ISTITUTO NAZIONALE DI ASTROFISICA
NATIONAL INSTITUTE FOR ASTROPHYSICS

Relazione tecnica sistema di comunicazione multiprotocollo con FTDI UM232h

Mattia Corpora, Pierluca Sangiorgi, Giuseppe Sottile

A Multi-protocol Communication System For Computers or Embedded Computer Systems

Corpora M., Sangiorgi P., Sottile G.

INAF - Istituto nazionale di Astrofisica
Via Ugo La Malfa 153, Palermo, Italy

Abstract

In this paper a system that allows a computer or an embedded system to easily communicate with devices which implement the usual standard communication protocols is proposed. The system consists of a hardware device and a software application. The hardware component is based on the FTDI FT232H chip, a single channel USB to serial/parallel port converter which can be configured in a variety of industrial standard serial or parallel interfaces. The software application manages the hardware component and provides a user friendly graphical user interface which allows the customers to choose the desired protocol, to configure some protocol and message parameters, and to communicate with their devices. Therefore, the system can be used without knowing programming languages and/or protocol characteristics.

Keywords: Serial communication, UART, SPI, I2C, multiprotocol, control software.

1. Introduction

Communications between computers or embedded systems and hardware devices through different physical channels and protocols are necessary in almost all simple and complex electronic systems [1]. The hardware devices producers can provide different solutions according to customers target [2]. The usual solutions provide devices with only a proprietary software or a proprietary software and/or the explanation of the protocol characteristics. Moreover, the users can

develop hardware device and protocol ad hoc [3].

So, communication with hardware devices requires an electrical connection which are direct or through an additional converter interface [4] [5]. The two scenarios are shown in Fig. 1.

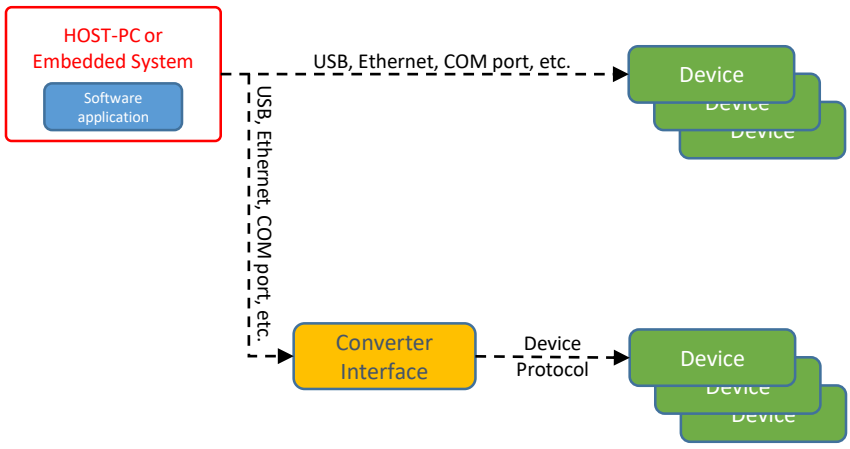

Figure 1: Communication scheme with/without interface converter

If the producers provide the devices with only a proprietary software, the communication criteria are fixed and the users can't integrate the provided software in their system software. If the producers provide also the explanation of the protocol characteristics, users can implement their own software to be integrated with other application software. It is clear that if producers don't provide a proprietary software, users must have some specific skills to communicate with the devices. In particular, users must know programming languages, protocol characteristics and how the possible converter interface works. There are some converter interfaces, as FTDI UM232H [6], that implement more standard communication protocols. FTDI UM232H is a multi-protocol converter equipped with a software library but without a software application suitable for immediate and easy use. In fact, to use the converter, users must know: the converter in itself, the software library provided by FTDI, the C/C++ pro-

gramming language and the protocols characteristics. The present work offers a system of a hardware component and a software component able to avoid the issues explained providing users with an easy-to-use tool for communications with devices that employ some standard protocols. The proposed system, of which a diagram is shown in Fig. 2, can achieve UART (RS232, RS422 and RS485), SPI, I2C and FIFO245 (synchronous and asynchronous) protocols.

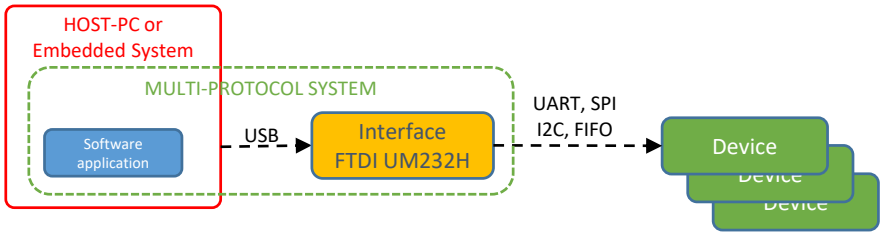

Figure 2: Scheme of the proposed system.

