IEC-60870-5 application layer for an Open and Flexible Remote Unit

Verónica Medina, Isabel Gómez, Enrique Dorronzoro, David Oviedo, Sergio Martín, Jaime Benjumea, Gemma Sánchez
Departamento de Tecnología Electrónica
Universidad de Sevilla
vmedina@us.es

***Abstract*-This paper presents the development and test of the standard IEC-60870-5 application layer protocol for a Remote Terminal Unit (RTU) based on open hardware and software. The RTU hardware is an embedded system, a SoC-type design using FPGA that has been programmed with the open core LEON with Linux operating system running over it, so both the hardware and IOS are open source. For prototyping the GR-XC3S-1500 board has been used. There is no open source code available for the IEC standard protocols, so application layer protocol has to be implemented. All the software design has been made in a PC platform using standard development tools. The source code generated for the protocol has been compiled with the standard Linux gcc compiler in LEON. Several tests have been made to prove the right behavior of the protocol as well as its performance over different transmission mediums.**

I. Introduction

The first steps in the development of an Open and Flexible Remote Unit applied to telecontrol/telemetry is presented in [1]. The main idea is to develop a Remote Terminal Unit (RTU) that is open both in hardware and software.

The hardware platform is an embedded system, a SoC-type design using FPGA. The FPGA itself has been programmed with an open core called LEON [6], an SPARC compliant system capable of running Linux for SPARC. The processor is an open core (i.e. an open hardware) , this means that hardware platform is open, and also it is the operating system running over it (Linux Debian for Sparc has been installed in the system [7]). So, the RTU is, in essence, a Linux-Sparc system. This means that every program available may be used for this platform and, more important, the whole software can be developed in a very similar way than in any standard Linux programming environment.

For prototyping the RTU, the GR-XC3S-1500 board, Fig. 1, has been used. This board is supported by the co-operation between Gaisler Research and Pender Electronic Design.

For communication, the transmission channels available [1] in the RTU are Radio Frequency (RF), GSM (Global System Mobile) and GPRS (General Packet Radio System).

Telecontrol protocol stack usually implements the specification provided by the International Electrotechnical Commission (IEC), called IEC-60870-5 [2]. This document,

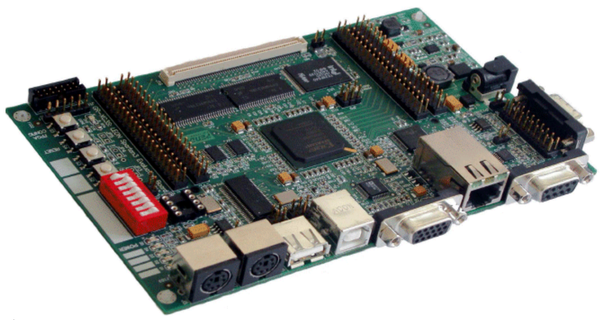
Fig. 1. RTU Prototype.

which specifies a suite of protocols, is divided into six parts and specifies an application-layer protocol and a data-link layer protocol. This standard also defines a combination of the application layer and TCP/IP transport services. Although open source software is widely used in Internet, it is not very common in the telecontrol networks area [4], so it is necessary to developed the open software for this protocol stack in the RTU .

The open source implementation and test of the IEC 60870-5 data link layer protocol, described in IEC 60870-5-2, is presented in [3] and [4]. As justified in [1] when the transmission channel is half-duplex, RF, it is necessary to use a protocol that control the medium access, that is what the IEC 60870-5-2 protocol makes.

This paper is a continuation of the RTU software development but focusing this time on the IEC 608070-5 application layer.

This paper is organized as follows; first, an overview of IEC 60870-5 series (section II) is shown. Some details of software design and field testing are described in section III and IV. Finally, in section V some conclusions and future work are presented.

II. IEC 60870-5 SERIES

As mentioned before, telecontrol protocol stack usually implements the specification provided by the International Electrotechnical Commission (IEC) called IEC 60870-5. This series follows the EPA (Enhanced Protocol Architecture) model, which simplifies the ISO standard (OSI model) in three layers, application layer, data-link layer and physical layer. The standard is divided into six parts and specifies a suitE of protocols for both application and data-link layer.

