VOL. 13, NO. 2, JANUARY 2018 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

SERIAL DATA TRANSMISSION THROUGH THE MICRO-USB PORT OF THE SYSTEM STM32F407

Andrés Tovar, Dario Amaya and Olga Ramos
Virtual Application Group-GAV, Nueva Granada Military University-UMNG, Colombia
E-Mail: olga.ramos@unimilitar.edu.co

ABSTRACT

In this paper the results of the design of an algorithm for transmission data in the systemSTM32f407 to computer in C# will be presented. Initially, the hardware structure is showed, later it is evidenced the algorithm performed in IDE Keil and the Configuration Tool STM32CubeMX, in which it is described, libraries and development code. Besides, it is showed the architecture of the receive data algorithm on the PC, through the emulation of USB such as RS232 terminal. The main contribution in this researched is in the solution that is presented to communicate two system using the USB port emulating UART port.

Keywords: ACM, CDC, serial virtual, STM32F407, UART, USB-OTG.

1. INTRODUCTION

Nowadays, USB (Universal Serial Bus) communication is used such as interface to data transmission and energy distribution. Which it has been introduced in modern systems and consumer electronics to enhance the slow serial interfaces (RS-232) and parallel [1]. As well it has advantages like one single interfaces for several devices and automatic configuration. Nevertheless traditional systems and embedded platforms have serial ports such as UART, SPI, and I2C [2].

PC has been one of the devices that the USB communication has become an indispensable interface. For this reason, it is necessary an interconnection method between PC and UART device. The developments of peripherals based on USB are designed to be user friendly but it is not with the developers. That is the reason why a lot of manufactures and developers still used traditional ports. However, they have considered design USB to RS-232 converters to overcome this inconvenience, which are available in the market with different communication parameters, such as transmission speed, power consumption, cost and other [3].

Additionally, the communication between PC and embedded platforms can be development through software [4], performing a certain amount of protocol conversion between UART and USB serial ports [5]. Some applications have needed modernize and simplify their designs in terms of communication with computers or other devices. Agood example can be seen with Roland Szabó *et al*. [6] that developed in a Raspberry PI a control system of an intelligent robotic arm, it is controlled through a Serial port to USB, which solves not having to add other circuits to the system.

On the other hand, a lot of applications have used hardware to convert the USB communication to serial or vice versa. B.B. Shabarinath *et al* [7], who used it in the communication process, show in the HyperTerminal, connecting all units to PC COM ports. R. Kandilarov [8], who using a USB/UART converter and serial protocol, intended for communication in the current control of 4 leds through PC. In addition, the USB/serial converters can help to reduce and complement another kind of communication such as the example of stereo audio streaming via Visible Light [9], which uses USB to serial hardware in the communication architecture of the Visible Light, to facilitate the free space of the VLC channel.

2. METHODS AND MATERIALS

a. Hardware description

In the development of this project, the connection can be observed in Figure-1. In the embedded platform STM32F407 are highlighted the modules of an analogue digital conversion channel (ADC), digital output pins, the integrated STM32F407VG and the micro USB port that finally connects to the computer.

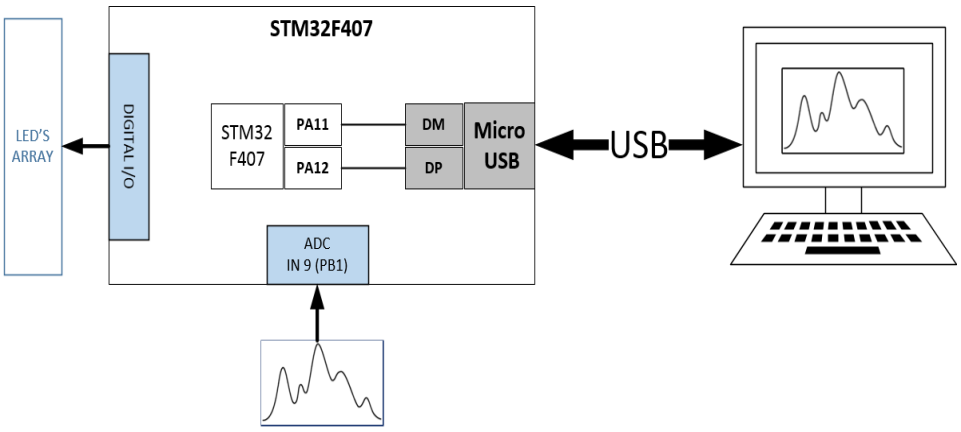

Figure-1. Device connection.

