Firmware implementation of UART using Bare metal programming

Pavithra K[1], Manjunath HV[2]

[1]*Student, M.Tech, Electronics and Communication Engineering, Dayananda Sagar College of Engineering*
[2]*Professor, Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering*

---***---

Abstract - *The development of real time applications requires various peripheral interfaces and communication channels. UART is standardized protocol used to establish communication with different hardware. The paper proposes to implement UART protocol at firmware level using bare metal programming model on STM32 CortexM4 microcontroller. The communication is established between two ports of board to demonstrate the implementation and the status and message is displayed on LCD for UI. The implementation provides in depth analysis and reliability of data transfer*.

Key Words: **UART, STM32, Cortex M4, bare metal programming, firmware**

1. INTRODUCTION

STM32Fxx series of microcontroller supports 2 USART ports that is configured to operate in asynchronous mode. The pins allocated are multiplexed, hence the configuration registers are used to indicate the required operation. Liquid Cristal Display (LCD) is specifically interfaced to indicate different operations being performed during communication. The success and failure of the communication can also be indicated on the panel in real time.

Cube IDE (Integrated Development Environment) is configured for STM32F4xx Cortex M4 microcontroller to facilitate programming. Bare metal programming method is used to improve the code efficiency and reduces executable file size. The UART communication is programmed to work in loopback mode and display the status of communication on LCD interfaced. The coding is done in C language to make the code microcontroller architecture independent.

(As UART is hardware for asynchronous serial communication, with configurable data format and transmission speeds, data bits are sent one by one from LSB to MSB. For precise timing by communication channel start and stop bits are framed. By cross connection of Tx and Rx pins of two devices using RS232 protocol, bidirectional communication is established.)

A universal asynchronous receiver-transmitter is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. It sends data bits one by one, from the least significant to the most significant, framed by start and stop bits so that precise timing is handled by the communication channel. The bidirectional communication is established by cross connection of Tx and Rx pins of two devices using RS232 protocol.

The pins are multiplexed with more than one operation, by default all port act as input output pins. Therefore, when the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the Tx pin is at high level. When a transmission is taking place, a write instruction stores the data in the data register and which is copied in the shift register at the end of the current transmission [1]. When no transmission is taking place, a write instruction places the data directly in the shift register, the data transmission starts, and the transmission bit is immediately set. After writing the last data into the data register, it is mandatory to wait for transmission completion bit to set before disabling the UART or causing the microcontroller to enter the low-power mode.

During an UART reception, data shifts in least significant bit first through the Rx pin. In this mode, the data register consists of a buffer between the internal bus and the received shift register. When a character is received, the receive completion bit is set which indicates that the content of the shift register is transferred to the receiver data register.

The data transmitted should be sampled at the appropriate moment, otherwise there may be a data loss or data may get erroneous. The transmitter and receiver must be compatible on the baud rate to receive data accurately. The different control registers are configured to set appropriate baud rate and establish successful communication which is also verified and validated using status bits. The data word length is 8 bits with NRZ standard format and also includes a parity bit for validation [2]. Data registers available consists a buffer between the internal bus and the transmit shift register.

2. UART FRAME STRUCTURE

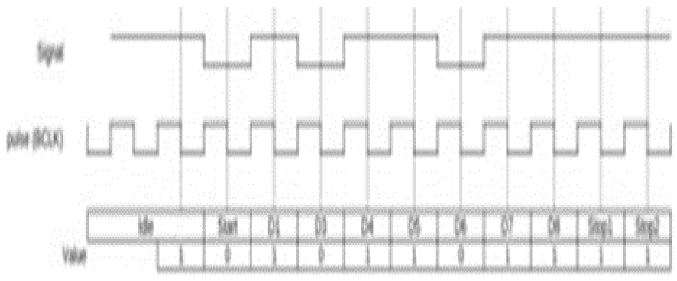

Figure 1: Timing Signal

Start bit: The start bit (indicates to the receiver that a new character is being transmitted) is used to signals the receiver that a new character is being transmitted.

Data bit: These bits represent the character in ASCII standard.

Parity bit: It is placed after all of the data bits whenever used. The parity bit detects if there is any error in data.

Stop bit: The next bit is always in the mark (logic high, i.e., '1') condition and called the stop bit(s). This signal to the receiver that the character is completely received. Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters.

3. UART TRANS-RECEIVER:

The STM32 ARM Cortex M4 provides multiple UART ports. In this firmware development, we are using port 2 & 3 for establishing communication.