2. Proposed system

2.1. System practical applications

The Multi-protocol system allows the users to:

- communicate with hardware devices whose protocol is explained by producers;
- equip their own hardware systems with a communication port and a communication protocol;
- integrate software component for management of hardware subsystem devices into own control software system;
- test devices that implement standard communication protocol

. The most important advantage of the proposed system is that the users can handle it without having particular knowledge on programming language and/or on the protocol structures.

2.2. FTDI hardware and software component

The proposed system employs, as hardware component, the FTDI UM232H, a single channel USB 2.0 Hi-speed to UART/FIFO interface converter. It can be configured in a variety of industrial standard serial or parallel interfaces and has the following advance features:

- entire USB protocol handled on the chip. No USB specific firmware programming required;
- USB 2.0 Hi-Speed (480Mbits/Second) and Full Speed (12Mbits/Second) compatible;
- Asynchronous serial UART interface option with full hardware handshaking and modem interface signals [7];
- Multi-Protocol Synchronous Serial Engine (MPSSE) to simplify synchronous serial protocol (USB to JTAG, I2C, SPI (MASTER) or bit-bang) design [8];
- USB to asynchronous and synchronous 245 parallel FIFO [9] ;

FTDI UM232H board is shown in Fig. 3.

Figure 3: FTDI UM232H Board

FTDI provides two alternative software interfaces for the converter management. One interface consists of a Virtual COM Port (VCP), which is available to the system as a legacy COM port. The second interface, called D2XX [10],

is provided via a proprietary DLL. The D2XX interface has special functions that are not available in standard operating system COM port APIs, like SPI and I2C functions. Some protocols can be chosen by programming a EEPROM, while others can be selected by sending commands to the interface in run-time. To program the memory, FTDI provides a software application called "FT PROG" [11], which allows the user to select the protocol and its relative parameters. This proprietary application is only Microsoft Windows operating system compatible and does not enable changes between protocols in run-time. Another way to program the EEPROM is to use the features of the D2XX, but it is necessary to know the memory structure which is not provided by FTDI group. To send/receive data to/from the devices connected to interface using a protocol, D2XX library must be used. Therefore, users must know both the programming language and the characteristics of the protocols, or they must contact specialists to request custom solutions. In summary, the D2XX library can only be used by advanced software developers, so that, in general, the use of the interface is complex. Moreover, due to the low level of functions, software developers must learn deeply how the UM232H interface works.

2.3. Proposed software

The purpose of the proposed software is to provide an application for the host-PC equipped with a user-friendly GUI, to easily manage the FTDI UM232H board as a multi-protocol converter and to easily communicate with hardware devices. The application is developed in order to satisfy the follow functionalities:

- to manage the UM232H interface converter by hiding the low-level implementation from the users;
- to communicate with hardware devices by providing the user with an interactive console (*interactive mode*);
- to manage a complex interaction to monitor and control hardware devices executing multiple communications managed by proper script stored in

file with definition of the communications parameters and the message data (*script mode*);

- to provide the users with a tool able to create the script mode files;
- to store the communication logs.

The Fig. 4 shows the application use case diagram.

