ARM7TDMI based Low Cost Portable Serial Asynchronous Protocol Analyzer

Manoj Kumar Meena
PEC University of Technology
Sec-12, Chandigarh, India

Neelam Rup Prakash
PEC University of Technology
Sec-12, Chandigarh, India

ABSTRACT
This brief proposes a novel low–cost serial asynchronous communication protocol analyzer based on ARM7TDMI micro controller. The paper also suggests the hardware configuration & software algorithms to implement such an embedded system. The ARM-7 microcontroller, a 32-bit architecture along with a full fledge 16C550 UART, high speed configurable timers, number of power saving modes & efficient algorithms explained in the paper, becomes very good choice for implementing a portable serial link debugging tool. The system compared to basic analysis tool like oscilloscope & logic analyzer, will not be able to detect voltage levels but will certainly be able to extract timing parameters. The system will be able to detect serial parameters like baud rate, data bits & parity at data layer & refresh rate, ideal time, transmission time, message format at application layer. This system unlike the traditional protocol analyzers can be made very light, compact, low power, low cost & rugged, making it suitable for field operations. The protocol analyzer proposed here has been thought off keeping in view the resource lacking & not so friendly debugging environment of most of the military systems operating in the field.

Keywords
ARM-7, Protocol Analyzer, RS-232, RS-422, Serial Asynchronous Communications, UART.

1. INTRODUCTION
Serial communications [1] introduced in 1960 as RS-232C standard for connecting devices like mouse and Keyboard to computer, is still in very much use for communication in embedded systems. It owes its popularity to the small hardware required & almost no protocol stack, for making a working communication link. Also the serialization of data for transmission helps in reducing the clock skew & cross talk [2] problems which are inherent in parallel communication links. As a result the serial links can be clocked considerably higher than parallel links for achieving higher data rates over longer distances extending up to hundreds of meters. Though most of the modern communications has shifted to high speed Ethernet, USB etc. for exchanging large volumes of data, serial asynchronous communications is still used for low cost communication interfaces in embedded systems. Further the industrial applications & legacy systems in military especially navy, use serial interfaces for real time data exchange between different weapon systems.

The serial communication used can be of asynchronous or synchronous type. Asynchronous communication synchronizes for every byte and hence uses more overheads. Synchronous serial communication or more commonly known as Synchronous Data Link Control (SDLC) [1], [15] is able to achieve higher data rates than asynchronous as the synchronization is achieved over frames rather than individual bytes. However SDLC requires extra lines for clock & as a result is less popular for most of the moderate speed, small data exchange serial links. The emphasis of this paper is on asynchronous serial links.

Monitoring serial communications is necessary in R&D phase for creating dummy systems for simulating integrated communication environment, and during testing and debugging phases. The classical tools, such as digital oscilloscopes or logic analyzers offer information at the bit level like bit timings & voltage levels, but they lack the capability to provide extensive data layer & application layer analysis for serial interfaces. The required solution is to build a dedicated system for one or more serial protocols, using a high speed, low power, low cost micro controller for monitoring and testing the data exchange. The terms serial communication & serial asynchronous communication will be used interchangeably hereafter conveying the same meaning.

The remainder of this paper is organized as follows. Section 2 provides a brief tutorial about serial asynchronous communication & its parameters. The similar work is discussed in section 3 and the proposed hardware configuration along with a detailed functional block diagram is presented in Section 4. Section 5 presents software algorithms required for the acquisition, analysis & extraction of the serial parameters. Section 6 reports performances & section 7 outlines the conclusions & future scope.

2. SERIAL ASYNCHRONOUS COMMUNICATION
Typically, asynchronous serial communication [1]-[3] is used to transmit ASCII data. Since the data exchange is not periodic in asynchronous hence communication is completed using 3 transmission lines: (1) Ground, (2) Transmit, and (3) Receive. Using these separate lines, both transmission & reception can be done simultaneously. The serial port also known as Universal Asynchronous Receiver & Transmitter (UART), generally has additional lines for handshaking & modem signals but are generally not used. The important serial characteristics that two communicating sides must agree on for correct communication are baud rate, data bits, parity and stop bits [1]. In serial communication the least significant bit is transmitted first after the start bit, followed by the rest of the data bits, parity & stop bits. The asynchronous serial transmission format for Hex value 4DH with even parity is explained in figure 1.