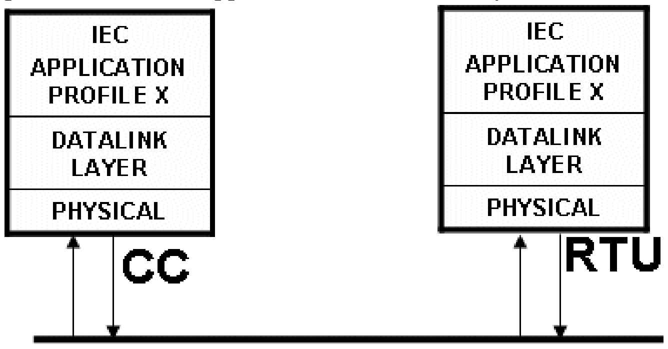

Fig. 2. RTU and CC permanently connected.

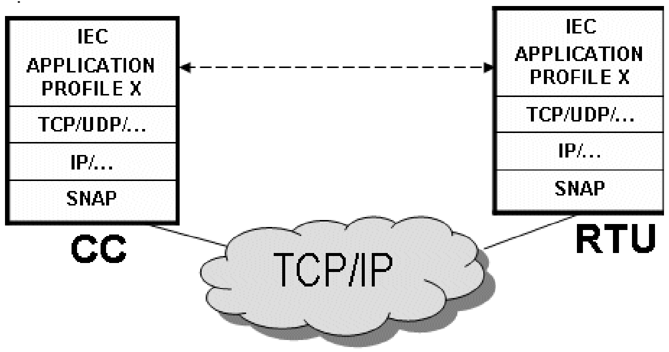

Fig. 3. RTU and CC connected via Internet

In a typical telecontrol scenario one station (primary station), called CC, controls the communication with other stations (secondary stations), called RTUs, so IEC 60870-5 specification allows real-time telecontrol applications to take place. In this sense, the series defines a set of functions (profiles) that performs standard procedures for telecontrol systems. Not all the implementation performs the same functions so an application profile must be described depending on the telecontrol system functionality.

Two different scenarios are possible where or not the CC and RTUs are permanently connected, Fig. 2, or connected via an Internet using TCP/IP Architecture, Fig. 3. For the first scenario all the protocols described in the standard has to be implemented, included the IEC 60870-5 (application functions) and for the second one, only the IEC 60870-104 because the other layers are yet implemented on most operating system (TCP/IP stack, the Internet protocols). This paper focuses only on the implementation and tests of the first scenario.

A. IEC 60870-5-5 Overview

This document specifies an assortment of basic application functions for use in telecontrol systems. Each function is composed of transfer procedures of specific ASDUs (Application Service Data Unit) between remotely communicating application processes, that is, the CC application process and the RTU application process.

There are application functions to: acquire data by polling, to send command, to transmit integrated total, to transmit a file and others. Not all the functions must be implemented, so depending on telecontrol application only a set of these functions are available, that is called application profile.

Data acquisition by polling is the only function included in the first profile of the RTU prototype.

B. IEC 60870-5-2 Overview

This document specifies the standard transmission procedures applied to different channels configurations, point-to-point or multi-point.

The transmission services available by the application layer are:

1) SEND/NO REPLY: CC sends a message to a RTU. There is no error or success notification from the RTU. Mainly used for global messages and for cyclic setpoints in control loops.

2) SEND/CONFIRM: CC sends a message to a RTU. CC waits for a positive or negative acknowledgment from the RTU. Mainly used for control command and setpoint commands as in the RTU initialization.

In case of error or any notification, an error message is sent to the application layer.

3) REQUEST/RESPOND: CC makes a survey to the RTU. If the RTU has data to transmit, it sends them to the CC. Used for polling.

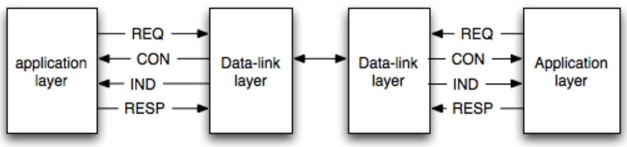

Fig. 4. Data Link Layer Service Primitives.

SEND/CONFIRM and REQUEST/RESPOND services imply a dialog between the CC and RTU. In this dialog the window size is one.

The application layer invokes the following primitives (Fig. 1) to request the above services:

1) Request Primitive (REQ): a REQ is sent by the application layer to request a certain service to the data link layer.

2) Confirmation Primitive (CON): a CON is sent by the data link layer to the application layer. It is the response to a REQ primitive.

3) Indication Primitive (IND): an IND is sent by the data link layer to inform the application layer that a message has arrived.

4) Response Primitive (RESP): a RESP is sent by the application layer as an answer to a previous IND.

