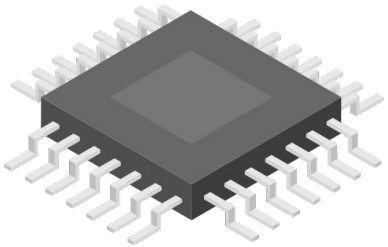
PADO

Labs

Microcontroladores



Comunicação UART

Prof.º: Pablo Jean Rozário

pablo.jean@padotec.com.br

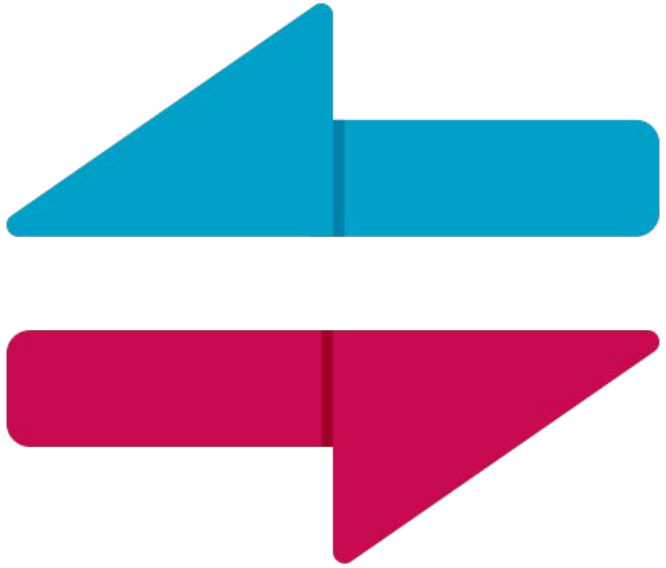
/in/pablojeanrozario

https://github.com/Pablo-Jean

Índice da Aula #5

PADO Labs

Universal Asynchronous/Synchronous Receiver Transmitter



Introdução

Para realizar a troca de informações entre computadores, microcontroladores e/ou dispositivos são utilizadas **interfaces de comunicação**, que por sua vez, estabelecem um *link* através de uma interface física para realizar as transações.

Antigamente se utilizava interfaces paralelas, mas atualmente, se utiliza muito interfaces serializadas.

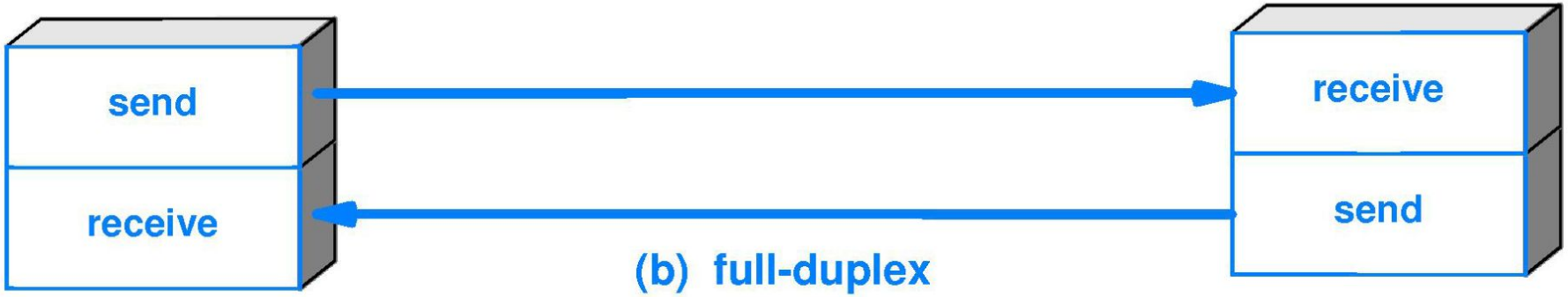
Estas interfaces são divididas em três categorias: *Full-Duplex, Half-Duplex* e *Simplex*.

