DATA ACQUISITION SOFTWARE FOR INDUSTRIAL SYSTEMS

MSc.Eng.Drd. Adrian PELCZ
Prof.Dr.Eng. Francisc SISAK
Senior Lecturer Drd.Eng. Liviu PERNIU
Assoc.Prof.Dr.Eng. Adrian CRACIUN
Assoc.Prof.Dr.Eng. Sorin Aurel MORARU

Automatics Department, "Transilvania" University of Brasov, M.Viteazu street, no.5, 500174, Brasov, Romania, phone: +40 0268 418836, apelcz@vision-systems.ro; sisak@marconi.unitbv.ro; roxanab@unitbv.ro; craciun@vega.unitbv.ro; smoraru@vision-systems.ro.

Keywords: data acquisition, monitoring, serial communication, Internet, JBus

***Abstract**: This paper presents a data acquisition software for an industrial network of electronic counters, which measure currents, voltages, energies, frequencies and many other measures. This software is a key part in the architecture presented in the "Architecture for Industrial Monitoring Systems" paper and represents the Driver level.*

The modern monitoring systems usually contain specialized software used to do all the monitoring tasks on the computer that is connected by hardware components to the industrial equipment being monitored. Because the newly presented architecture is based on a multi-tier approach, the data acquisition level, represented by the Driver, is one of the most important aspects in monitoring: it gets the data from the physical process into the computer.

The presented software program collects the data using a 9-pin serial port on a computer. The protocol used is JBus, which is derived from ModBus.

1. INTRODUCTION

In this paper we present a software program designed for fast and safe data acquisition from a serial RS-485 network of Sepam 2000 electronic counters. The tool used to develop the software application is Microsoft Visual C++ .Net. The serial communication protocol used is JBus (a protocol based on ModBus).

The RS-485 network of Sepam 2000 counters connected is to the 9 pin serial port of a computer using a RS-485 / RS-232 converter.

This application has the following roles:

- Tests the communication with the Sepam counters connected to the PC;
- Collects the values measured by the Sepam network of counters, using the JBus protocol over the PC serial port;
- Shows the measured values in text mode (configurable from option);
- Connects to a server application using the TCP/IP protocol;
- Sends the measured values as they are being read from the Sepam network to the server over the Intranet or Internet connection.

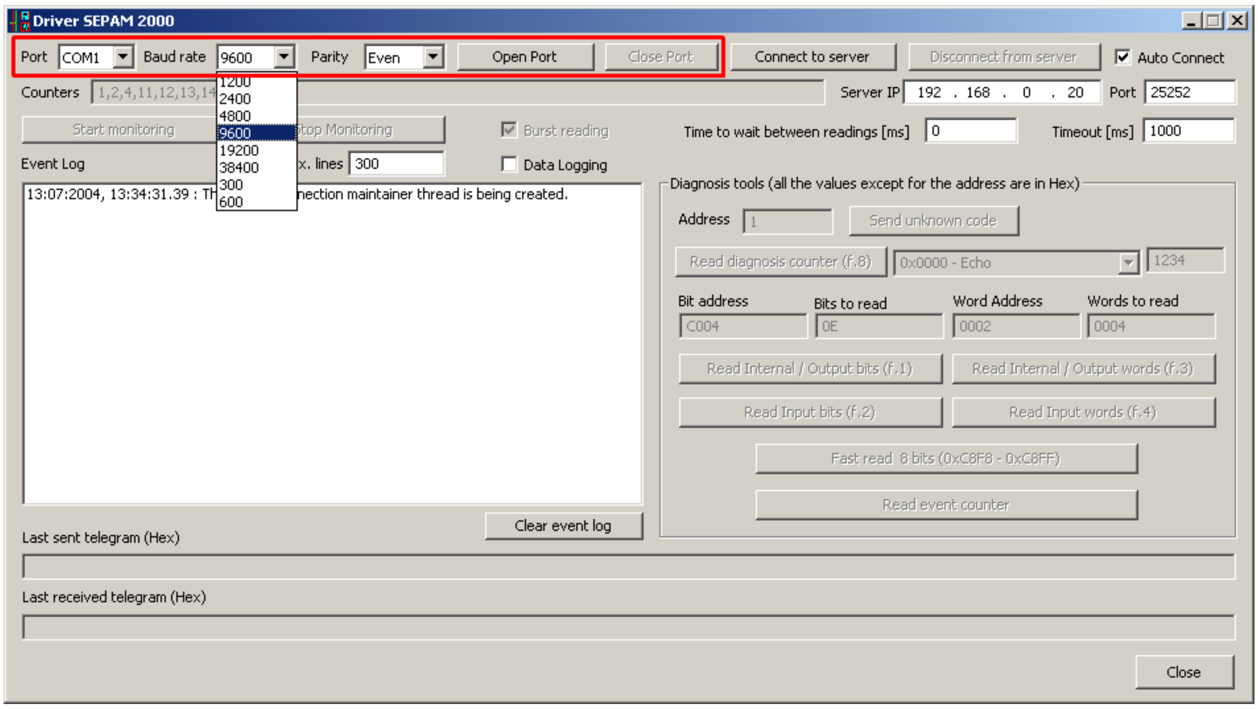
The application tests the CRC for each telegram sent and received for assuring the validity of data. This software was optimized in terms of acquisition speed and reliability. It acquires data at almost full baud-rate by processing data and sending information to the server in parallel with the data acquisition. It features a server connection maintaining mechanism (when the server is restarted for example, the Driver automatically reconnects) and also many mechanisms to handle hardware errors such as timeouts, CRC errors or inconsistent data.