The important serial asynchronous communication parameters [1]-[3] are explained as

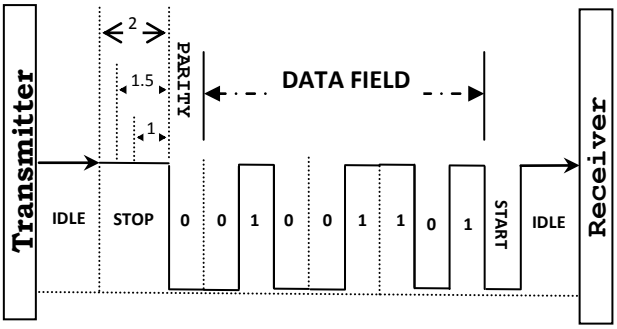


Fig.1 Asynchronous Byte Format

2.1 Baud Rate

It is the number of possible events, or data transitions, per second. It is a speed measurement for communication. Baud rate in case of serial communication is also called bits per second (bps) rate. Common baud rates for telephone lines are 14400, 28800, and 33600. Baud rates like 9600, 19200, 38400, 57600 etc. are frequently used in industrial applications & military systems for exchanging data. Baud rates greater than these are possible, but these rates reduce the distance by which devices can be separated. These high baud rates are used for device communication where the devices are located near one another.

2.2 Start Bit

It is used to signal the start of the communication for every byte or character transmitted. This bit has the same period time as the other bits and is used for synchronization between the two communicating devices. This bit is also called space (low) as it goes low from the default mark (high) state on the data line.

2.3 Data Bits

They are the measurement of the actual data bits in a transmission. When the computer sends a packet of information, the amount of actual data may not be a full 8 bits. Standard values for the data packets are 5, 6, 7 or 8 bits. The number of data bits depends on the size of the symbol set required for communication. For example, standard ASCII has values from 0 to 127and hence require 7 bits. Same way Extended ASCII uses 8 bits to represent 0 to 255 different symbols. Data bit size of 5 & 6 are rarely used in modern day communications as the symbol set they represent is very small.

2.4 Parity

It is a simple form of error checking that is used in serial communication. There are four types of parity: even, odd, marked, and spaced. The option of using no parity is also available. For even and odd parity, the serial port will set the parity bit (the last bit after the data bits) to a value to ensure that the transmission has either an even or odd number of logic high bits. For example, if the data to be transmitted is 1001, then for odd parity, the parity bit would be 1 to keep the number of logic high bits odd. Marked and spaced parity does not actually check the data bits, but simply sets the parity bit high for marked parity or low for spaced parity. Parity bit allows the receiving device to check if noise is corrupting the data or if the transmitting and receiving devices' clocks are out of sync. It is very useful in case the two devices communicating are not using highly stable clocks.

2.5 Stop Bits

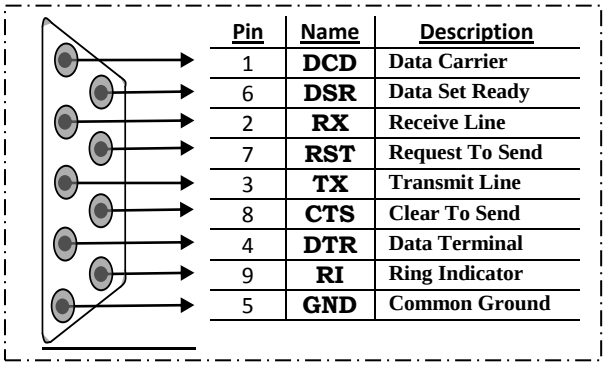
They are used to signal the end of communication for every byte. Typical values are 1, 1.5, and 2 bits. Since the data is clocked across the lines and each device has its own clock, it is possible for the two devices to become slightly out of sync. Therefore, the stop bits not only indicate the end transmission but also give the computers some room for error in the clock speeds. The number of stop bits used, depends on the responsiveness of the receiver device. Slower the receiver, higher the number of stop bits. Further time for processing can be provided by inserting inter byte gap, which is decided at application layer.