Introdução

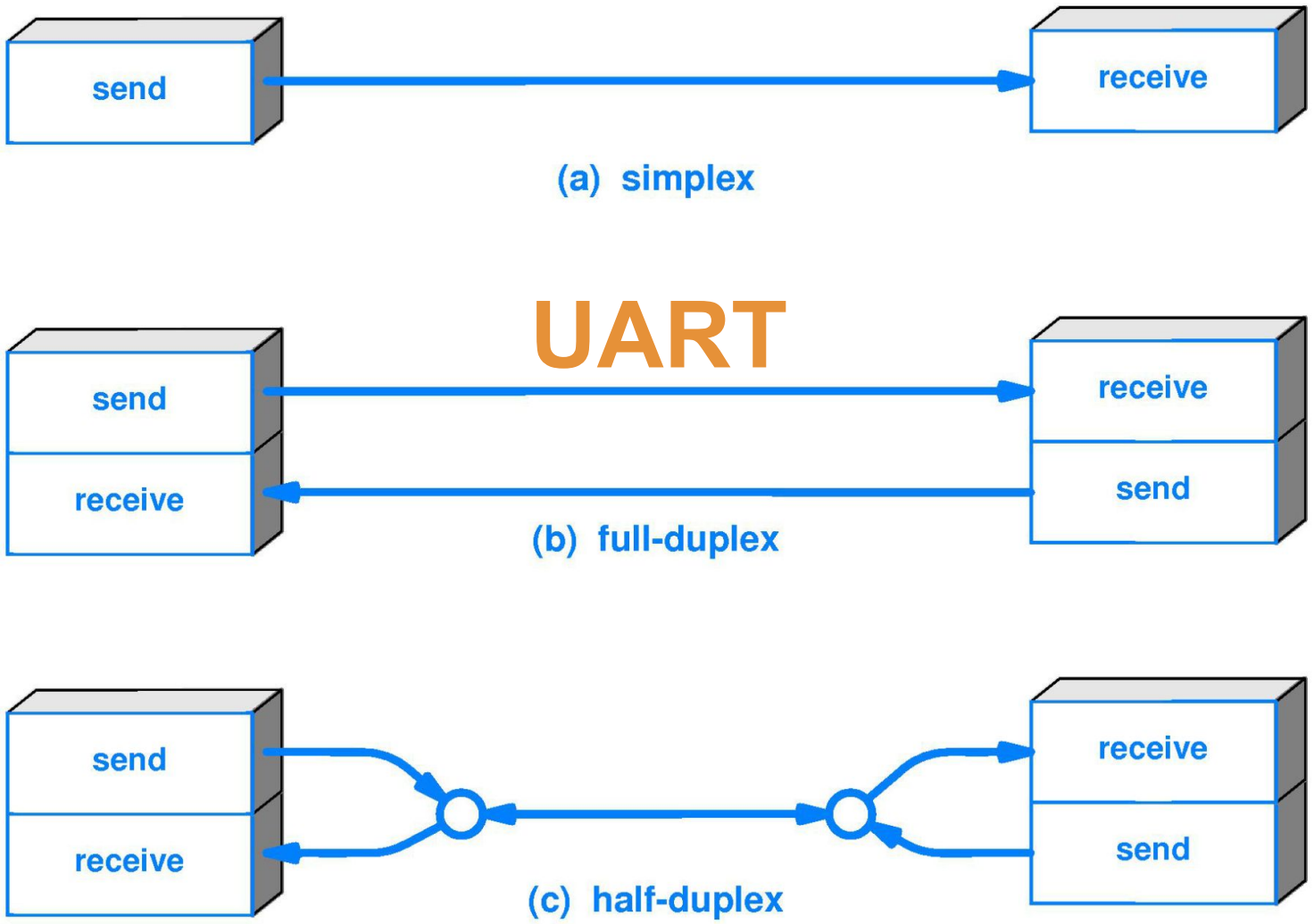
(a) simplex

Introdução

Introdução

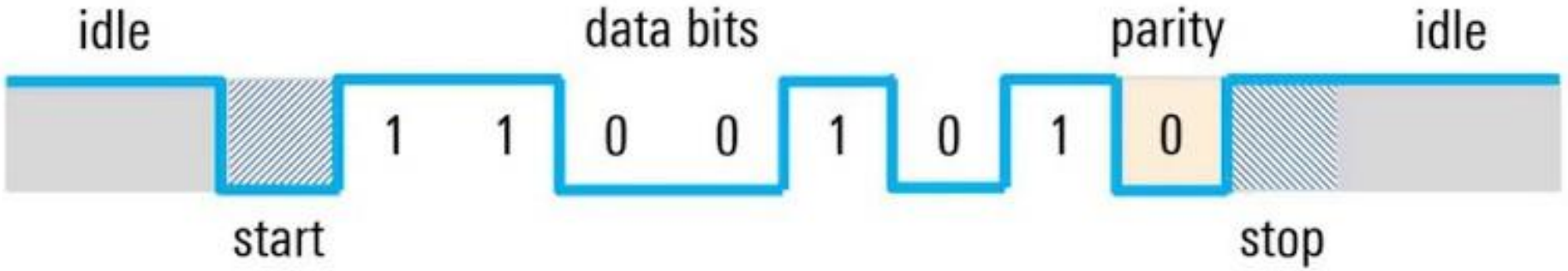


Introdução

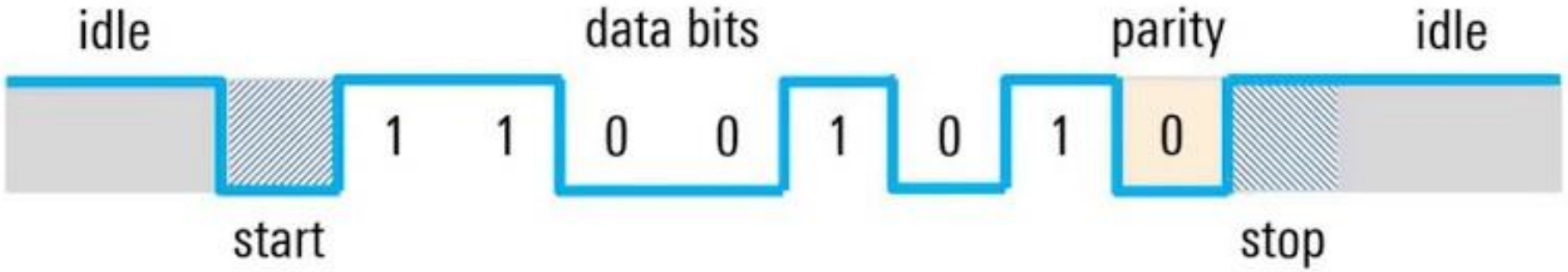

send
receive
(a) simplex
UART
send
receive
receive
send
(b) full-duplex
send
receive
receive
send
(c) half-duplex

USART - *Frame*

A USART é um *data link* (padrão OSI) utilizado para **transmissão** e/ou **recepção** de dados. Onde cada *bit* é enviado um a um, do menos significativo ao mais significativo (LSB), feito entre um *start* e *stop bit*. A amostragem é feita através do **tempo**, utilizando o sincronismo pelo ***clock*** interno.



