**Software Serial on the STM32F411CEU6**

# Table of Contents

[1. Table of Contents 2](#_Toc1)

[2. Table of Figures 3](#_Toc2)

[3. UART 4](#_Toc3)

[4. Console App 5](#_Toc4)

[5. Implementation 6](#_Toc5)

[6. Result 7](#_Toc6)

# 2. Table of Figures

[Figure 1. Data frame 4](#_Toc3)

[Figure 2. Transmission example 4](#_Toc4)

[Figure 3. Example of a transmission with 8 data bits 1 parity bit and 2 stop bits. 4](#_Toc5)

[Figure 4. RX state diagram 5](#_Toc6)

[Figure 5. TX state diagram 5](#_Toc7)

[Figure 6. RX state struct 6](#_Toc8)

[Figure 7. Settings struct 6](#_Toc9)

[Figure 8. Timer configuration 6](#_Toc10)

[Figure 9. Start RX function 7](#_Toc11)

[Figure 10. Falling edge interrupt 7](#_Toc12)

[Figure 11. RX timer interrupt 7](#_Toc13)

[Figure 12. TX code 8](#_Toc14)

# 3. UART

UART (Universal Asynchronous Receiver Transmitter) is a simple full duplex protocol which is used in almost all embedded applications. As evident by the name the protocol is asynchronous, this means that both parties have to do their own time keeping. This makes UART a little bit less reliable and robust it does make it very simple to connect because UART uses only one wire (if transmission only goes one way) and a common ground of course. The only issue is that you cant just plug into a UART bus without knowing what baud rate is used because it has to be the same on both devices so that the bits are read or sent at the right time.

UART uses data frames to send its data (see Figure 1). These frames can take on different forms depending on the settings like: the amount of data/stop bits or the presence of a parity bit. This parity bit can be used to verify if the data is received correctly.



Figure 1. Data frame

***analog.com/-/media/images/analog-dialogue/en/volume-54/number-4/articles/uart-a-hardware-communication-protocol/335962-fig-10.svg?w=900&imgver=1*. (n.d.). Retrieved June 22, 2023**

These frames are then sent bit for bit at the baud rate this is shown in Figure 2. The formula for the time between bits (in seconds) is the following: 1/(baud + 1). Note that TX is connected to RX (and vice versa).

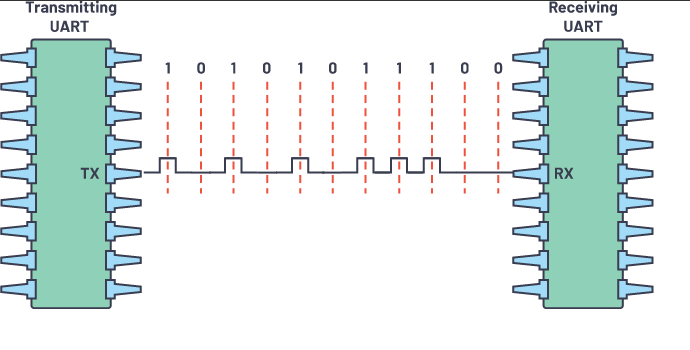


Figure 2. **Transmission example**

***analog.com/-/media/images/analog-dialogue/en/volume-54/number-4/articles/uart-a-hardware-communication-protocol/335962-fig-10.svg?w=900&imgver=1*. (n.d.). Retrieved June 22, 2023**

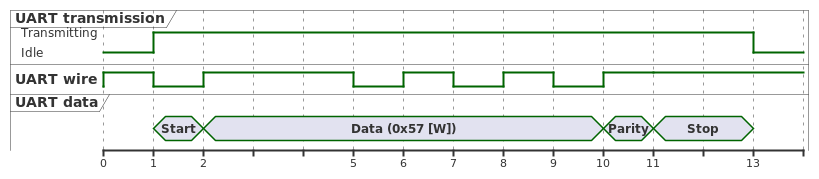


Figure 3. Example of a transmission with 8 data bits 1 parity bit and 2 stop bits.