Manage converter use case includes all operations for the converter man-

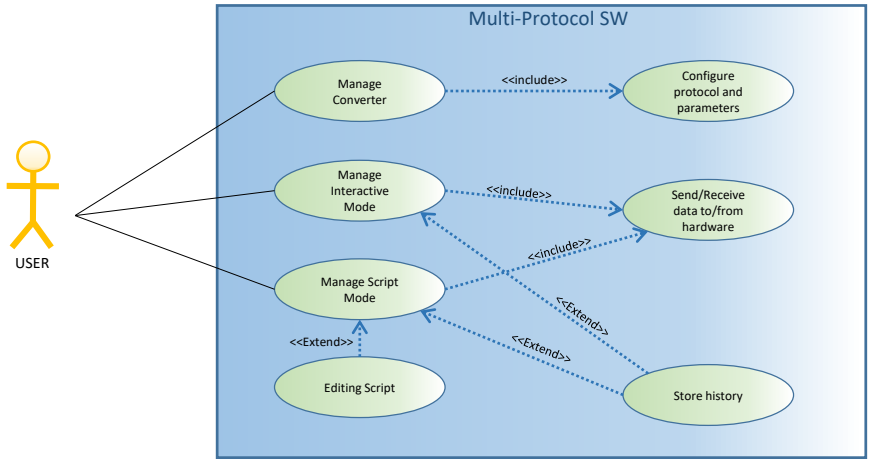

Figure 4: Application use case diagram

agement: programming the EEPROM according to the chosen protocol, management the commands for sending and receiving data and configuration of the protocol parameters in according to the users' choices. *Manage interactive mode* use case provides the users with a interactive console in order to communicate with hardware devices. In this way, the users can use the keyboard to digit the message to be sent and read the devices response on the screen. The users can also choose the data format and can store the communication history. *Manage script mode* use case allows the users to manage complex interaction in order to allows *monitor* and *control* operations of hardware devices. The monitor operations consist of a number of ordered messages to be periodically sent/received in order to monitor the hardware devices status. The users can choose when

monitor operations start or finish. The control operations consist of a number of ordered messages to be sent/received asynchronously in order to control the hardware devices. The users can decide when the control actions must be run. In order to satisfy this functionality we adopted a proper structured description of the multiple operations to be executed to realize the communications with the hardware devices (Fig. 5).

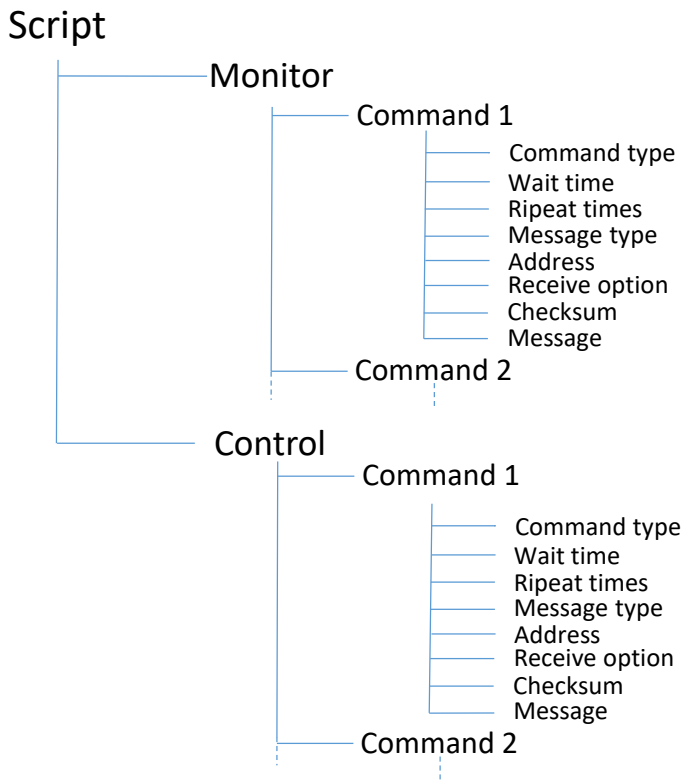

Figure 5: Data Script structure

Implementation of this structure is made by the use of the data interchange standard JSON. This choice also allows to store this structured data in proper JSON file format. The script files contain the commands, the messages and the

parameters for the monitoring process and the control process. The advance users, which know the file structure, can create the script files by a simple text editor. However, the system provides a simple graphic tool to create the script files without knowing syntax and structure (*Editing Script* use case). *Store communication history* use case allows users to save communication logs both in script mode and in interactive mode. The communication logs contain the data message , the command type and the timestamps.

The proposed software is a multiplatform application which is developed in Java language, taking advantage of a third-party Java library called "YAD2XX" [12], a Java Native binding of D2XX. This guarantees a cross-platform application working on Windows, Linux and macOS.

3. Application GUI description

As shown in Fig. 6, the application GUI consists of four sections: *Interface Configuration*, *Protocol Configuration*, *Message Configuration* and *Communication Console*. The Interface configuration section allows the users to select a specific converter and to select the desired protocol. Moreover, it guides the users in the process of wiring by a proper image that show the connection schema. The protocols configuration section dynamically changes according to the chosen protocol. The configuration protocol relative to the UART mode is shown in Fig. 6. This section allows the user to set the protocol parameters (like clock frequency, baudrate, parity, MTU, etc.) and to start and stop the connection. The communication section consists of two tab: *Interactive mode* and *Script mode*. The *Interactive mode* tab allows the user to communicate with the devices through an easy-to-use console. It is the same for all the protocols and it is equipped with two text fields: in the first one the user can directly write the messages to send to the devices, while the second one shows the responses and the communication history. Two buttons allow the user to send and receive messages, respectively, while a third and a fourth button clear the history or save it into a CSV file. If the Interactive mode is chosen, the message parame-