UART - *Frame*

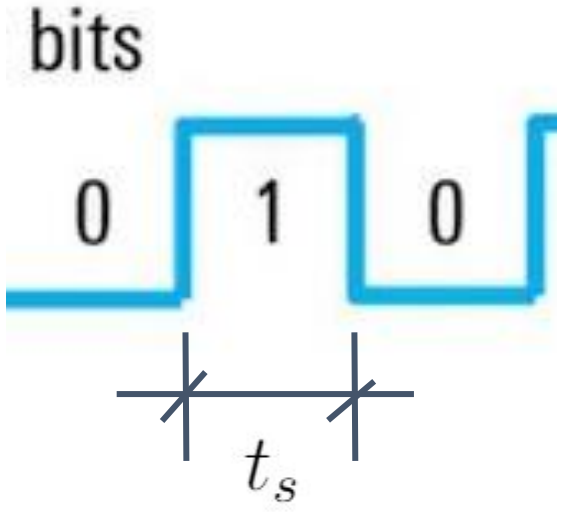


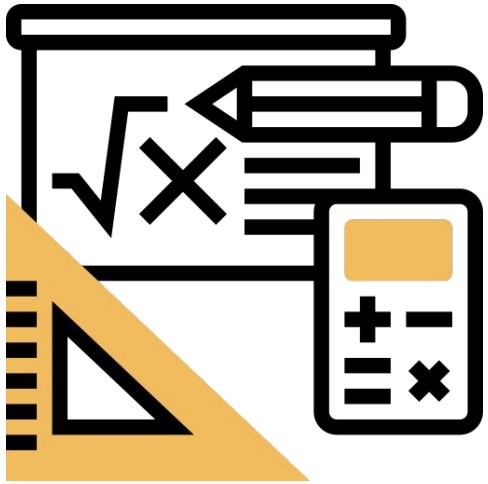
idle: Estado ocioso, enquanto não há transmissão de dados.
start: Sinal de nível baixo que sinaliza um início de transmissão.
data bits: informações que está sendo transmitida.
parity: utilizado para realizar a checagem de integridade do pacote (opcional).
stop: sinal alto que sinaliza o fim da transmissão (1 ou 2 *bits*).

UART - Temporização

A frequência de transmissão é chamada de *baudrate*, que é a quantidade de bits que serão transmitidos por segundo. O tempo de transmissão de cada *bit* é o inverso do *baudrate*.

$$t_s = \frac{1}{baudrate}$$



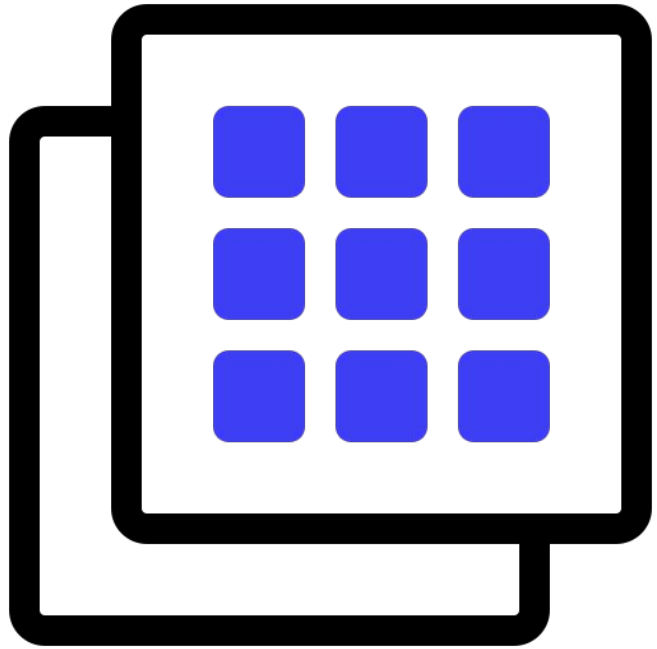


UART - Baudrate

Existem valores padrões que podem ser utilizados para o *baudrate*, mas não são obrigatórios serem seguidos.

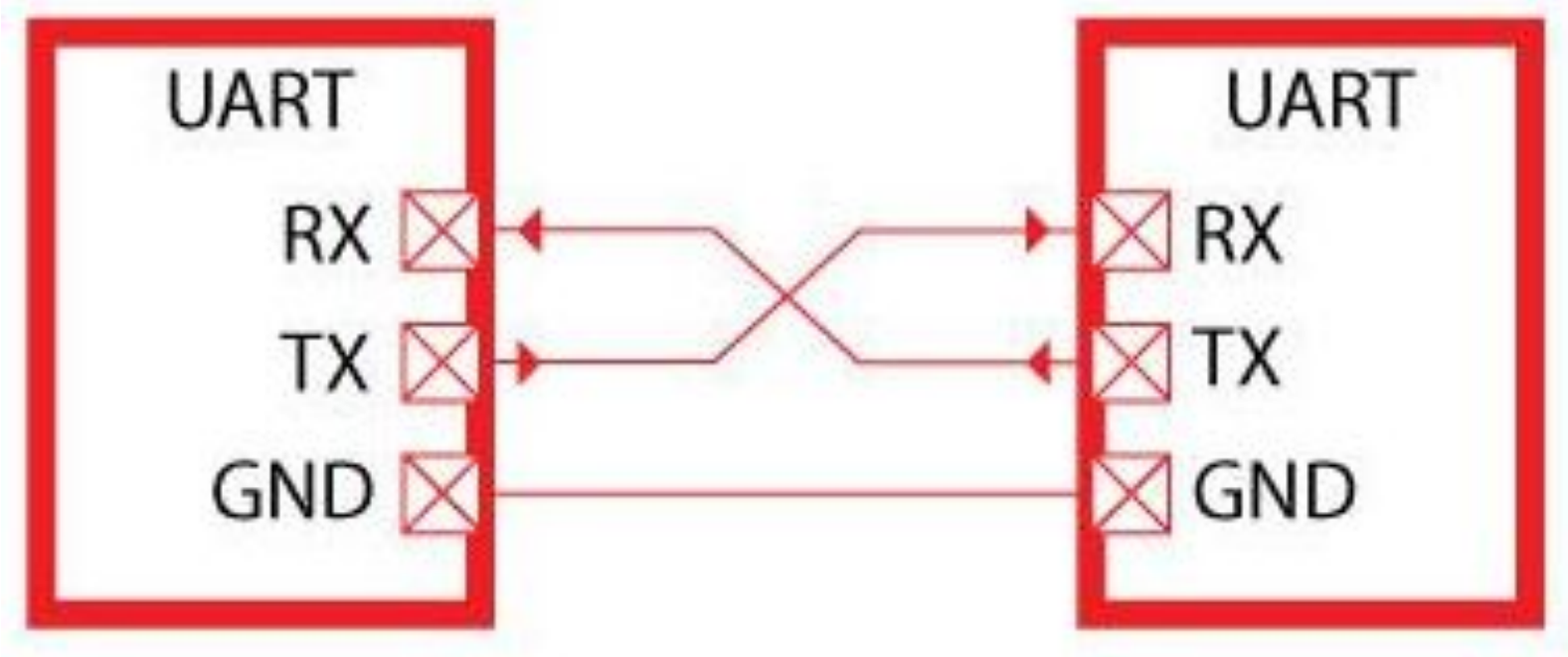
Alguns valores são:

2400	**38400**
4800	**57600**
9600	**115200**
14400	**256000**
19200	**...**



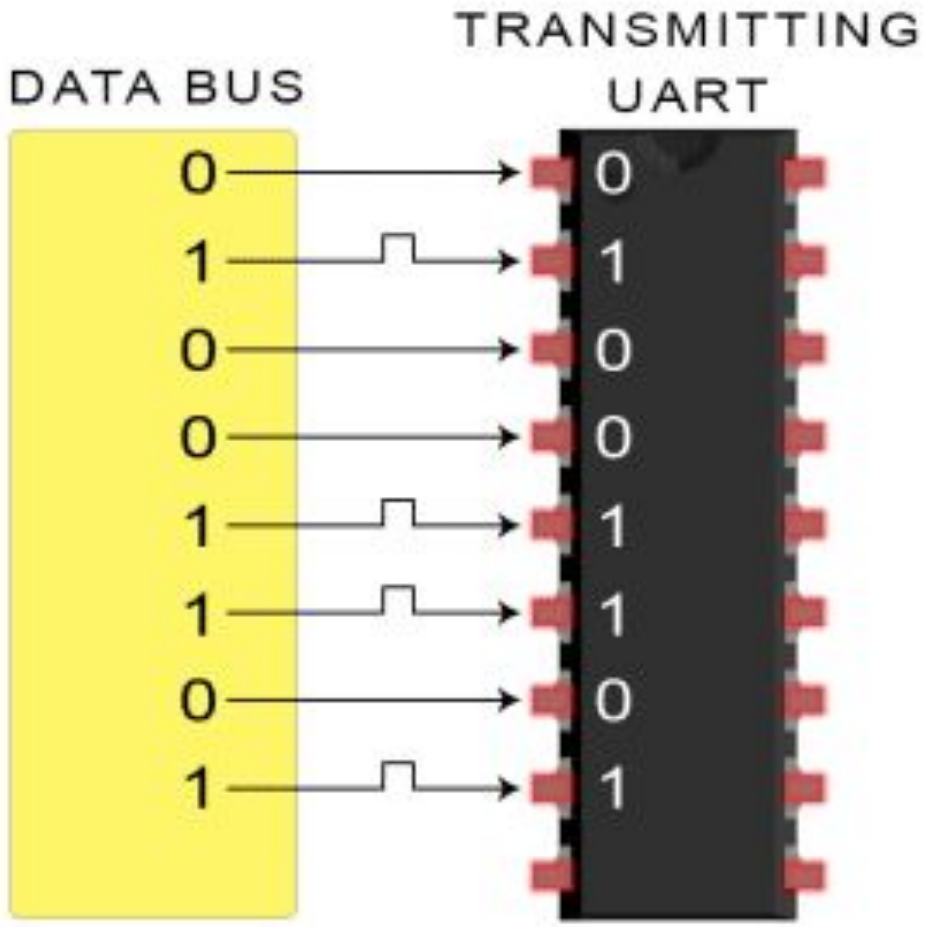
UART - Camada Física

A USART em modo **assíncrono**, utiliza apenas dois terminais: **Tx***(transmitter)* e **Rx***(receiver)*. Onde



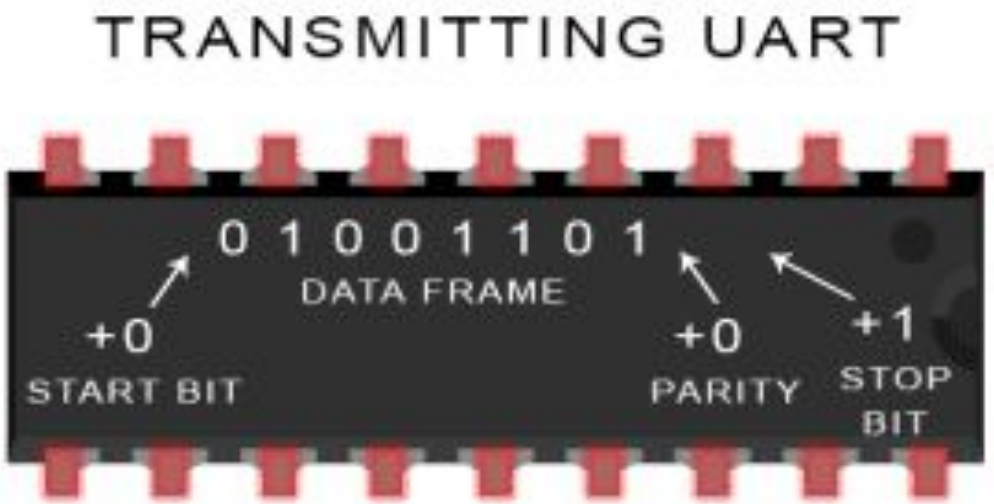
UART - Fluxo

O periférico recebe o byte do barramento para ser enviado.