# 4. Implementation

The RX code will follow the behavior seen in Figure 4. The only difference is that when an error is detected the code will simply continue whilst setting an error flag instead of ending early because there may be a possibility to repair the data (this is up to the user).

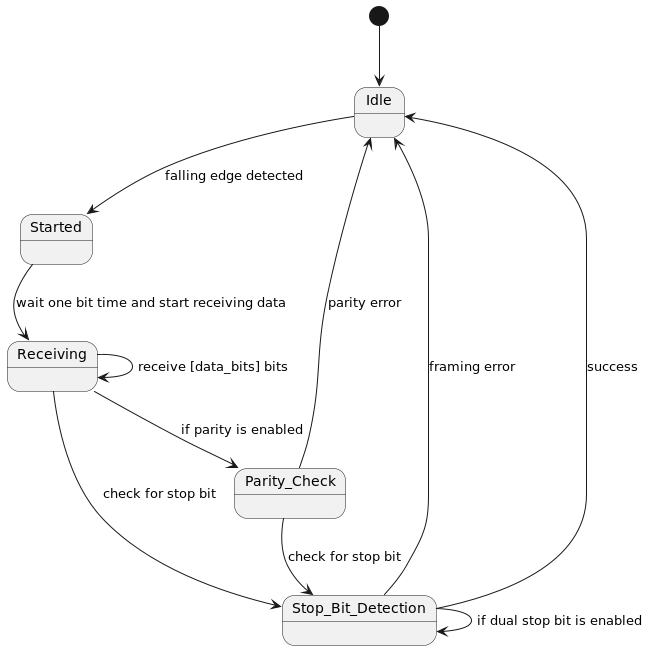


Figure 4. RX state diagram

This is the TX state diagram which basically does the same as RX.

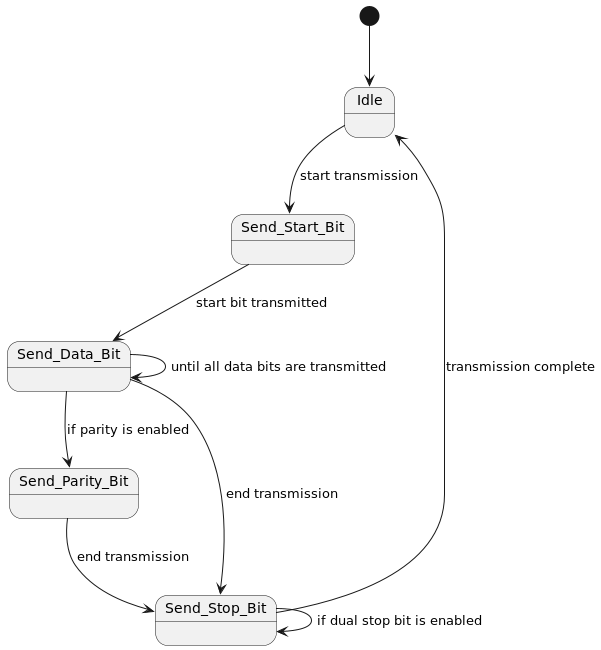


Figure 5. TX state diagram

The implementation has a TX and RX portion. The TX code is a bit simpler than the RX code and this is because TX is made to be blocking whereas RX is completely interrupt based, this means that for RX a state has to be kept this is done in the struct seen below (figure 6).



Figure 6. RX state struct

All settings for the data frames are stored in the struct seen below.

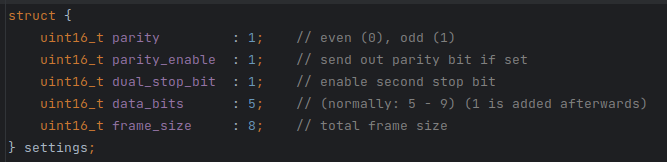


Figure 7. Settings struct

The bit timing is done using timers the initialization of which can be seen in figure 8. The RX timer runs twice as fast as the TX timer but this is because it has to generate an interrupt and this is only done when it has ticked twice so the code effectively runs on the same timing.