When a primitive is sent by the application layer to the data-link layer, data-link layer generates a service. In case of a service that requires an acknowledgment the service is sent a number of times, called retries, until it gets response or it exceeds the number of maximum retries. This maximum retries is sent by the application layer in the primitive parameters, i.e. the ICI (Interface Control Information) of the IDU (Interface Data Unit). Depending on transmission channel latency a higher or lower number of retries is necessary as shown afterward.

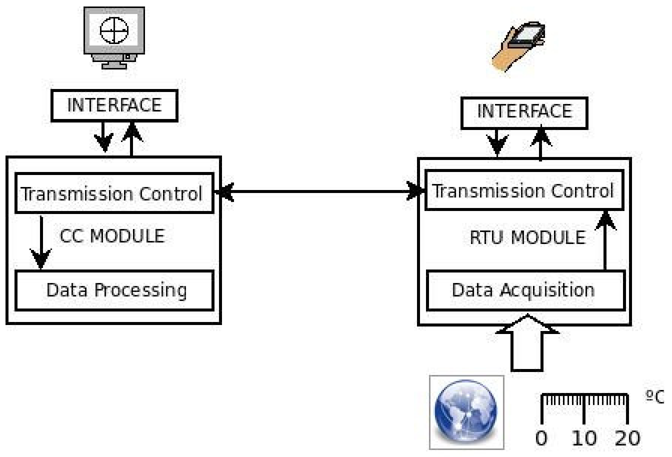

Fig. 5. Software Design.

III. Software Design

The software design is divided into two different modules: Control Center module (CC module) and Remote Module, (RTU module). Both modules have been developed in the programming language C++, under the Code::Blocks IDE on a hardware platform x86 (PC) and operating system Linux. Only standard libraries have been used in the development. In the case of Remote Module, the software also has been compiled on LEON (SPARC architecture).

In the Remote Module, two independent processes have been implemented, data acquisition and transmission control. Data acquisition process is in charge of getting the data that have to be sent to CC Module, for test purpose a temperature sensor has been used.

The data saved in memory by the data acquisition process are sent to the CC by the transmission control process. This process behaves as described in IEC 60870-5-104 specification after an initialization phase. The implementation of this process is based on the C++ Sockets Libraries.

The data acquisition process is replaced by a data processing process in the CC module, which analyzes and prepares the data received from the RTU. The transmission control process in the CC module is in charge of requesting and receiving data from the RTUs.

All functions in the CC and the RTU modules, are available through an interface, that serves as an entry point to the operations asked for by a final user, Fig 5.

As LEON has a standard Linux Debian distribution with standard C++ libraries, getting a binary file is as simple as compiling the same source code developed in the PC. It has been used gcc compiler to compile and link the RTU code in LEON.

IV. Protocol Testing

Two parameters have been analyzed, initialization time (IT), mean time required complete the initialization on a RTU, and Poll Time (PT), mean time required to complete a poll over a RTU.

Initialization process consists in the exchange of four different messages. CC sends a status of link request message to check the availability of the data link. RTU responses with a status of link. CC request for a RTU reset. Once the RTU is reset, it sends an ACK message to the CC and the initialization is completed. In all this process a total of 288 bytes are transmitted.

Polling process consists in a poll request from the CC to the RTU and a response with the requested data from the RTU.In polling a total of 144 bytes are transmitted.

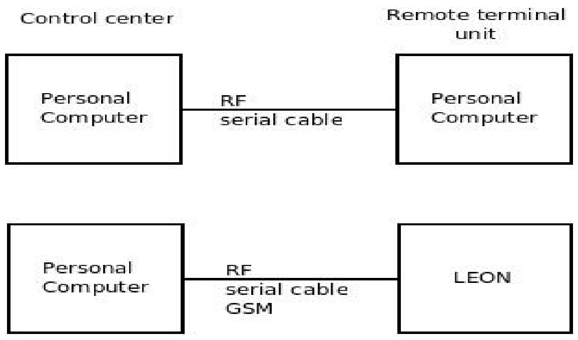

Fig. 6. Testing scenarios.

A. Field Testing

Field testing has been made over two differents scenarios: Using a PC and LEON as RTU, fig. 6.

In both cases different transmission technologies have been used in order to obtain performance measurements. The transmission technologies used were a point to point link with a serial cable, RF and GSM. For the tests the next devices have been used.

1. GSM equipment: A GSM modem has been used, specifically a Wavecom Fastrack M1306B[9]. This

modem behaves as a standard AT-command modem via a RS232 port, so the modem is connected to the RTU this way. According to device's specification, it allows a data transmission up to 14.400 bps for GSM but this feature is dependent on the GSM operator used, so it might not be available (in fact, our tests run at 9600bps).