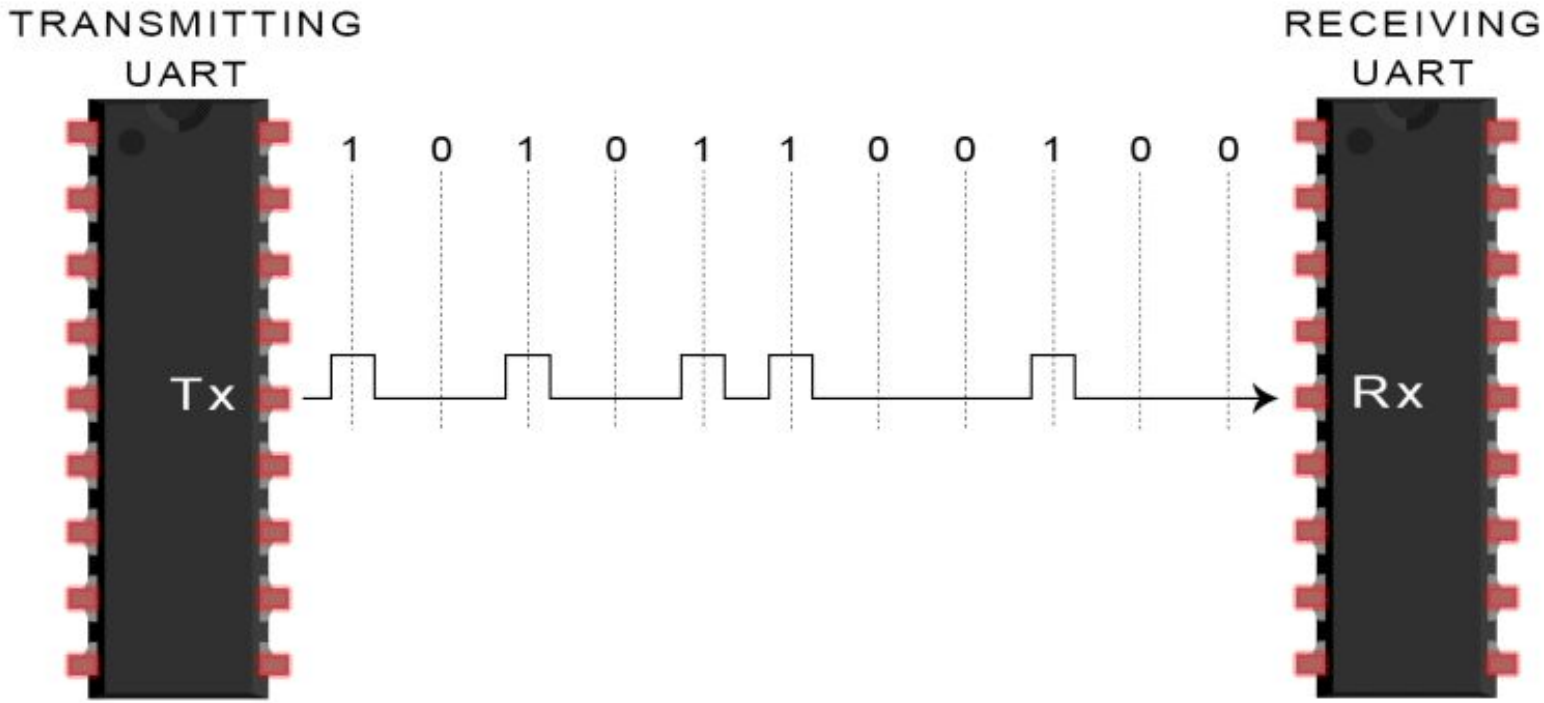
UART - Fluxo

Então, o periférico então prepara o *frame* para o envio pelo terminal transmissor.