2.6 Flow Control

Flow control implies harmonious communication between a slow receiving device like printer & fast transmitting device like a computer. It helps in controlling the amount of data exchanged and the timing between the two communicating devices. It is done by using actual hardware lines. Similar to TX and RX lines, the Request To Send / Clear To Send (RTS/CTS) and Data Terminal Ready/Data Set Ready (DTR/DSR) lines work together with one being the output and the other the input as handshake signals. When a receiver is ready for data, it will assert the RTS line indicating it is ready to receive data. This is then read by the sender at the CTS input, indicating it is clear to send the data. The next set of lines is Data terminal Ready (DTR) and Data Set Ready (Data Set Ready). These lines are used mainly for communication between modems for status indication.

2.7 Electrical Interfaces

RS-232 (ANSI/EIA-232 Standard) [2], [3] is both an electrical & physical recommended standard given by Electronics Industry Association. It is commonly found on IBM-compatible PCs. It uses single ended transmission where in a common ground line between receiver & transmitter is used for communicating 1's & 0's. The commonly used DB9S connector for rs-232 in computers is explained in figure 2.



Pin	Name	Description
1	DCD	Data Carrier
6	DSR	Data Set Ready
2	RX	Receive Line
7	RST	Request To Send
3	TX	Transmit Line
8	CTS	Clear To Send
4	DTR	Data Terminal
9	RI	Ring Indicator
5	GND	Common Ground

Fig. 2 D-Type 9-Pin RS-232 Connector

RS-232 is limited to point-to-point connections between PC serial ports and devices. RS-232 hardware can be used for serial communication up to distances of 20 meters.

RS-422 (EIA RS-422-A Standard) [3] is the recommended electrical serial communication standard used on Apple Macintosh computers. RS-422 uses a differential electrical signal, as opposed to unbalanced signals referenced to ground in RS-232. Differential transmission, which uses two lines, each for transmit and receive signals, results in greater noise immunity and longer distances as compared to the RS-232. The distance over which communication can be done using

RS-422 interface is 1200 meters. These advantages make RS-422 a better fit for industrial and military applications.

2.8 Encoding Schemes

Serial asynchronous communication uses a non-return-

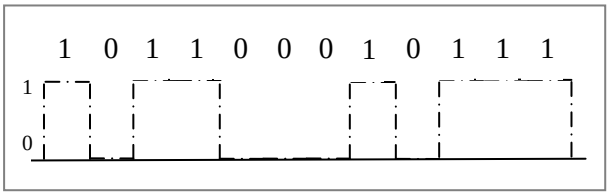


Fig.3 NRZ Encoding Scheme

to-zero (NRZ) line code [1], [3]. The NRZ is explained in fig 3. In this 1's are represented by one significant condition (usually a positive voltage) and 0's are represented by some other significant condition (usually a negative voltage), with no other neutral or rest condition.

This method wastes power, as DC level is maintained even when there is no information transmitted. Further the synchronization is lost if there is a long series of one or zeros. However due to its simplicity of implementation it is popularly used in RS-232 serial communication.

3. RELATED WORK

There is a lot of asynchronous communication monitoring tools available in the market but all of them are costly & bulky, PC based applications. Further they mostly run windows OS and hence does not give time parameters of the captured data correctly. Few of them which have scored well on the ease of operation & decent data analysis features are explained next.

232Analyzer [4] and Packet Sniffer [5] are two very popular analyzing tools for RS-232, 422 and 485 type of electrical interfaces. They offer good data capture, analysis & logging capabilities. These tools come with good Graphical User Interfaces & require additional RS-232/422 convertors for data acquisition.