The micro-USB port architecture on the embedded system STM32F407, works as a USB OTG (on the go) port and is shown in Figure-2. It has a flip-chip EMIF02-USB03F2 in parallel which is a highly integrated array designed to suppress EMI / RFI noise for USB OTG ports. In addition, this filter includes ESD protection circuitry which prevents damage to the protected device when subjected to ESD surges up to 15 kV on external contacts.

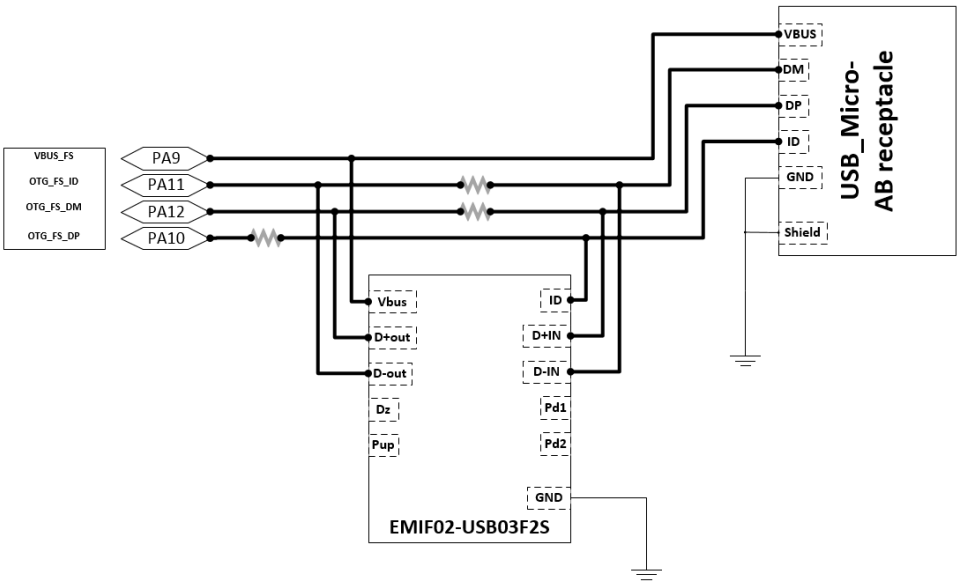

Figure-2. Connection of USB OTG FS OF STM32F407.

According to the manufacturer's specifications, this micro-USB port can be configured such as a virtual Serial port (VCP). In order to the serial port simulation to be recognized by the computer, a driver provided by STMicroelectronics is required. Using the STM32CubeMX software, which assist as a configuration tool for the embedded system STM32f407, it was configured in a general way, the micro-USB port as a VCP, the 8MHz physical crystal, an analog conversion input Digital, button and four user LEDs, as shown in Figure-3.

