RS232 to RS485 Link Module Design for InDustrial Applications using Verilog HDL

Ajay Rajput  
M.Tech Scholar  
Deptt. of EC  
LKCT, Indore, India  
Rajputajay84@gmail.com

M.A. Khan  
Professor and Head  
Deptt. of EC,  
LKCT, Indore, India  
abuzerlkct@gmail.com

Abstract: In this paper, we propose a novel protocol conversion scheme between RS232 and RS485 interfaces. Sometimes there is a need to monitor devices like tesla meter when a beam is produced. Tesla meter is useful to provide the value of magnetic field and this field information is required to distant places in the industries so that some communication protocol help us to transfer the information from one place to another. In our work we take a situation in which a control module is to be connected to the Tesla Meter over RS232 communication interface and Operator Console (a personal computer at a distant place) over RS485 communication interface. Both Tesla Meter and operator console have different communication interfaces and their respective protocols. Therefore there is a need to have controllers within control module to handle both RS232 and RS485 ports and to convert the protocols.

Keywords: Buffer, FPGA, Protocol Conversion, RS232, RS485, Serial Communication

I. Introduction

In several industrial applications, an ECM communicates with the operator console over RS485 interface. The communication is done in a custom predefined protocol. The ECM needs to be interfaced with various subsystems. All the subsystems do not have the same interface and some of them communicate over RS232 serial interface directly with operator console. “Tesla Meter” is one such subsystem (Fig. 1). For modularity, it is necessary to bring all subsystems over RS485 interface therefore it is required to design a logic to communicate over RS232 & RS485 with their respective protocols and sent data from Tesla Meter to operator console over RS485 interface (Fig. 2).

The operator console may send a ‘history data’ request to ECM for offline analysis and the ECM must respond to this request in correct way. Therefore, a module is designed to have facility to write a data into internal memory and read & send the data as and when required by the operator console in a proper format.

II. Communication Protocol

2.1 Protocol for Tesla Meter

Tesla Meter receives a single character ‘F’ as a command and transmits ‘Magnetic field data’ as response. The ECM transmits ‘F’ and receives ‘D0D1D2D3D4D5D6D7T’, Dn represents nth character.

2.2 Protocol for Operator Console

Protocol for the communication between ECM and Operator Console is designed in the form of Command – Acknowledge-Response frames. The ECM receives Command frame and transmits Acknowledgement and Response frames in a predefined format as shown in Fig. 3, 4 and 5. The acknowledgement can be positive or negative depending upon the checksum in the Command frame.

Fig. 1: Communication between Tesla Meter and Operator Console on RS232

Fig. 2: Communication between Tesla Meter and Operator Console on RS485

Fig. 3: Command Frame
III. Controller and Conversion Unit

For the two serial communication interfaces the design is divided into three modules:

1. Controller for communication over RS232.
2. Controller for communication over RS485.
3. Inter protocol conversion unit

Other than the protocol conversion unit both the controllers have one transmitter and one receiver control logic. The modules communicate at the baud rate of 19.2 kbps.

3.1 RS232 Transmitter

RS232 Transmitter transmits ‘F’ every tens of milliseconds to the Tesla Meter. A counter is implemented to define delays between two ‘F’ characters. An internal register is used to store ‘F’ and shift registers to transmit ‘F’.

3.2 RS232 Receiver

RS232 Receiver receives the serial data from the Tesla Meter. The data consists of 11 characters which are received one by one. A character reception is initiated by the bit transition from logic ‘1’ to ‘0’ on idle line (logic ‘1’).

3.3 RS485 Receiver

On receiving the characters in the Command frame between Start Delimiter ($) and End Delimiter (*), the ASCII equivalent of HEX character is converted into the corresponding HEX value. We validate the start of frame with Start Delimiter ($) and terminate the reception on getting End Delimiter (*). The storage is done on the verification of the destination address of the received frame. Frame integrity is done by checksum calculation which is last 4 nibbles of sum of all the data values between ‘$’ and ‘*’ in hex format. The ‘Command type’ value is also extracted from the command frame for the RS485 Transmitter. This all is done with the synchronization on baud rate clock generated inside the design.

![Flow chart of RS232 Transmitter](image)

This logic ‘0’ is a start bit which is sampled in the mid of the bit duration to validate its logic states. On validation the next 8 bits from LSB to MSB are stored in an internal register followed by a stop bit at logic ‘1’. All the data bits are sampled at the mid of the bit duration. On receiving the correct stop bit, a ‘Character Valid’ signal is generated otherwise the character is discarded. The procedure is repeated for complete data from the Tesla Meter and whole data is stored in an internal register. On storing the data, a ‘data valid’ signal is generated which is used by the protocol conversion unit to move from one unit to another.
3.4 RS485 Transmitter

RS485 Transmitter has two important signals, the ‘data enable’ and ‘Serial data out’ signals. The RS485 is a multi-drop interface in which many slaves can communicate with each other over a single bus. All slaves can receive data simultaneously but only one slave can transmit data and all other must drive their transmitter output in high impedance.
A ‘data enable’ is generated within the module which is used to enable different driver ICs whenever ECM is required to send the data over RS485 interface. An Acknowledgement frame is sent when Command frame received is either correct or incorrect and a Response frame is sent when Command frame received is correct. The Acknowledgement and Response frames are stored in an internal register array. The Acknowledgement frame consists of various fields in which the address of the source and destination, type of command and type of acknowledgement are given. The Response frame contains address of the source and destination along with data from Tesla Meter and subsystem’s status information.

An Acknowledgement frame is sent when Command frame received is either correct or incorrect and a Response frame is sent when Command frame received is correct. The Acknowledgement and Response frames are stored in an internal register array. The Acknowledgement frame consists of various fields in which the address of the source and destination, type of command and type of acknowledgement are given. The Response frame contains address of the source and destination along with data from Tesla Meter and subsystem’s status information.

4. RS232 to RS485 Protocol Converter

The Protocol Converter is fed by the RS232 Controller which inputs the Magnetic Field value. The value is written into the temporary buffer of RS485 transmitter continuously till transmission from the RS485 Transmitter is not active.

IV. Conclusion

RS232 & RS485 controllers are designed, simulated and implemented in FPGA with the help of Verilog HDL. The digital design for controllers is synthesized in terms of logic resource of the Spartan FPGA. The modules are designed to work at a baud rate of 19.2 kbps. This is verified through implementation and controllers are working successfully with Tesla Meter and Operator Console. It is noted that for complete communication at RS485 interface, it takes around 60 ms from reception of command to the transmission of response. This implies that data from 15 subsystems can be updated at this baud rate in 1s. Thus it may be concluded that this protocol conversion is useful for Microtron like systems where data update rate is of the order of a second and number of sub-systems are less than 15. The circular buffer is also designed and simulated using ModelSim XE & Xilinx ISE 8.2i. The implementation part will be completed in the near future.

Reference

[1]. Puneet Maheshwari, Yogendra Sheth, Pravin Fatnani, C.P. Navathe. RRCAT, FPGA based Control System Hardware for Microtron, DAE-BRNS Indian Particle Accelerator Conference, ©2013, Kolkata, West Bengal, India.
[7]. http://www.asciitable.com/