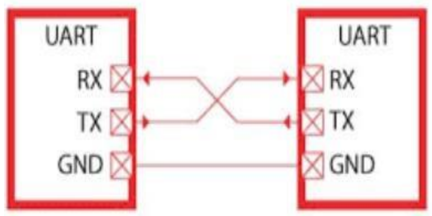

Figure 2: UART

If the line is held in the logic low condition for longer than a character time, this is a break condition that can be detected by the UART.

Transmission procedure on STM32:

● Enable the USART by writing the UE bit in USART_CR1 register to 1.

● Program the M bit to ‘0’ in USART_CR1 to define the word length of 8 bits.

● Program the number of stop bits as ‘00’ in USART_CR2 for 1 stop bit.

● Select the desired baud rate of 9600 using the USART_BRR register, 0x683 is loaded.

● Set the TE bit in USART_CR1 to send an idle frame as first transmission.

● Write the data to send in the USART_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.

After writing the last data into the USART_DR register, wait until TC=1.This indicates that the transmission of the last frame is completed. This request for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

Receiver procedure on STM32:

● Enable the USART by writing the UE bit in USART_CR1 register to 1.

● Program the M bit to ‘0’ in USART_CR1 to define the word length of 8 bits.

● Program the number of stop bits as ‘00’ in USART_CR2 for 1 stop bit.

● Select the desired baud rate of 9600 using the USART_BRR register, 0x683 is loaded.

● Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

● The RXNE bit is set. It indicates that the content of the shift register is transferred to the

● RDR. In other words, data has been received and can be read (as well as its associated error flags).

● The error flags can be set if a frame error, noise or an overrun error has been detected during reception.

● In multibuffer, RXNE is set after every byte received and is cleared by the DMA read to the Data Register.

● In single buffer mode, clearing the RXNE bit is performed by a software read to the

● USART_DR register. The RXNE flag can also be cleared by writing a zero to it. The

● RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

● The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV.

$$\text{Tx/Rx baud} = \frac{f_{CK}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

A 16x2 LCD is used to display the operations being performed my microcontroller. It can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data

The connections for UART is shown in Figure 1, the communication is established between UART2 (Tx-PA2 & Rx-PA3) and 3(Tx-PC10 & Rx-PC11) ports of STM32F405 microcontroller. The baud rate is set to 9600 and configuration of both transmission and reception is shown step below..

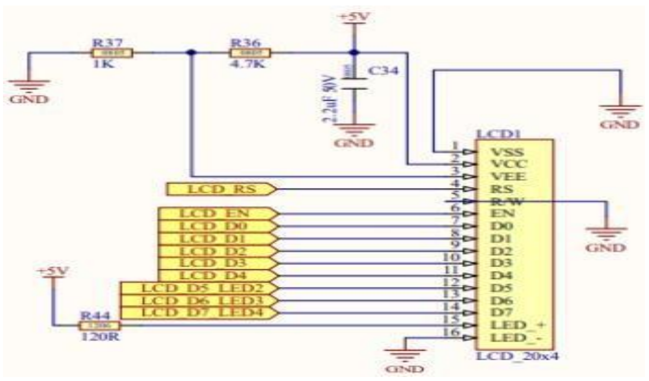

Figure 3: LCD connections

LCD interfacing is shown in figure 2 and respective pin mapping is illustrated in table 1. The obtained result is shown in figure 3, in which the status as well as the actual data communicated are displayed. The connections shown using jumper wires can be directly printed on PCB (Printed Circuit Board) once the design is finished.

Table 1: LCD interfacing pin mapping

SL No	STM32-Pin No	LCD Pin	Remarks
1	PA0	RS	Register Select
2	PA1	EN	Enable
3	GND	RW	Ground
4	PC4,PC5,PB0,PB1, PB12,PB13,PB14, PB15	D0,D1,D2,D3,D4, D5,D6 ,D7	Respective-order to be maintained
5	+5V	A, Vdd	Backlight Anode, Power supply (+)
6	GND	K, Vss, Vd	Backlight Cathode, power supply, Vd,(contrast)

4. ALGORITHM:

Initialization and configuration of UART port.

● UART is connected to AHB1 bus, hence clock is to be enabled using AHB1ENR register. Bit number 18 is enable (HIGH) bit for UART3.

● The clock is also enabled to GPIO port C which is used to exchange the data using AHB1ENR register, bit number 2.

● Pin number 10 (Bit 19 & 20) of port C is configured to operate in alternate function (AFR) mode by setting Bit19 to LOW and Bit 20 to HIGH, for UART operation using moder register.

● Bit number 12, 13 and 14 of 1st AFR register is set to HIGH for UART functionality.

● Baud rate of communication is set to 9600 kbps using BRR register.

● Clock frequency is 16 MHz

● BRR value is 1667.

● Baud rate calculation: Clock frequency / BRR (Baud Rate Register) register value.