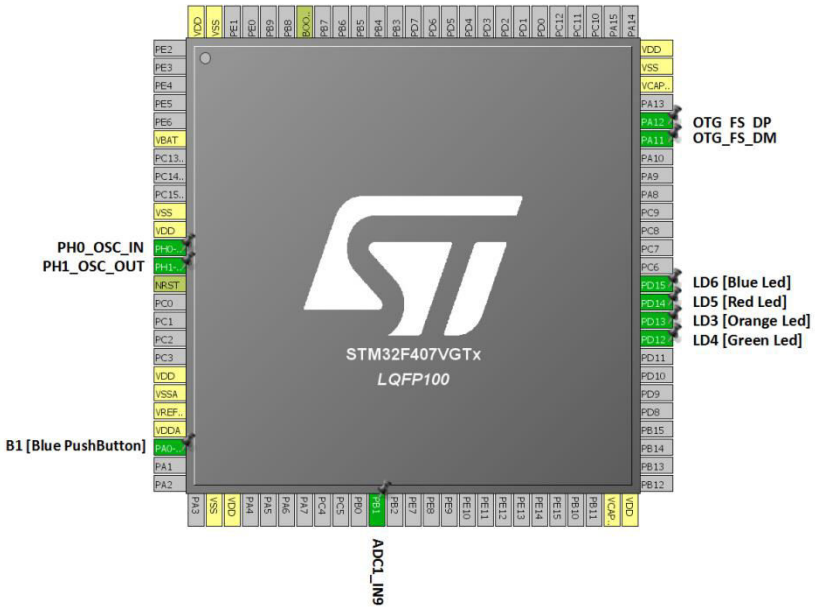

Figure-3. STM32F407 pin configuration.

This tool also supports the creation of a project in Keil u Vision 5 software, with all variable definition codes and class main.c. It also attaches the necessary libraries for the correct operation of all the parameters of the configuration.

b. Software function

It can be seen in the Figure-4 the change in the software architecture of a communication with an RS-232 port to a system emulating a serial port with USB. On the left part of Figure-4 the serial communication is represented, where the PC and the embedded device are located. In the block of the PC it can see the application developed, a driver for UART communication and a physical serial port. Connected to this port is the UART hardware of the device, a driver and the application developed in the embedded platform.

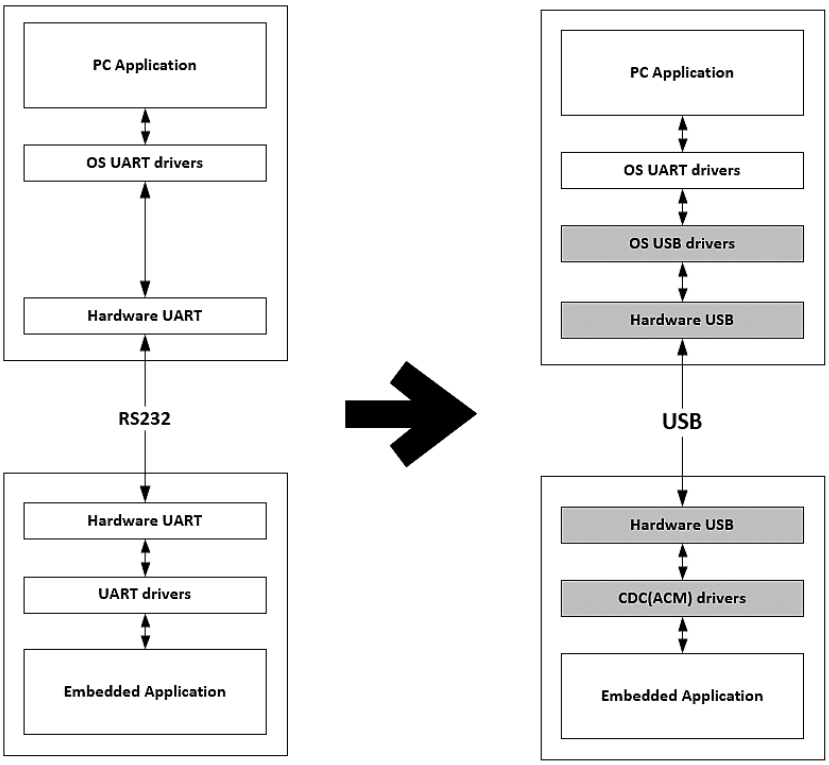

Figure-4. RS232 communication to USB serial virtual.

On the right side of Figure-4, it is the communication simulating a serial port, this type of connection modifications do not affect the applications developed in the computer and embedded platform. The serial physical port is replaced by a USB port and also on the device.