2. FUNCTIONAL DESCRIPTION

Because the Driver has very precise tasks to accomplish, we will describe each functional module separately, pointing out the connections with the other functional modules.

2.1. The Serial Connection

The Sepam Driver allows the use of any free serial port, as well as any communication parameters configuration. So, before opening the serial connection, the user can choose the serial port, the baud rate and the parity, using the combo-boxes marked in fig. 1. The port opens when clicking the "Open Port" button. It can be closed at any time by clicking "Close Port". There is a separate thread that deals with the serial port operations such as writing and reading. This thread actually waits for characters to come on the serial port and when it reads some characters, it places them into a buffer and another thread analyzes the buffer all the time to check whether a valid telegram was received.



Port COM1 Baud rate 9600 Parity Even ☑ Auto Connect

1200 2400 4800 9600 19200 38400 300 600

Counters 1,2,4,11,12,13,14 Server IP 192 . 168 . 0 . 20 Port 25252

☑ Burst reading Time to wait between readings [ms] 0 Timeout [ms] 1000

Event Log Max. lines 300 ☐ Data Logging

13:07:2004, 13:34:31.39 : Th… nection maintainer thread is being created.

Diagnosis tools (all the values except for the address are in Hex)

Address 1

0x0000 - Echo 1234

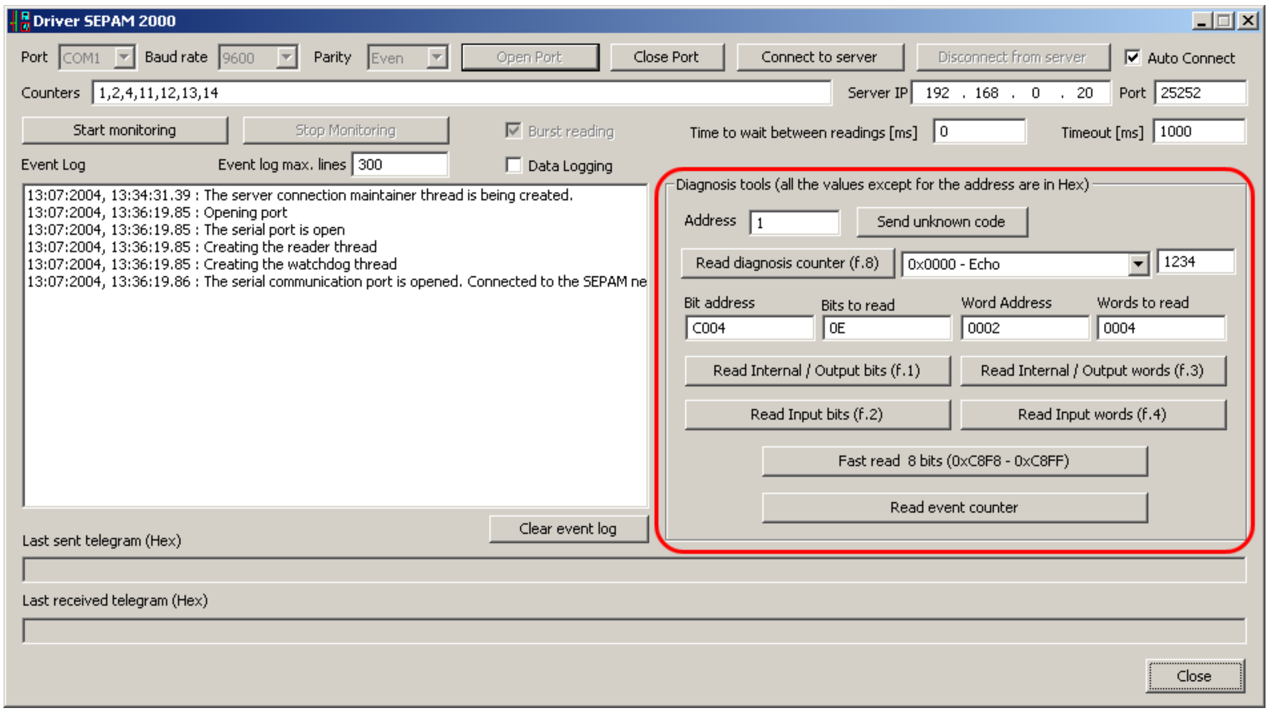
Bit address C004 Bits to read 0E Word Address 0002 Words to read 0004

Last sent telegram (Hex)

Last received telegram (Hex)

Fig. 1. The Driver window when the serial port is not opened.