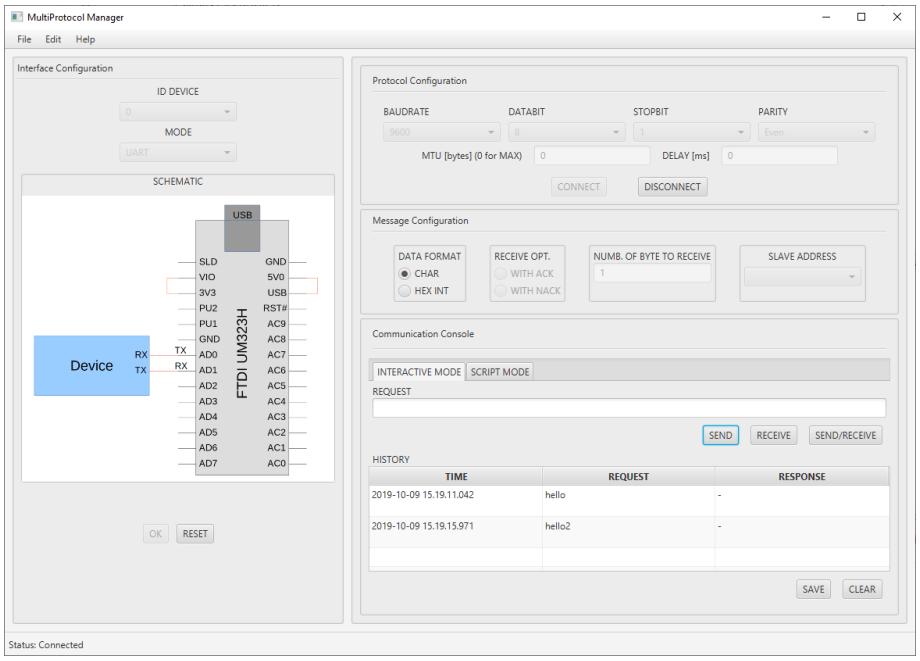

Figure 6: Application gui

ters must be configured through the message configuration section. In this way, the user can choose the data format (alphanumeric characters or two-digit hexadecimal integers) and the type of response from the devices (with or without acknowledge) within I2C mode. In addition, there is a further text box, where the user can specify the number of bytes to receive (if necessary for a specific message protocol implementation), and a combo box to select the addresses of the slave devices, within the I2C mode.

The *Script Mode* tab allows the user to communicate with the devices by script files. The commands inserted in the script can be executed in two different process depending on the user's choice. The commands entered in the "Monitoring Process" are repeated periodically until the user stops the thread. The commands entered in the "Control Process" are executed only once after clicking on the "Run Control" button. The Control Process has priority over Monitoring Process. So if the first one is executed while the second one is running, it is

stopped. The execution of the Monitoring process resumes when the Control Process is terminated. The tables summarize the commands already entered and their respective characteristics. When the script is loaded and executed into the main GUI, the commands are shown in two different tables that show the type of command and the data exchanged. The tables can be exported in CSV files. The JSON script files can be loaded by clicking on the "Open" button.

If users do not want to write the JSON file themselves, they can use the script creation tool. By clicking on the "Create" button, a second window (Fig. 7) is displayed for the creation, loading and saving of the scripts.