The communication drivers, present modifications in the computer, where a UART-USB driver is needed, which is responsible for reading the USB port buffer and pack the data in a serial form and can use for the application without having to modify it. The block of the device uses a CDC driver that allows the communication of an USB OTG port.

c. ACM (CDC) software structure

In the implementation of the virtual serial ports communication, it is used Communication Device Class (CDC), which supports a large range of devices. That can perform telecommunications functions, network and the execution of features of a USB component, such as the emulation of a virtual serial port using Abstract Control Model (ACM), sub-class of the CDC.

In the algorithm of STM32F407, each instance of the CDC has a separate file and interface functions such as: usbd_cdc_if.c and usbd_conf.c.The endpoint events manipulation of the CDC class are implemented in USBD_CDCn_Int_Thread and USBD_CDCn_Bulk_Thread that are started with USBH_Initialize that can see better in Figure-5.

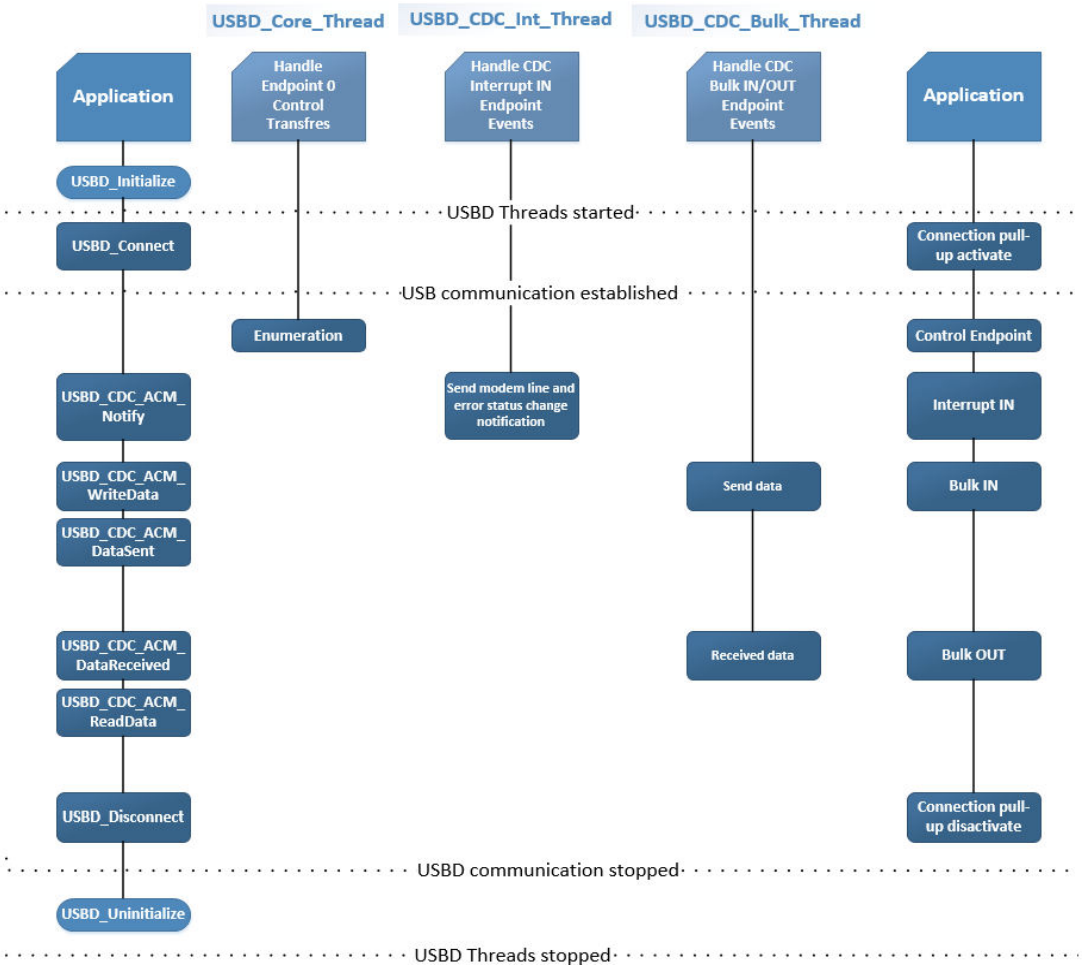

Figure-5. ACM software structure.