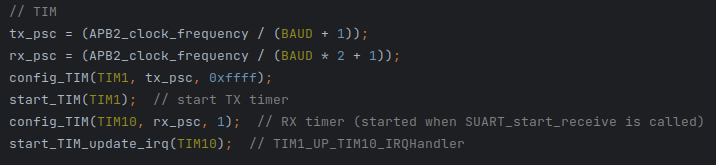


Figure 8. Timer configuration

This function starts the RX timer and enables the falling edge interrupt on the RX pin, it also sets the buffer and resets the state. The falling edge interrupt is used to detect when a start bit is sent after which the state is unlocked.

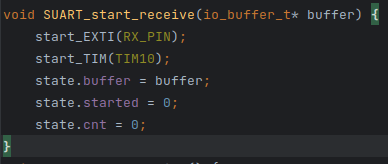


Figure 9. Start RX function

This is the falling edge interrupt which unlocks the state by setting ‘started’

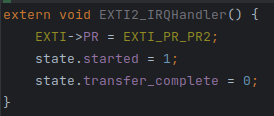


Figure 10. Falling edge interrupt

This is the RX timer interrupt which samples the bit on the RX pin and updates the state accordingly (see Figure 4 for the RX state diagram).

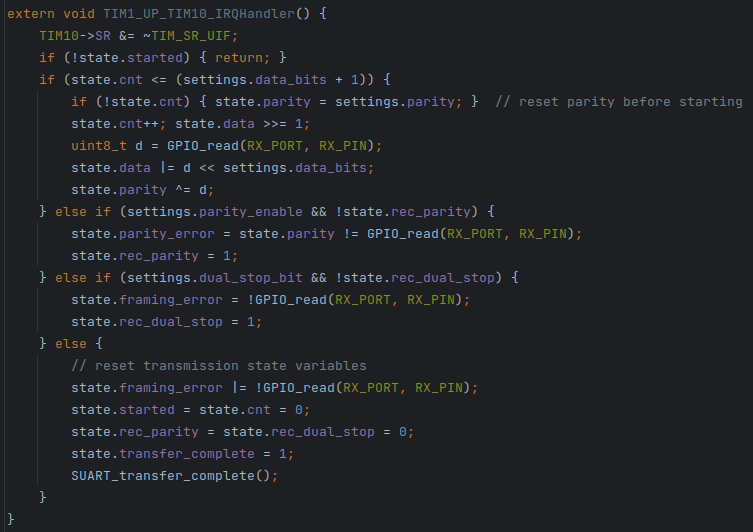


Figure 11. RX timer interrupt

This is the TX code which is described in the TX state diagram (Figure 5).

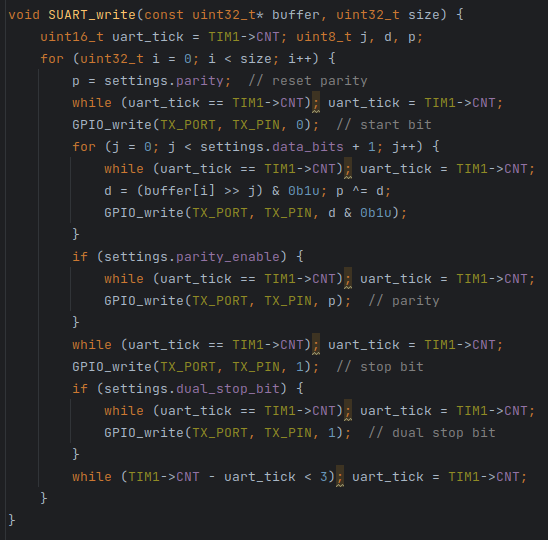


Figure 12. TX code

# 5. Console App

The console app built on top of the implementation of the Software Serial is used to get information about the micro-controller.

# 6. Result