The tools provide support for most of the windows OS.
Reference [6] presents SI Scope RS232 Analyzer. It comes with a dual port Passive Monitoring Cable for simultaneous capturing serial data & forwarding it to the receiving end. Its interrupt driven design helps in maintaining the correct sequence of events and the time between the events is measured accurately.

Serial Port Monitor is another classical monitoring tool for RS232 bus [7]. It runs on Windows-7/9x/XP/NT/2000 platforms and displays the messages captured from the serial link in easy to read format like table view, line view, dump view & terminal view.

MSB-RS232 analyzer [8] offers all necessary features for an effective examination of RS232 connections. It uses a dedicated hardware for the Simultaneous sampling of all lines, data analysis in real time, detection of invalid line levels, detection of asynchronous or drifting baud rates between sender and receiver. This tool provides interesting features but again requires a PC for the analysis.

All these tools are purely software tools except MSB-RS232 analyzer. They require the use of a personal computer or laptop for the analysis & a separate hardware for handling various electrical interfaces. They are not just costly, power inefficient & bulky but also not able to provide real time data as they all depend on OS (mostly standard windows) for time stamping, which is not very reliable.

These application software are good for user friendly lab environment but does not serve an engineer working on industrial and military systems in the field.

The system proposed here is a dedicated embedded system running algorithms written in assembly & embedded C [1], [9] for real time requirements. Also the serial data & results of analysis can be shown using a character LCD. The different modes of operations can be easily selected using a passive matrix keypad. All this does away with the requirement of a full-fledged computer making it a highly compact, light & portable.

4. PROPOSED HARDWARE

The proposed hardware for building serial synchronous analyzer, uses ARM-7TDMI [9] as the main building

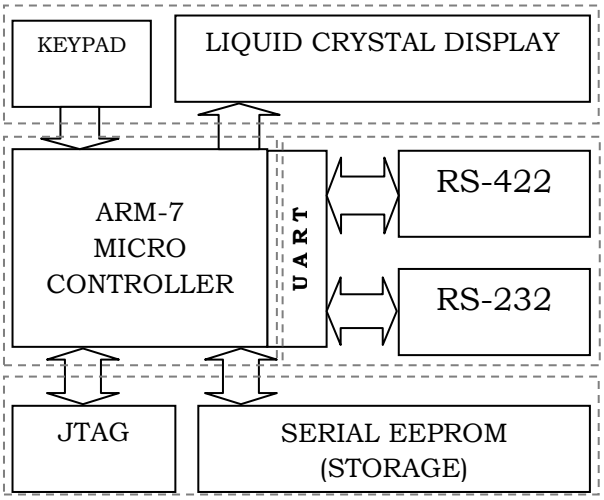


Fig.4 Functional Block Diagram

block & RS-232/422 drivers, character LCD, matrix keypad, Electrically Erasable Programmable Read Only Memory (EEPROM), JTAG port as supporting modules. The system can be functionally divided into four sections as explained in figure 4.

4.1 Data Acquisition Section

The data acquisition section is responsible for acquiring the serial parameters & data through oversampling of line transitions [5]. Both RS-232 & RS-422 electrical interfaces can be handled using appropriate line driver ICs like MAX-232, AM26LS31/32 [1], [3] etc. These line drivers not only protect the microcontroller but also convert the RS-232/422 signals into TTL signals. The selection between the types of interface to be handled is done using a two way switch.

4.2 Data Analysis Section

It is responsible for the extraction of required information like bit timings, serial parameters & application layer parameters from the byte or frame received. The general purpose input output (GPIO) pins of microcontroller along with the timer, interrupt & UART peripheral is able to do this under software control. A timer configured for nanosecond resolution is required for the determination of commonly used baud rates & refresh rates.

4.3 Data Input & Display Section

This section handles the displaying of the raw/processed data & results as selected by the user. Basically the user is provided with a number of options to select. The user interface is provided through a character LCD & a matrix keypad. The LCD is provided with an optional backlit for night operations & saving energy during day time. This section consumes most of the power in the system. A passive matrix keypad can be used to select different mode of operations and navigation through stored data.