In the CDC class is executed instances of USBD_CDCn_Int_Thread and USBD_CDCn_Bulk_Thread, where enumeration, status verification, send and receive functions are executed. The USBD_CDCn_Int_Thread subprocess handles the endpoint of the IN interrupt while the USBD_CDCn_Bulk_Thread controls the Bulk IN and Bulk OUT endpoints.

d. Software in the STM32F407

In the program of embedded system STM32F407, the data is catched in a global unsigned integer variable of maximum size of 64 which is created previously in the file main.c. In order to send the data, first a value of an analogue input is read, it is equal to the variable adcValue and the function CDC_Transmit_FS is called, with the parameters of the variable adcValue and its length in bytes, as shown in Figure-6.

```
adcValue = HAL_ADC_GetValue(&hadc1);
CDC_Transmit_FS(adcValue,strlen((const char*)adcValue));
HAL_Delay(200);
```

Figure-6. Transmit data algorithm.

In the configuration, USB OTG FS (Full Speed, 12MHz) port global interrupt is activated, in other words, when the platform STM32F4 receive a serial data the function CDC_Receive_FS is immediately called, the code is described in Figure-7 and is defined in the usbd_cdc_if.c library.

```
static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  usb_receive_data[_index]=Buf[0];
  if(++_index==4)//if(++_index==64)
  {
    _index=0;
  }
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);

  return (USBD_OK);
  /* USER CODE END 6 */
}
```

Figure-7. Receive data function.

Also in this library, it can be found the definition of the CDC_Transmit_FS function, which is responsible for sending data, this definition is presented in Figure-8. This function override in the memory the transmission buffer, the data and length, after defines the handler type and calls the function USBD_CDC_SetTxBuffer.

```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
{
  uint8_t result = USBD_OK;
  /* USER CODE BEGIN 7 */
  memcpy(UserTxBufferFS,Buf,sizeof(uint8_t)*Len);
  USBD_CDC_HandleTypeDef *hcdc =
  (USBD_CDC_HandleTypeDef*)hUsbDeviceFS.pClassData;
  if (hcdc->TxState != 0){
    return USBD_BUSY;
  }
  USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
  result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
  /* USER CODE END 7 */
  return result;
}
```

Figure-8. Transmit data function.

e. Graphical user interface in C#

A graphical user interface was made in C# Visual Studio to corroborate the communication between the platform STM32F407 and the PC simulating a serial port. This interface has 1 Graph-chart where the variable received by the serial port is graphed, 4 buttons that turn on and off a respective LED, changing the status of an "O" variable that means on and "F" off. Finally it sends the statefor each LED. An example of the button algorithm con be seen in Figure-9.

Data reception is executed by interruption, which a serial object is created and instantiated. After that the in data variable is equaled to the SERIAL. Read Existing function, which reads all bytes enabled in the input buffer.

```
private void GREEN_Click(object sender, EventArgs e)
{
    DataWrite[0] ^= (byte)1;

    if (DataWrite[0] == 1) {
        G = "O";
        ledgreen.Image=Image.FromFile("LGREEN.jpg");
    } else {
        G = "F";
        ledgreen.Image = Image.FromFile("Off.jpg");
    }

    serialPort1.Write(G);
    serialPort1.Write(O);
    serialPort1.Write(R);
    serialPort1.Write(B);
}
```

Figure-9. Button function.

3. RESULTS

It performed different tests, one channel of the communication was analyzed, hence only the sending of data or the reception of data separately. Finally a general test was executed which the whole system was checked as shown in Figure-10. While the data is received and plotted, the activation signals of the green, yellow and blue LEDs are sent.