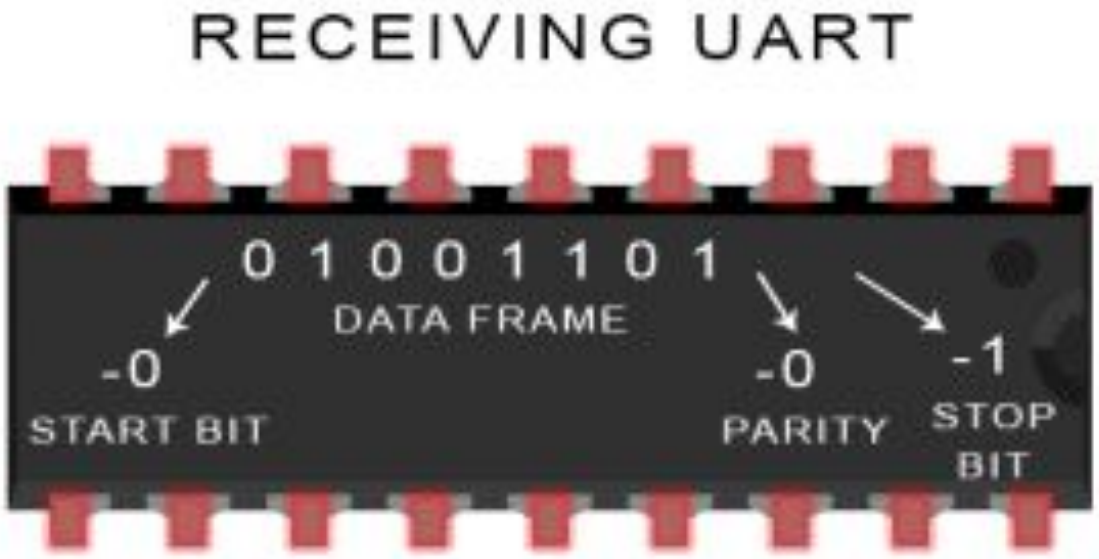


UART - Fluxo

O *frame* após preparado é enviado pelo terminal de transmissão.



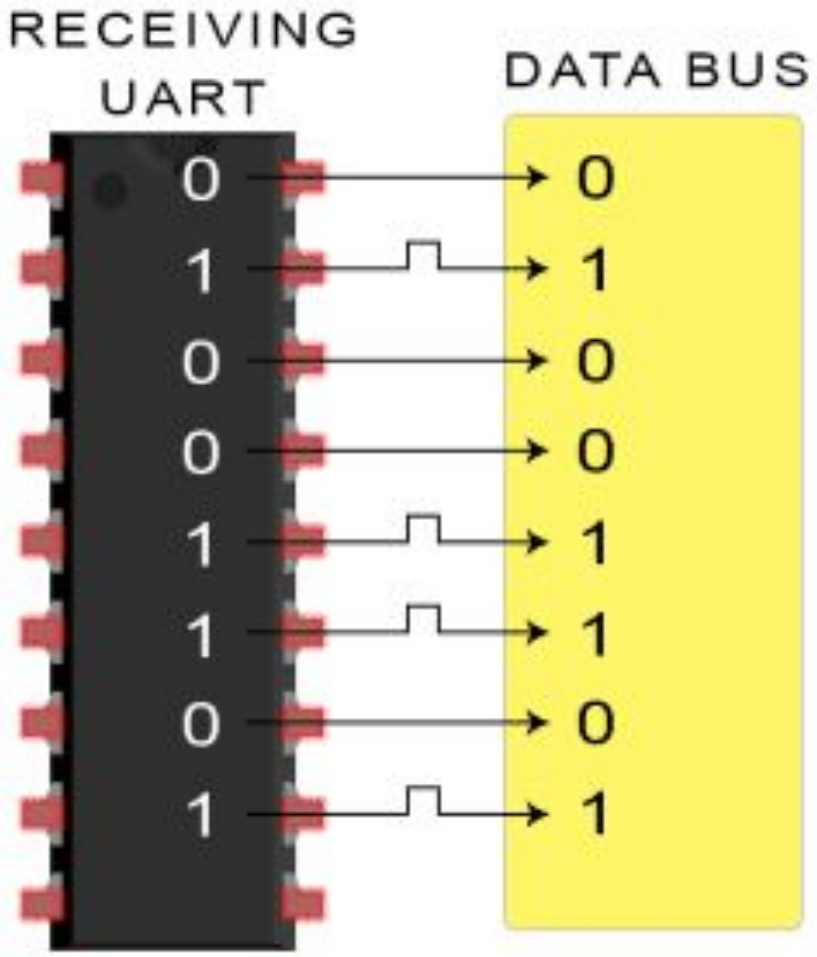
UART - Fluxo



Quando o receptor detecta o *start*, inicia então a montagem do *frame* amostrado pelo *baudrate*, até o *stop* bit.

Caso esteja habilitado a paridade, o periférico checa a integridade.

UART - Fluxo



Por fim, o periférico de UART seta uma *flag* para sinalizar que foi recebido um dado, para que então a CPU possa coleta o dado pelo barramento.

UART - Erros

Quando um erro ou algo anormal acontece, o periférico pode gerar alguns erros para sinalizar para a CPU que ***Algo de errado não está certo***.

Veremos alguns destes erros.

UART - *Overrun Error*

O *overrun* ocorre quando o *buffer* de recepção está cheio e um pacote é recebido, ocasionando o descarte deste último *frame* recebido e a geração do erro.

Isto ocorre quando a CPU ou o DMA não consegue coletar os dados com velocidade
suficiente.