When the serial port is opened, all the controls regarding the communication with the Sepam network will activate, like in fig. 2, where the diagnosis tools are marked.



Port COM1 Baud rate 9600 Parity Even
[✓] Auto Connect
Counters 1,2,4,11,12,13,14
Server IP 192 . 168 . 0 . 20 Port 25252
[✓] Burst reading
Time to wait between readings [ms] 0
Timeout [ms] 1000
Event Log
Event log max. lines 300
[ ] Data Logging

```
13:07:2004, 13:34:31.39 : The server connection maintainer thread is being created.
13:07:2004, 13:36:19.85 : Opening port
13:07:2004, 13:36:19.85 : The serial port is open
13:07:2004, 13:36:19.85 : Creating the reader thread
13:07:2004, 13:36:19.85 : Creating the watchdog thread
13:07:2004, 13:36:19.86 : The serial communication port is opened. Connected to the SEPAM ne
```

Diagnosis tools (all the values except for the address are in Hex)
Address 1
0x0000 - Echo 1234
Bit address C004
Bits to read 0E
Word Address 0002
Words to read 0004
Last sent telegram (Hex)
Last received telegram (Hex)

Fig. 2. The Driver window when the serial port is opened.
Diagnosis tools are marked.

2.2. The Diagnosis Section

This section fully implements the JBus protocol read functions and it is very useful in testing the serial communication with any of the counters in the RS-485 network connected to the serial port. This section is marked in fig. 2 with red.

The diagnosis can be made for a single cell (counter), it's address being specified in the "Address" field. There can't be two or more cells with the same address in the same RS-485 network, so a specific cell is identified by the address, with a value from 0 to 255. The read functions are:

- Read diagnosis counter:
  - Echo;
  - Reset all to 0;
  - Frames with no CRC error;
  - Frames with CRC error;
  - Exception replies;
  - Frames addresses to this SEPAM;
  - Broadcast messages received;
  - Exception replies (incl.broadcast);
  - Sepam 2000 not ready replies;
  - Characters not processed;
- Read Internal/Output bits;
- Read Internal/Output words;
- Read Input bits;
- Read Input words;
- Fast Read 8 bits;

- Read event counter;

For reading a diagnosis counter value, the user has to choose which diagnosis counter wants to read and, if applicable, a supplementary hex value. For reading the value (bits or words) of some memory area, the start address and number of bits/words must be specified.

When the button corresponding to a specific JBus function is pushed, the Driver will create and send a telegram corresponding to the user's action and wait for the specified (by address) Sepam cell to send the response. The response, if it arrives in time, it is processed by the Driver, and the value, in a human readable form, is printed in the Event Log. If the response telegram does not arrive in the time specified in the "Timeout" field, the Driver stops waiting for that telegram and prints a timeout error message in the Event Log. This timeout is being generated by a "watchdog" thread, which always monitors the communication process and checks the time from the last request. If this time becomes larger than the time specified as Timeout time, the watchdog thread cancels the current operation and sets the Driver in the initial status (that is, waiting for commands from the user).

2.3. Serial Communication Procedure

The communication procedure is a bit more complex than it appears. There are several problems that may occur and with which the Driver must deal. At first, the problem seems simple: The Driver sends a telegram to the serial network, and the counter to which it was addressed, sends the response back to the Driver. In this situation, the procedure would have been very simple, but also very unreliable. In a system that must be working around the clock, with no interruptions, the smallest error could be fatal, stopping the data acquisition process. So, we had to build a data acquisition software that can handle any kind of error.

Since the possible error causes can be many, instead of taking all the situations into account, we came up with a little trick: we expect everything to go well, and if it does not in a certain amount of time (specified in miliseconds), we consider that the current operation failed because of an error and drop it, allowing the Driver to function well for the next operations. The goal was to make the Driver stable under any circumstances, and more than that, if an error persists for a while, when the error cause is gone, the Driver should work correct. For example, the Driver should not block if a telegram with a CRC error arrives, or an incomplete telegram arrives, or if nothing arrives at all.

