator interrupt block read cycles. Their priority is higher than that of block readings. The information flow has only an insignificant fraction of operator commands, and they don't influence the rate of data updating.

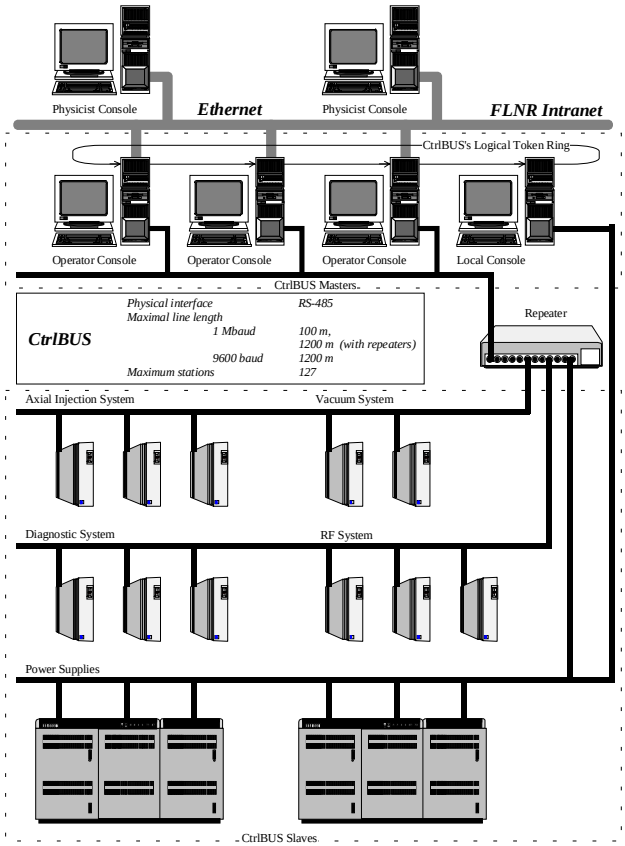

Figure 1: The integration of CtrlBUS in a control system of FLNR

Interception, which allows a controller to use the answers to the others controllers, has been considered only for active user nodes (masters). Having received a block of complete data from the module, every controller writes it in the memory of its host computer. Using those data, the program that runs on PC finds out which state units are in and in the case of emergency, issues the corresponding commands (for example, to disable, to unload certain units etc.). Because those emergency commands are to be issued by that program, then if it is out of order, there won't be any emergency commands. Undoubtedly such a situation must not take place. It is necessary that emergency situations can be processed at different layer. This suggests that the task to deal with emergencies should be entrusted to the program executed by **the processor of the module**. If such a module intercepts the messages transmitted through the bus, then it will be capable of picking up that information and taking appropriate measures to avert emergency. Again we are coming to the idea of **interception by a slave**. In this case interception is advantageous not only from the viewpoint of the traffic being decreased but also from the

viewpoint of the system response to emergency getting faster and the reliability becoming higher.

There is another question. “What will happen if the control console turns off?” This means that the cyclic status scan of the modules performed by the controller installed in PC will end. The passive user nodes will no longer receive the latest information about the other units. Those units will be disabled automatically. It’s bad because it may take a long time to start some disabled units again. That problem can be solved if one of the slaves is allowed **to act as a master**. In the normal mode the slave doesn’t need that ‘activity’, but in the situation involved it will substitute the master successfully. It will do well with status scan and support the bus life-cycle.

For our applications we have developed CtrlBUS communicating protocol. Based on a real-time capable token bus principle, CtrlBUS defines multi-master and master-slave communication relations, with cyclic or acyclic access, allowing transfer rates of up to 1 Mbit/s. In Figure 1 the integration of CtrlBUS in a control system of U-400 cyclotron is shown.

3 CTRLBUS CONTROLLER

As shown in Figure 2 the ATMEL CPU AT89S8252 [9] is the core of the CtrlBUS controller. Its kernel bases on the i80C51 compatible microcontroller. It includes 8K bytes of Downloadable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchdog timer, two Data Pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry.

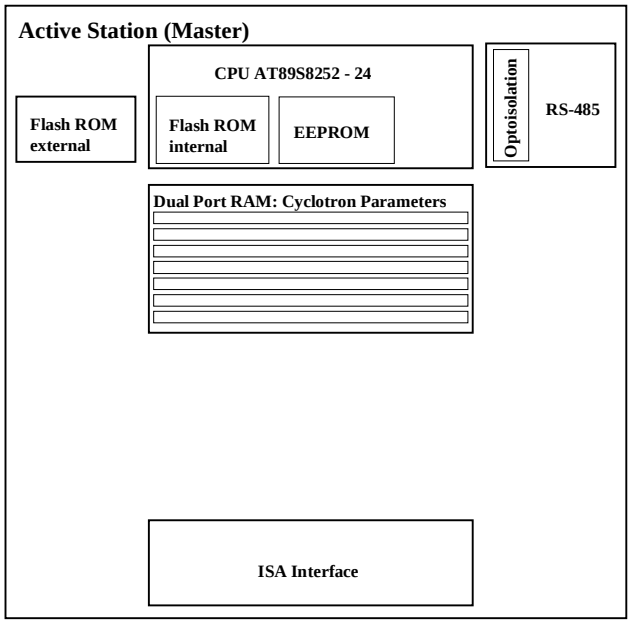

Figure 2: CtrlBUS controller block diagram.

The CtrlBUS controller can communicate with various parallel host busses via the Dual-Port RAM. It holds data from all modules that are on-line. The firmware is stored in the internal EEPROM and the external Flash ROM. The internal Flash ROM holds protocol data and operation parameters. External fast UART TL16C550C [10] is used for sending and receiving CtrlBUS messages. The standard CtrlBUS baud rate (preset 920 kBaud) are derived from the 16 MHz quartz.

4 CONCLUSION

From the above it is seen that the services of an active and a passive user node are identical in many respects. This allows one to unify the electrical circuit and software for all nodes.

To date, all the parts of the modules of the CtrlBUS system have been designed, created and tested. The results achieved allow one to state that the CtrlBUS system meets the requirements that specified for the field level communication in the control system of the FLNR. The transmission performance is adequate to our control requirements.

REFERENCES

[1] http://www.modicon.com/techpubs/toc7.html
[2] DIN 19245, Part 1+2, “PROFIBUS – Process Fieldbus”, Beuth Verlag, Berlin, Germany, 1991.
[3] http://www.profibus.com/data/main.html
[4] http://www.elmomc.com/PPT/CAN/ppframe.htm
[5] http://www.worldfip.org/
[6] http://www.fieldbus.org/
[7] Jane S. Gerold, “Fieldbus of Dreams”, Control Engineering, September 1996.
[8] http://www.gespac.com/html/Divers_product/
[9] “ATMEL AT89S8252 datasheet”, ATMEL, 1997.
[10] “TL16C550C Asynchronous communications element with autoflow control”, Texas Instruments, March 1994.