● Bits 2 & 3 of Control register1 (CR1) are set HIGH to activate Receive Enable (RE) & Transmit Enable (TE).

● Bits 12 & 13 Control Register2 (CR2) are set LOW to configure 1 bit stop indicator.

● 13th bit of CR1 is used to enable / disable UART.

Implementation of "write" function.

● The character is loaded to transmit buffer through parameter passed to function.

● Enable to UART transmission using 13th bit of CR1 register.

● Wait till status 7th bit of Status Register (SR) is reset, which indicate transmission buffer empty.

● Then write new character to Data register (DR) .Implementation of "read" function.

● Wait till status 7th bit of Status Register (SR) is reset, which indicate transmission buffer empty.

● Function returns ASCII value of character received.

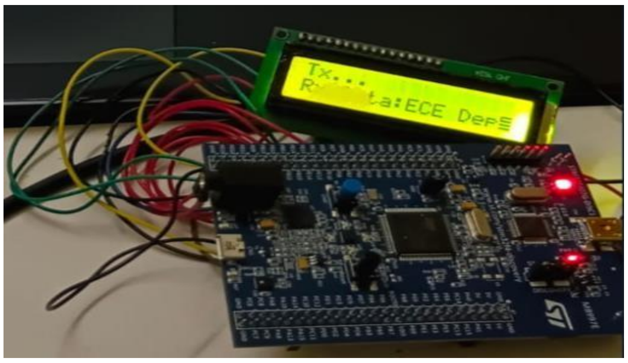

Figure 4: Connections and result

5. RESULTS AND CONCLUSION

In this work, designing and implementation of bare-metal coding is used for establishment of UART communication. The UART communication is established successfully at 9600 baud rate using bare-metal programming model. The firmware implementation using bare metal programming provides very fine control of operations. This also reduced code length as it avoids linking library function during compilation process. String of characters are sent over the channel and displayed on LCD display, which also indicates the status of operation. Once the transmission begins then system enters wait state till RXNE bit is set indicating successful transmission and reception of data.

REFERENCES:

1) Ashok Kumar Gupta, Ashish Raman, Naveen Kumar, and Ravi Ranjan, "*Design and Implementation of High-Speed Universal Asynchronous Receiver and Transmitter (UART)*" 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN) ©2020 IEEE

2) U. Nanda and S. K. Pattnaik, "*Universal Asynchronous Receiver and Transmitter (UART)*" 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, 2016, pp. 1-5. DOI: 10.1109/ICACCS.2016.7586376

3) Shahu, K., "*ASIC Design Implementation of UART using Synopsys EDA tools*" 2019 (Doctoral dissertation, California State University, Northridge).

4) Teerth Patel, Usha Mehta, "*Design and Simulation of UART Communication Module using Different Approach*", International Journals Digital Communication and Analog Signals

5) Jiayu Hu, Xiaoen Yan, "STM32-Based Design of Race car Running Status Monitoring System and Instrument Display", 2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), IEEE

6) QL. Cui and X. Chen, "Research on STM32 internal temperature sensor and realization of temperature measurement system", Digital technology and application, vol. 2011, no. 10, pp. 61-62, 2011

7) R. Zhai and J.L. Zhou, "Design of USB serial communication port based on STM32", Foreign electronic measurement technology, vol. 40, no. 1, pp. 92-95, 2021

8) Shihua Tong, "Design and simulation analysis of UART IP Core," 2011 Second International Conference on Mechanic Automation and Control Engineering, Inner Mongolia, China, 2011, pp. 5685-5688, doi: 10.1109/MACE.2011.5988319.

9) Liu Xiaoyue and Li Xing, "Serial Communication System of Mobile Devices and Embedded Computer Based on C/S Structure", Future Computer Science and Education (ICFCSE) 2011 International Conference, pp. 598-600, 20-21 Aug. 2011.

10) J. Ducloux, P. Petrashin, W. Lancioni and L. Toledo, "Embedded USB dual-role system for communication with mobile devices", Argentine School of Micro-Nanoelectronics Technology and Applications (EAMTA) 2011, pp. 1-7, Aug. 2011.

11) M. Sharma, N. Agarwal and S. R. N. Reddy, "Design and development of daughter board for USB-UART communication between Raspberry Pi and PC," International Conference on Computing, Communication & Automation, Greater Noida, India, 2015, pp.944-948,10.1109/CCAA.2015.7148532.

12) J. Ducloux, P. Petrashin, W. Lancioni and L. Toledo, "Embedded USB dual-role system for communication with mobile devices", *Argentine School of Micro-Nanoelectronics Technology and Applications (EAMTA) 2011*, pp. 1-7, Aug. 2011.