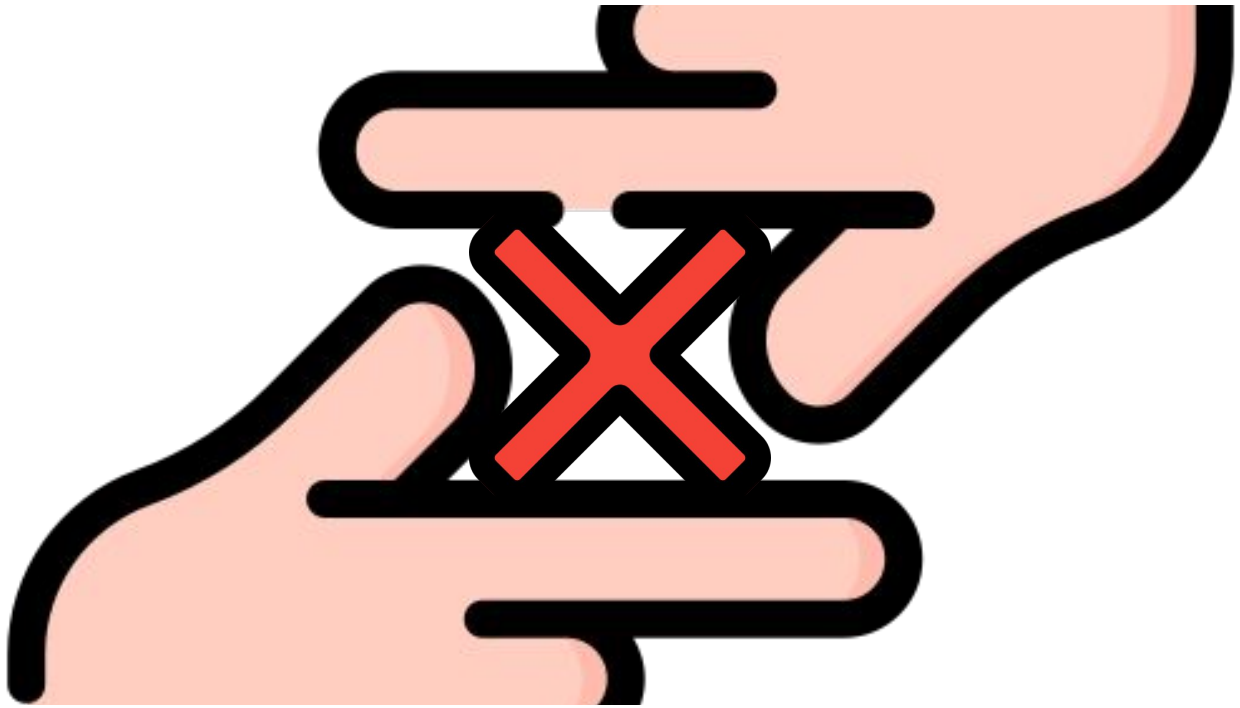
UART - *Underrun Error*

O *underrun* ocorre quando o *buffer* de transmissão e este tenta enviar mais um pacote. Em modo assíncrono é tratado como uma *flag* que indica o fim de uma transmissão.

Mas em modo síncrono, é um erro mais sério.

UART - *Framing Error*

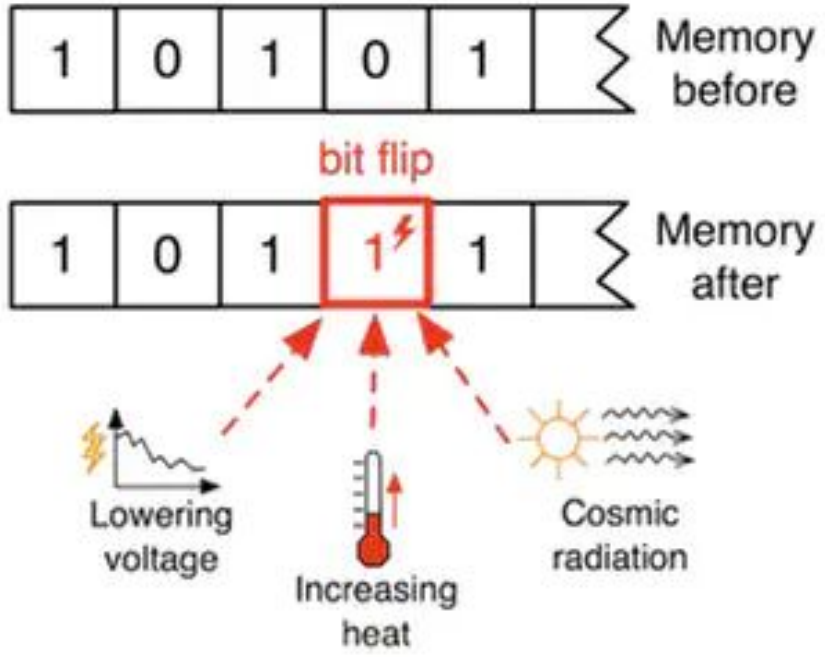
Este errado é gerado pela ausência do *stop bit*. Pois quando um *start* bit é detectado, o periférico inicia a amostragem do terminal e é esperado o *stop* ao final.



UART - *Parity Error*

Acontece quando um pacote recebido está com seu bit de paridade incongruente com o calculado, ou seja, houve um *bit* ou mais que durante a transmissão foram alterados, gerando um *frame* invalido.

Ocorre somente quando a **paridade** estiver habilitada.



UART - Vantagens

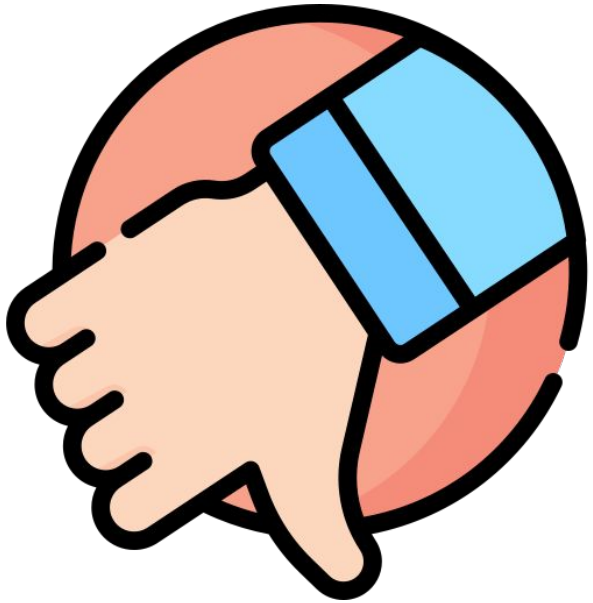
Algumas **vantagens** da interface UART são:

- Uso de apenas duas conexões lógicas (Rx e Tx)
- Não é necessário sinal de *clock*
- Permite uso de paridade para integridade de pacotes
- Bem documentado e amplamente empregado