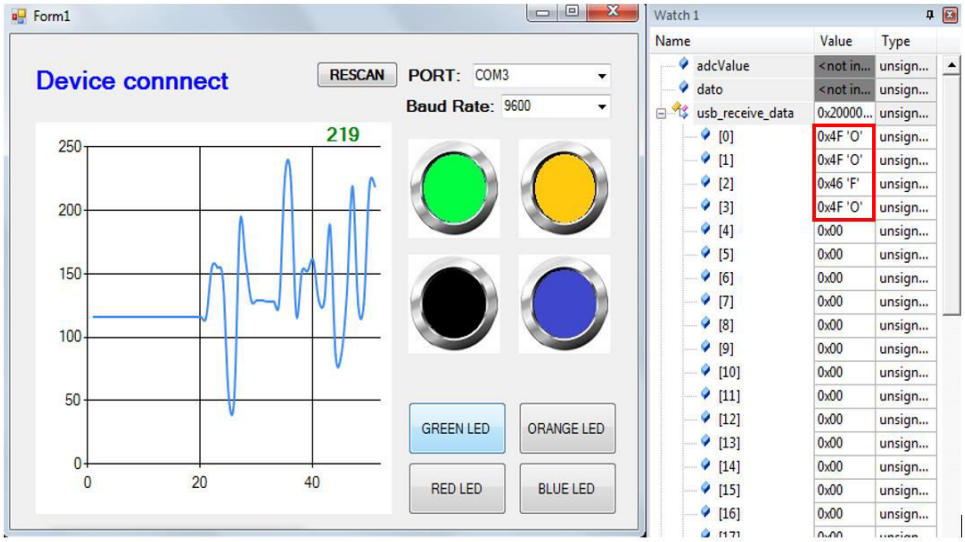

Figure-10. General test.

The Keil uVision5 software was configured in debug mode for USB / serial communication tests. It can see the data that receives the embedded system STM32FR407. In this case the variable with the name usb_receive_data.

4. CONCLUSIONS

It was verified that the USB communication simulating a serial port, can be established correctly without needing extra hardware like converters or modifying the applications in the involved systems. It was possible to transmit and receive data in C #, which is a High-level programming language, without any delay, loss of information or other common problem of USB / UART communication.

Furthermore, it can see the relevance of an interconnection between the UART-type system and the PC, thus currently RS-232 ports out of the consumer electronics market, but it is still hardly used in the industry and embedded systems.

REFERENCES

[1] A. K. a. R. R. Kanchi. 2014. Designing a learning platform for the implementation of serial standards using ARM microcontroller LPC2148. International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014). pp. 1-6.

[2] M. S. a. N. A. a. S. R. N. Reddy. 2015. Design and development of daughter board for USB-UART communication between Raspberry Pi and PC. International Conference on Computing, Communication Automation. pp. 944-948.

[3] F. S. a. S. H. a. H. Z. a. J. Liang. 2012. The FPGA verification of USB to RS-232 bridge controller. International Conference on Automatic Control and Artificial Intelligence (ACAI 2012). pp. 2123-2127.

[4] R. M. Yingbo Hu. 2016. Micro Digital, Ways to Use USB in Embedded Systems. [En línea]. Available: http://www.smxrtos.com/articles/usb_art/waysusb.htm. [Último acceso: 12 Diciembre 2016].

[5] ATMEL. 2016. Migrating from RS-232 to USB Bridge. [En línea]. Available: http://www.atmel.com/Images/doc4322.pdf. [Último acceso: 12 Diciembre 2016].

[6] R. S. a. A. G. a. A. Sfiraț. 2016. Robotic arm control in space with color recognition using a Raspberry PI. 2016 39th International Conference on Telecommunications and Signal Processing (TSP). pp. 689-692.

[7] B. B. S. a. N. Gaur. 2013. MODBUS communication in microcontroller based elevator controller. 2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE). pp. 1-5.

[8] R. Kandilarov. 2016. Digitally controlled 4-channel constant current source for LED luminary based on PIC microcontroler. 2016 19th International Symposium on Electrical Apparatus and Technologies (SIELA). pp. 1-4.

[9] D. S. a. L. J. a. K. E. G. a. Y. R. a. R. W. a. D. Dias. 201. Stereo audio streaming via Visible Light. 2016 Moratuwa Engineering Research Conference (MERCon). pp. 132-136.