The logic that solves all these problems is quite simple: The Driver works on a request-answer basis. So, when it makes a request, it knows what kind of response it is waiting for (from which counter, how many bytes the telegram must have, which function code should it contain, etc). Basically, when the Driver sends the request over the serial port, it also retains in a separate thread the kind of answer it expects, and the time in miliseconds when the request was posted. When data arrives on the serial port, it is placed into a buffer and a specialized function checks the buffer content to see if it matches the expected answer. When the buffer content matches the answer (length, CRC, type of telegram, etc), the buffer's content is moved into another memory area for further processing by another thread, and the buffer is voided. In this moment, the Driver can already send another request. All this time, the watchdog thread was "watching" if the time for the current operation has not been exceeded. If the time elapsed, the watchdog thread cancels all the operation (cleans the expected answer, the buffer, the current status, etc) and puts the Driver in the "standby" position. If the received telegram contains errors, it is being dropped also.

Using this mechanism, only complete and valid (from the communication point of view) telegrams are being processed.

Processing a telegram means getting the raw information contained into it and transforming it into a human readable value.

2.4. The Monitoring Section

This section is responsible for controlling the Sepam network monitoring. In fig. 2, in the "Counters" field the user can specify the list of counter addresses from which the monitored data will be acquired. The monitored measures are currents for each phase, voltages for each phase, cos(ϕ), frequency, consumed active energy and consumed reactive energy. The Driver collects the measured values for each counter in the list, and when it is finished it starts again with the first counter. The measured values can be listed in the Event Log if the user checks the "Data Logging" box. For each counter the Driver makes two reads from the counter's memory (the currents, voltages, cos(ϕ) and frequency are stored in a relatively compact memory zone and the consumed energies into a different compact zone. The Driver can be configured to wait a certain amount of time in miliseconds between reads (from receiving the response until sending the next request). This time can be 0. The timeout represents the time to wait from the start of the telegram sending, until the operation will be canceled by the watchdog thread. The estimated time for sending the request and receiving the response for a complete read, using 4800 bps, is 200 ms, and so, a Timeout smaller than the normal response time will always generate false timeout errors.

The monitoring mechanism retains the measures for all valid cells for each cycle into a buffer (the current measures buffer). If during monitoring a cell from the Sepam network does not respond, no data for it will be placed into the buffer. When the cycle ends, the buffer is voided and the process restarts.

By using a lot of separate threads, all these mechanisms do not affect the data acquisition speed (because the processing power is a lot larger than the communication speed). So, the data reading actually goes at full serial connection speed, the data processing and storing, together will all the other mechanisms, being done in the background. These mechanisms provide a very good acquisition speed (a 10 cells network having 120 total measured values is being completely refreshed every 2 seconds at the baud rate of 9600).

2.5. The Server Connection

This Driver can independently from the monitoring process also send the acquired data to a data acquisition server (the Server level in the "Architecture for Industrial Monitoring Systems" paper).

The user can specify the server IP address and the TCP/IP port on which the server listens for connections. When the Driver is connected to the server it sends the measured data at the end of each reading cycle.

Of course, sending the data to another application through TCP/IP is pretty simple and brings huge advantages in production: the Driver can communicate with the server using a local area connection (LAN) or the Internet.

Things get complicated when the possible problems with the TCP/IP communication with the server occur. These problems must not affect in any way the monitoring process or the other sections of the Driver to which it is related. So, for the server connection we have implemented a mechanism, which watches for the connection status, and, if any error occurs, the Driver manages it well.

A common problem in distance communication systems is the temporary malfunction, which breaks the connection (a short power failure, a server machine restart, server software updates, etc.). In such a case, a person would have to reconnect the Driver back to the server every time the connection interrupts. For preventing this, we have developed a mechanism - implemented into another separate thread, which always checks the connection status. If the connection is broken and it should be not, that thread tries to reconnect to the server every 5 seconds. Like this, when the connection can be re-established, the Driver will automatically connect and start sending data as if nothing would have happened.

3. CONCLUSIONS

In today's modern industrial monitoring systems, the data acquisition plays an important role. The software programs designed to acquire data from industrial processes must be reliable and must be able to deal with any kind of problem that may appear in the communication process with the equipment.

Our data acquisition software was built for integration with a larger distributed application system, but it can be very useful as a standalone application also.

By using advanced error handling mechanisms and parallel processing, we have developed an acquisition software program, able to acquire data at full baud rate speed from the process, this making the monitoring system able to show almost real-time evolutions for the measured values.

4. REFERENCES

[1] *Sepam 2000 – Metering and protection Functions*, Merlin-Gerin.
[2] Microsoft, *Visual C++ .net MSDN*
[3] http://www.ontrak.net/mfc.htm
[4] http://www.codeguru.com/Cpp/I-N/network/serialcommunications/article.php/c2483/