2. RF equipment: An ICOM IC-V82 (VHF transceiver) with the digital unit UT-118. This device is D-star [10] capable, and can be connected to any RS232 device for data transmission, with a data transmission speed of 1200bps. The specification for this equipment can be found at [8]

In the following sections the results obtained for the different raised scenarios are analyzed.

Scenario 1: Using a PC as RTU

In this scenario both the CC and RTU are PCs. In the case of a serial cable as a transmission channel, the average obtained is 313'76 ms for IT parameter and 156'1 ms for PT parameter. If the transmission channel is RF, the obtained values are 5.814'2 ms for IT and 2602'1 ms for PT.

It takes not significant time in order to process the data. The difference in time is produced because of transmission delays. When transmitting in RF as it has an effective speed of 480 bps, while serial cable has a data transmission speed of 9600 bps, so the transmission times are increased.

Scenario 2: Using LEON as RTU

In this scenario the CC is a PC and the RTU LEON, Fig 7. In the case of a transmission channel using a serial cable, the average obtained is 374'7 ms for IT and 191,3 ms for PT parameter, for GSM is 2.761'7 ms and 1.125'253 ms for IT and PT parameters respectively.

As the data processing time is not significant the RF measures for this scenario are similar to the previous one.

The difference in time between both scenarios using the same transmission technologies, is consequence of the greater time of CPU processing, in the case of LEON.

Finally, it has been analyzed how affects in the application layer retries that are made at the link layer. The performance of the protocol based on the number of retries is dependent of the transmission technologies.

Error rate	10%	20%	30%	40%	50%
Retries 0	9	22	29	42	52
Retries 1	1	3	5	12	15
Retries 2	0	2	3	4	7
Retries 3	0	0	0.8	1.3	2
Retries 4	0	0	0.8	1.3	2

Table I

As shown on table I, extracted from [3] and [4], built from experimental simulation, a low number of retries implies more unsuccessful polls. But a higher number of retries implies an additional delay, because a higher number of messages are sent. From these results it is possible to determine the additional delays caused in function of the number of retries.

For example, using RF, in case of a channel with a 20% of error rate and number of retries set to 0, implies no delay but a number of 22 uncompleted polls over 100 polls. Changing the number of retries to 1, increases the number of completed polls to 97 but it generates a delay of 20 multiplied by the latency of the transmission channel, measure in seconds, more than when is set to no retries. This is because even there is a higher number of successful polls this polls may need more than one request o reply message. Setting the number of retries to one imply that some of the successful polls needed more than one retry, as the error rate of the channel is 20%, the number of sent messages will be increased up to 20 over 100 polls. There

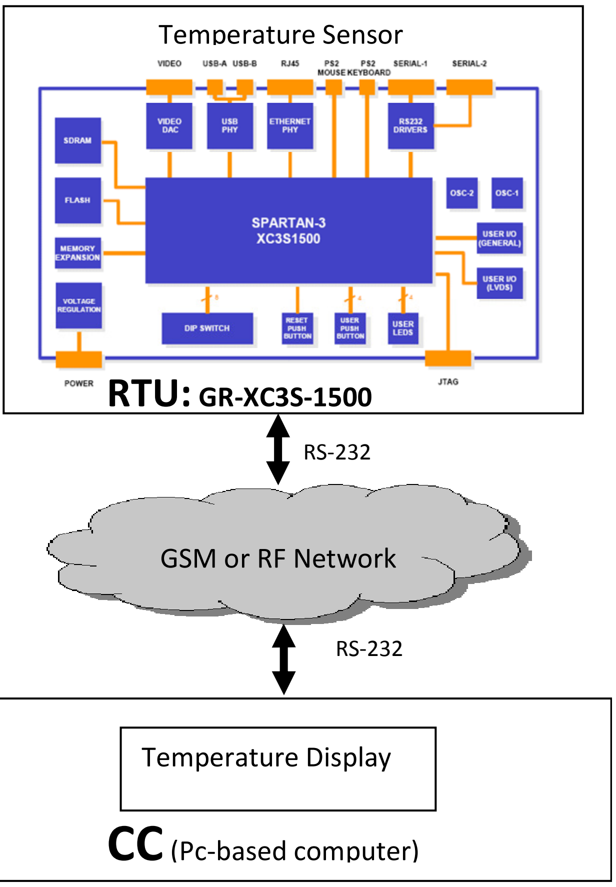

Fig. 7. Using LEON as RTU.

is a latency of 0,5 seconds in RF, which generates a delay of 11 seconds every fifty polls.