4.4 Offline Storage & Program Download Section

An EEPROM of suitable size 16/32/64K can be used for data storage. This huge stored data can be later on used for offline analysis. The good size of static ram (16kB) available in most of the ARM-7 based microcontrollers helps in storing large continuous raw serial data without losing any. The ARM-7 JTAG port has been provided for convenient & easy downloading of the updated or modified code into the system.

5. ALGORITHMS

The various algorithms for detecting various serial asynchronous communication parameters [10]-[14] both at data layer & application layer are as follow

5.1 Baud Rate Detection

Baud rate detection requires a high resolution timer having resolution of the order of few hundred nanoseconds. This is required because the common range of the baud rates for asynchronous serial communication is 150-250000bps i.e. on time scale it is 6.67ms to 4µs respectively. Initially the serial RX pin is to be used as a port pin for detecting the baud rate. An external interrupt pin is shorted to RX pin for detecting & reporting transitions on interrupt basis. The line is sampled at a rate at least 8 to 10 times higher than the baud rate being detected [10], [11], [12]. The timer is used for counting the no of nanoseconds passed between the two consecutive opposite transitions. The timer is started at the detection of the first high to low or low to high transition. It is stopped on detecting the next opposite transition. The time interval is stored & compared with subsequent readings for the minimum delay. The sampling is done over a good amount of time say 1 second for a baud rate of 9600. Assuming a variable data transmission & evenly spread 1's & 0's, the user is expected to get sufficient transitions for the detection of baud rate. The baud rate is calculated by taking reciprocal of the minimum time interval detected. For example in figure 5, if Δt is the smallest time detected over 100 readings then the baud rate is calculated as $1/\Delta t$.

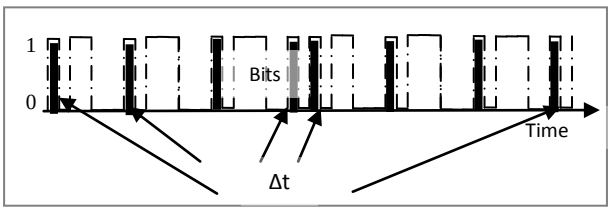


Fig.5 smallest bit intervals

Time to detect the baud rate depends both on the refresh rate of data, no of samples taken and the no of repetitions of the least time interval.

5.2 Data Bits Detection

Once the baud rate is detected with certainty the data bits can be determined. This requires an iterative process of assuming data bits, configuring Universal Asynchronous UART and then checking it against the parity & frame errors [11], [13], detected automatically by the UART. For this, various possible combinations like 1-8-N-1, 1-7-N-1 etc. are tried one by one for the correct match. If a match is found the same is done for number of times to make sure the reading was correct. This process being iterative in nature & due to the requirement of multiple readings for accuracy is very time consuming. This method is not so successful in case the data is corrupted by noise.

5.3 Parity Detection

Parity is detected more or less in the same fashion as data bits. The UART is configured for an assumed parity & the parity flag is checked for a correct match. However this process takes very little time as the combinations to be checked are only 4, i.e. even, odd, mark & space. Again it is assumed that there is no noise on the line.

5.4 Stop Bits Detection

Once the baud rate, data bits & parity is known the stop bits can be checked by using the time between last character received & start of the next character by using a timer. The reading is confirmed by taking multiple readings as the inter byte gap may wrongly be interpreted as stop bits.

5.5 Idle Time Detection

The idle time is mostly the largest time interval when there is no activity on the RX line. This time is sufficiently large

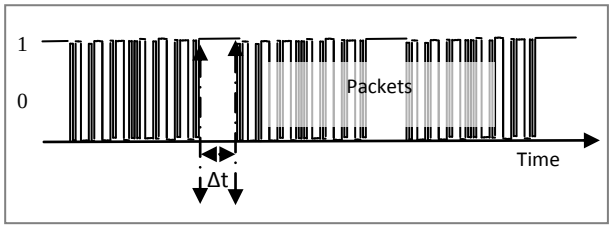


Fig.6 Idle Time Detection

As compared to inter byte gap. As a result the two can be differentiated easily. Again timer is used over a no of times to determine a precise value. Idle time is used to calculate the refresh rate & tolerance of the received packet. The idle time as shown in figure 6 is denoted by Δt.