The script creator window consists of some graphics elements that allow the

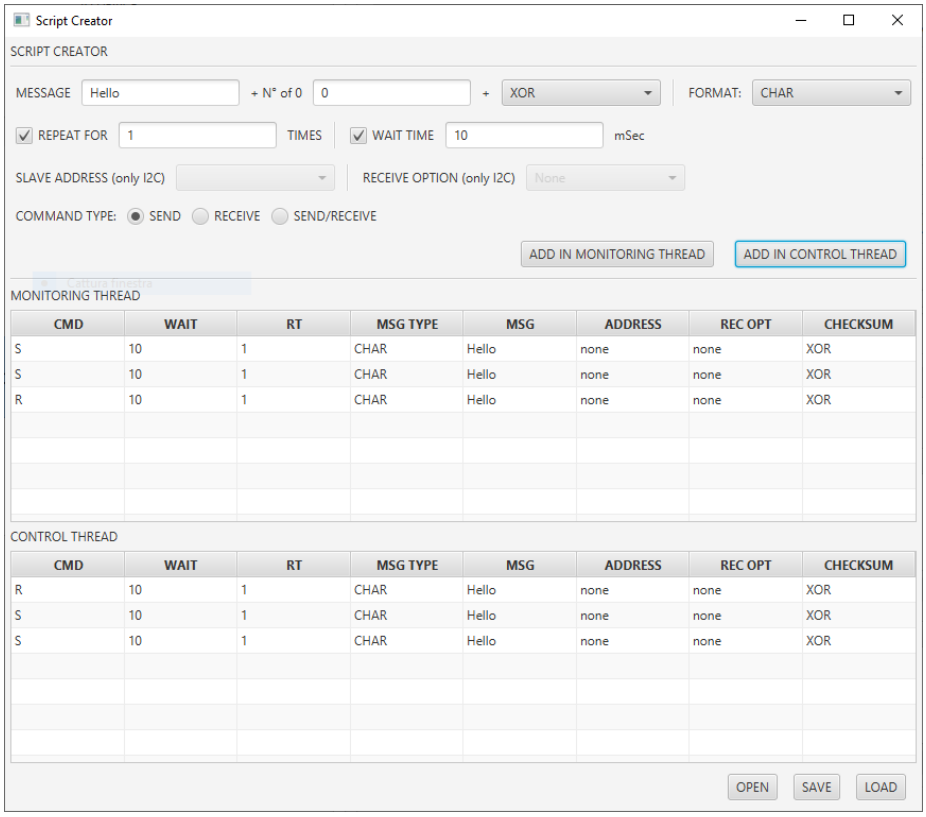

Figure 7: Script Creator Window

user to compose the messages, set data format and add, if necessary, any error checking data. By "Repeat for" text filed the user can specify how many times to repeat the message while by "Wait time" text field the user can specify the delay time (in millisecond) before the execution of the command. Once the command is defined, it will be executed by the monitor or control process according to the users choice.

4. Conclusion

According to all the features described above, it is clear that the combination of the proposed software with the FTDI interface UM232H could be a suitable solution for some of the most common scenarios, e.g. when the producers explain the communication protocol characteristics, regardless the hardware is provided or not with a proprietary software, or when the users need to create a new hardware and to equip it with a communication port and a communication protocol. [7] Overall, the proposed system has lots of significant advantages. First of all, the knowledge of programming languages, as well as the knowledge of the protocols, is not required to the users. Secondly, this system can be used as a universal communication tool in evaluating and testing any hardware that uses the available communication protocols. Lastly, the system provides a multiplatform and portable application.

References

[1] I. Maykiv, A. Stepanenko, D. Wobschall, R. Kochan, V. Kochan, A. Sachenko, Software–hardware method of serial interface controller implementation, Computer Standards Interfaces 34 (2012) 509 – 516. Intelligent DAQ's, Advanced Computing and Interfacing Systems.

[2] IEEE, Ieee draft standard for a smart transducer interface for sensors and actuators - serial point-to-point interface, IEEE P1451.2/D20, February 2011 (2012) 1–28.

[3] T. S, S. S, S. K, Fpga-based multi protocol data acquisition system with high speed usb interface, Lecture Notes in Engineering and Computer Science 2181 (2010).

[4] Shwetha S., Karunavathi R.K, The design of multiprotocol interface device, in: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 474–476. doi:10.1109/ICATCCT.2016.7912046.

[5] T. S, S. S, S. K, Fpga-based multi protocol data acquisition system with high speed usb interface, Lecture Notes in Engineering and Computer Science 2181 (2010).

[6] FTDI, Technical Report, 2019. https://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232H.pdf.

[7] FTDI, Technical Report, 2019. https://www.ftdichip.com/Support/Documents/AppNotes/AN_242_FTDI_UART_Terminal_User_Manual.pdf.

[8] FTDI, Technical Report, 2019. https://www.ftdichip.com/Support/Documents/AppNotes/AN_135_MPSSE_Basics.pdf.

[9] FTDI, Technical Report, 2019. https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_167_FIFO_Basics.pdf.

[10] FTDI, Technical Report, 2019. https://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide(FT_000071).pdf.

[11] FTDI, Technical Report, 2019. https://www.ftdichip.com/Support/Documents/AppNotes/AN_124_User_Guide_For_FT_PROG.pdf.

[12] S. Davies, Technical Report, 2019. https://sourceforge.net/p/yad2xx/wiki/Home/.