UART-Perifericos

Date: 02/09/2025	**UART-Perifericos Report**
Andrea Zarahi Rubio Quezada	**A01645257**

Introduction

In this project, the UART (Universal Asynchronous Receiver/Transmitter) protocol was implemented on an STM32H7 microcontroller using direct register access (CMSIS), without relying on the HAL library.

The goal was to configure the USART3 peripheral to transmit at 9600 baud, 8N1 format, using PB10 (TX) and PB11 (RX) pins.

The system periodically sends the character 'H' every second, and the transmitted signal is validated using an oscilloscope or a logic analyzer.

Theory

UART is an asynchronous serial communication protocol, typically used for point-to-point data transfer.

The standard 8N1 format means:

- 1 Start bit (0)
- 8 Data bits (LSB first)
- No parity
- 1 Stop bit (1)

Baudrate determines the speed: at 9600 baud, each bit lasts about 104.17 µs.

In STM32, each USART peripheral requires a clock source (in this case, HSI = 64 MHz), and a divisor (BRR) is calculated to generate the exact baudrate.

Implementation

Clock configuration

- Enabled the peripheral clock for GPIOB (pins PB10/PB11).
- Enabled the clock for USART3 through the APB1 bus.

GPIO configuration

- PB10 and PB11 set to Alternate Function (AF7) for USART3.
- TX (PB10) configured as push-pull, very high speed.
- RX (PB11) configured with a pull-up resistor for stable idle state.

USART3 configuration

- Reset the peripheral for a clean start.
- Set 8N1 format: 8 data bits, no parity, 1 stop bit.
- Oversampling ×16.
- Calculated BRR using the HSI clock (64 MHz) to achieve 9600 baud.
- Enabled the transmitter (TE) and the USART (UE).

Timing

- The DWT cycle counter was used to generate precise 1-second delays between transmissions.

Code

https://github.com/Andyrubio5/Embedded-Systems/blob/main/communication_protocols/UART_Perifericos/CM7/Core/Src/main.c

Key Points of the Code:

- Direct register access: all configuration is done by writing directly to RCC, GPIO, and USART registers.
- GPIO AF7: connects PB10/PB11 to USART3.
- BRR calculation: ensures exact baudrate using the USART clock (HSI = 64 MHz).
- Polling method: waits until TX is ready before sending the next byte.
- DWT counter: provides accurate millisecond delays without HAL.

Explanation

This program uses the USART3 peripheral hardware instead of emulating UART with GPIO bit-banging. By connecting the pins PB10/PB11 to the USART3 module (via Alternate Function 7), the microcontroller can generate UART frames automatically.
The BRR register ensures that the bit period matches the desired baudrate. The program then sends a 'H' character every second, and the delay is implemented using the CPU cycle counter (DWT).

Video Result:

https://drive.google.com/drive/folders/1JIuHIXCeUXGwLCJnHyIXJzaIKoOi6JnH?usp=sharing

Conclusion

The implementation of USART3 using direct register configuration on the STM32H7 microcontroller successfully demonstrated how to generate UART communication without relying on high-level libraries such as HAL. By configuring GPIO pins PB10 and PB11 in Alternate Function 7, enabling the USART3 peripheral clock, and correctly calculating the BRR value using the HSI clock (64 MHz), a reliable 9600 baud, 8N1 communication was achieved.

The use of the DWT cycle counter allowed precise 1-second delays, proving that low-level features of the Cortex-M7 can replace software delay functions. The oscilloscope measurements confirmed that the transmitted bits matched the expected baudrate timing, and the test terminal displayed clean 'U' characters at regular intervals.