5.6 Refresh Rate Detection

The packet refresh rate, an application layer parameter, can be determined, once the serial parameters & idle time are detected correctly. The time interval between the two consecutive idle times is taken as refresh rate. However this sometimes is unreliable as idle time is very short. Hence a second method i.e. time interval between two consecutive header field or footer field (explained later) is sometimes used for calculating refresh rate.

Fig.7 Refresh Rate Detection

The refresh rate as shown in figure 7 can be calculated as Refresh rate = (1/Δt) Hz

5.7 Transmission Time

The transmission time is the time taken by the frame to get transmitted fully. This is calculated by subtracting the idle time from the refresh rate. The idle time as shown in figure 8 is denoted by Δt.

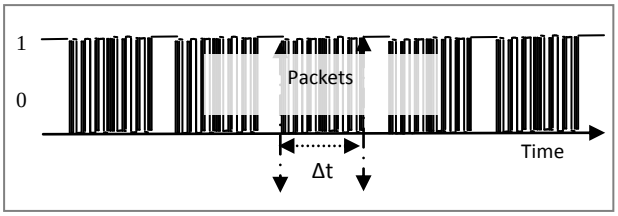


Fig.8 Transmission Time Detection

5.8 Packet Header & Footer Detection

The Byte or Bytes repeated for every packet, after the end of the idle time on RX line are called Packet header bytes or header field. Packet header is important information which tells the position of particular information in the packet. Similarly the Byte or Bytes repeated before the start of the idle time on RX line are called Packet Footer bytes. Like header field this field also should be consistent for every packet or frame captured.

5.9 Packet or Frame Size Detection

The header, footer & no of bytes in between the two are called packet size. The packet size is sometimes required for checking the data integrity at packet level. Further it helps in calculating the storage memory requirements in case of large data & high speed serial link.

5.10 Polarity Detection for RS-232 & RS-422 Interface

The polarity of the RS-232 interface needs to be checked in case the RX & TX points of a system are not clear. It can be done by checking the status of the TX line of the transmitter. If the polarity is correct then the RX line of the receiver is logic high (pulled high), for idle or inactive time. This is done assuming there is nothing wrong with the TX & RX lines electrically.

5.11 Polarity Detection for RS-422 Interface

The RS-422 serial link requires both RX, TX & their '+', '-' ends to be detected for correct communication. It is done in a manner similar to RS-232 polarity detection.

In fact the polarity detection is done first to be able to determine other parameters explained above correctly. The coding done in embedded C and assembly [9] helps in achieving real time requirements.

6. PERFORMANCES

The proposed system was implemented using LPC2129 [15] ARM-7TDMI micro controller. The system automatically detected baud rates in the range of 150 to 250000bps with 95% accuracy at highest baud rate. The data bits generally take longer long time as a number of combinations that needs be tried out for arriving at correct decision are large. The success rate for the data bits was around 95%. Other serial parameters like parity, stop bits & line polarity were detected correctly all the times. The application layer parameters like refresh rate, idle time, transmission time, header & footer parameters had 100% accuracy. The total code size (after compilation), written in embedded C & inline assembly was less than 10kbytes. The RAM utilized, mainly for storing captured raw data & extracted serial parameters, was under 1.5kbytes.

The MMI provided using a 20x4 character LCD & a passive 4x4 matrix keypad was able to meet all user requirements. The user could easily understand & select required mode of operation. Further the results of the analysis like serial parameters & raw data (both in ASCII & Hex format) were displayed on LCD in 80character page format. Both forward & backward navigation between pages was possible using the keypad. The data acquisition was implemented using max3223 & 26LS31/32 line drivers for RS-232 & RS-422 interfaces respectively. Though in application programming, is available in LPC2129 micro controller for storage, yet 24LC256 EEPROM was provided for additional storage & data portability. The system was also provided with a standard 20-pin JTAG port & serial port programming for easy code modification. The system used only UART, GPIO, interrupts & timer peripherals to achieve all the required features.