The number of retries must be calculated based on the real application. While one real application must need a higher number of completed polls another one may need lower delays. For example, in a positioning application applied to vehicles in order to know in real time the position of vehicle required a lower delay, but in order to know completely the route the vehicle has followed, a higher number of completed polls is required.

V. Conclusions

In this work the IEC 67080-5 application layer protocol has been developed and tested for an open and flexible RTU. The RTU hardware is based on FPGA that has been programmed with the open core LEON with Linux operating system running over it.

The first IEC application profile for the RTU includes the poll function, that is, all data acquired from the RTU have to be specifically requested.

All the software design has been made in a PC platform using standard development tools. The source code generated for the protocol has been compiled with the standard Linux gcc compiler in LEON.

To test the protocol, two scenarios have been tested, in one the CC and RTU were PCs and in the other the RTU was LEON.

The RTU prototype has been tested with serial cable, RF and GSM transmission channels in the field testing. In this case the focus was on setting the appropriate number of retries. As it has been shown, in the case of RF because of the high transmission delays, the cost in time for any retries is very high but setting a high number of retries, implies a lower protocol performance, although there are more successful poll each are more costly, i.e., the whole time spent is higher. Using a faster transmission technology, GSM or serial cable, allows increasing the number of retries to higher values, although the protocol performance is also reduced there are more completed polls, and the information received is more accurate.

Further researching work will be made to develop and test the IEC 60870-5 application layer for the RTU but this time over TCP/IP.

Acknowledgments

This work has been undertaken in the framework of two research projects: OFU (EXC-2005-TIC-1023) - Open Flexible Unit funded by Junta de Andalucía and TOMARES (TEC2006-08430) -Multimedia Operatives Techniques applied to Supply Electric Networks funded by the Ministry of Education and Science of Spain.

References

[1] Jaime Benjumea Mondéjar, Ana Verónica Medina Rodríguez, Isabel María Gómez González, Enrique Dorronzoro Zubiete, Gemma Sánchez Antón, Sergio Martín Guillén: Choosing the Right Protocol Stack for an Open and Flexible Remote Unit. ISIE'2008. International Symposium on Industrial Electronics. International Symposium on Industrial Electronics. Cambridge, Reino Unido. IEEE. 2008. Pag. 1668-1673. ISBN: 1-4244-1666-0.

[2] International Electrotechnical Comission, “International Standard IEC-60870-5” (6 parts). http://www.iec.ch.

[3] Enrique Dorronzoro Zubiete, Isabel María Gómez González, Ana Verónica Medina Rodríguez, Jaime Benjumea Mondéjar, Gemma Sánchez Antón, Sergio Martín Guillén: Abstract Implementing Iec 60870-5 Data LINK Layer for an Open and Flexible Remote Unit. 34th Conference of the IEEE Industrial Electronics Society (IECON 2008). Num. 34. Florida, Orlando (USA). IEEE. 2008. Pag. 268-268

[4] Enrique Dorronzoro Zubiete, Isabel María Gómez González, Ana Verónica Medina Rodríguez, Gemma Sánchez Antón, Sergio Martín Guillén. Implementing Iec 60870-5 Data LINK Layer for an Open and Flexible Remote Unit. 34th Conference of the IEEE Industrial Electronics Society (IECON 2008. Num. 34. Florida, Orlando (USA). IEEE. 2008. Pag. 2471-2476

[5] Jaime Benjumea, Francisco Pérez, Joaquín Luque: "Encouraging the use of Open Source Software in high-sensitive environments", CIGRE, Study Committee D.2, Colloquium. Rio de Janeiro (Brasil), Sep, 2003.

[6] “GRLIB/LEON3 manual”, http://www.gaisler.com

[7] A. Muñoz, E. Ostúa, P. Ruiz, M. J. Bellido, J. Viejo, A. Millán, J. Juan, D. Guerrero, “Un ejemplo de implemetación de una distribución Linux en un SoC basado en hardware Linux”, Actas de las IV jornadas de computación reconfigurable y aplicaciones (JCRA’07), pp. 85-92, Sep-2007.

[8] “ICOM IC-V82 brochure”, http://www.icom.co.jp/world/products/pdf/IC-V82_U82_LM.pdf

[9] “Wavecom Fastrack 1306M User Manual”, http://www.wavecom.com/media/files/support/Hard_platforms/Modems/Fastrack_M1306B/User_manual/Fastrack_M1306B_User_Guide_rev003.pdf

[10] D-star protocol http://www.arrl.org/FandES/field/regulations/techchar/D-STAR.pdf.