The backlit of the LCD was provided with a switch & most of the line drivers were enabled under LPC2129 control, making it highly power efficient. The total power consumption for the system was less than 1750mW for worst case. Therefore a standard 5V, 1500mAh battery can support it for 2 to 3 hours.

Most of the components being IC, added very little to the weight of the system. The main components that added to the weight of the system were LCD, keypad, battery & the enclosure. However the system was still less than 1 Kg, which is not bad for a rugged, portable test equipment of this type. The system was easily assembled on a PCB of 15x10cm size. The system because of its small size & light weight can be easily set up, held in hand & can be moved around. Further the system was provided with all the required connectors & convertors on the same PCB, hence the system was very compact making it ideal for portability.

The system in addition to above features also included ASCII set, Hex to ASCII & vice versa conversion, XOR, Modulo-8/16, CRC-16 & CRC-CCITT [14] calculations for the specified string.

7. CONCLUSION & FUTURE SCOPE

A prototype of the proposed system was developed specifically for Indian Naval requirements. The system realized was able to meet all the requirements of a system integrator, serial application programmer, device driver developer & an engineer involved in reverse engineering.
Future scope for the system may include

- Extending auto baud rate detection to 1mbps
- Inclusion of SDLC analysis
- Increasing power efficiency for enhanced battery life
- Reducing system weight further for ultra portability
- Traffic generation & simulation capabilities

8. REFERENCES

[1] Serial Port Complete, 2nd Ed. by Jan Axelson.

[2] Texas instruments, "Interface Circuits for TIA/EIA-232-F", SLLS037A, September 2002.

[3] Frank Dehmelt, Matthias Feulner, Carmen Gonzalez, Michael, Groenebaum, Firoj Kabir, Arek Kacprzak, Clark Kinnaird, Johann Zipperer, "Comparing Bus Solutions (High Performance Linear Interface)", Application Report-SLLA067A-MARCH2000-Revised February 2004, Texas Instruments, 2004. 2011.

[4] CommFront Knowledge Base and FAQs - 232Analyzer, available at: http://www.commfront.com/232analyzer-faq.htm

[5] Serialtest®: Asynchronous RS-232/422/485 Serial Protocol Analyzer and Packet Sniffer, available at: http://www.fte.com/products/serialanalyzers-RS232.aspx

[6] New UltraTap RS232 Analyzer Cable, Available at: http://www.sinnovations.com/htdocs/siscope_rs232_analyzer.htm

[7] Serial Port Monitor - Eltima Software, Available at: http://www.eltima.com/products/serial-port-monitor/

[8] IFTOOLS: https://iftools.com/analyzer/msb-rs232/index.en.php

[9] ARM System Developer's Guide Designing and Optimizing System Software by A. N. Sloss, D. Symes, C. Wright, J. Rayfield, Elsevier 2004.

[10] Yongjian Tang, Lenian He and Xiaolang Yan, "A novel data processing in high speed serial communication", IEEE, pp. 1228-1231, 2005.

[11] Li Pang, Houde Liu, Baohua Li and Bin Liang, "A Data Recovery Method for High Speed Serial Communication based on FPGA", IEEE, pp. 664-667, 2010.

[12] M. Popa, A.S. Popa, V. Cretu and M. Micea, "Monitoring Serial Communications in Microcontroller Based Embedded Systems", IEEE, pp. 56-61, 2006.

[13] Jie Liang and Ran Duan, "Design of the Embedded Serial and Parallel Communication Protocol Controller", IEEE, pp. 436-439, 2010.

[14] Wang Qi, Fan Jianwei, CUI Wei and Yang Duwei, "Design of sdlc synchronous serial communication based on intel 8274", IEEE, pp. 315-317, 2008.

[15] LPC2119/2129/2194/2292/2294 USER MANUAL, http://www.nxp.com/technical-support-portal/50809/45994/user-manuals.