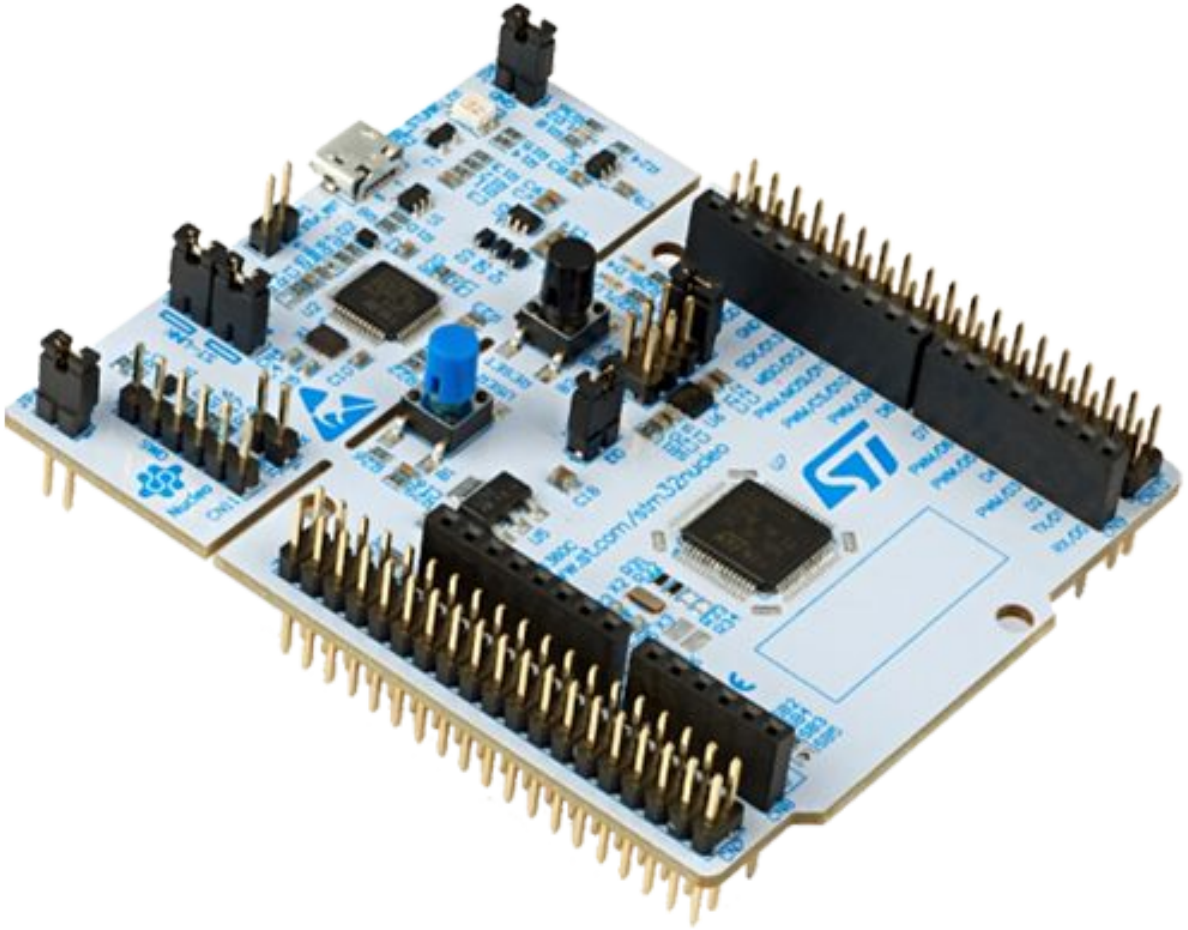
UART - Desvantagens

E algumas **desvantagens** da interface UART são:

- Cada *frame* suporta no máximo 9 *bits.*
- Sem suporte a múltiplos escravo e mestres.
- O *baudrate* dos periféricos devem estar com no máximo 10% de desvio

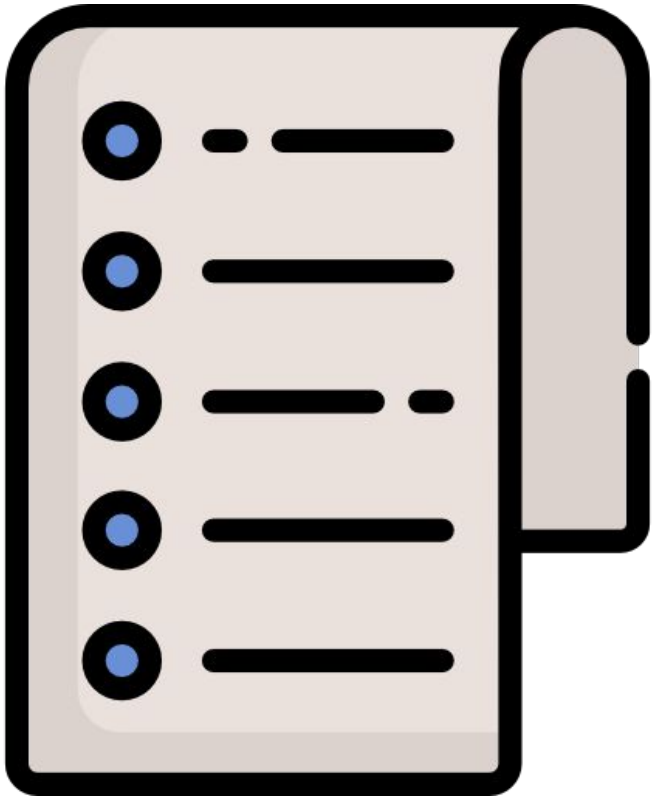


USART no STM32G0

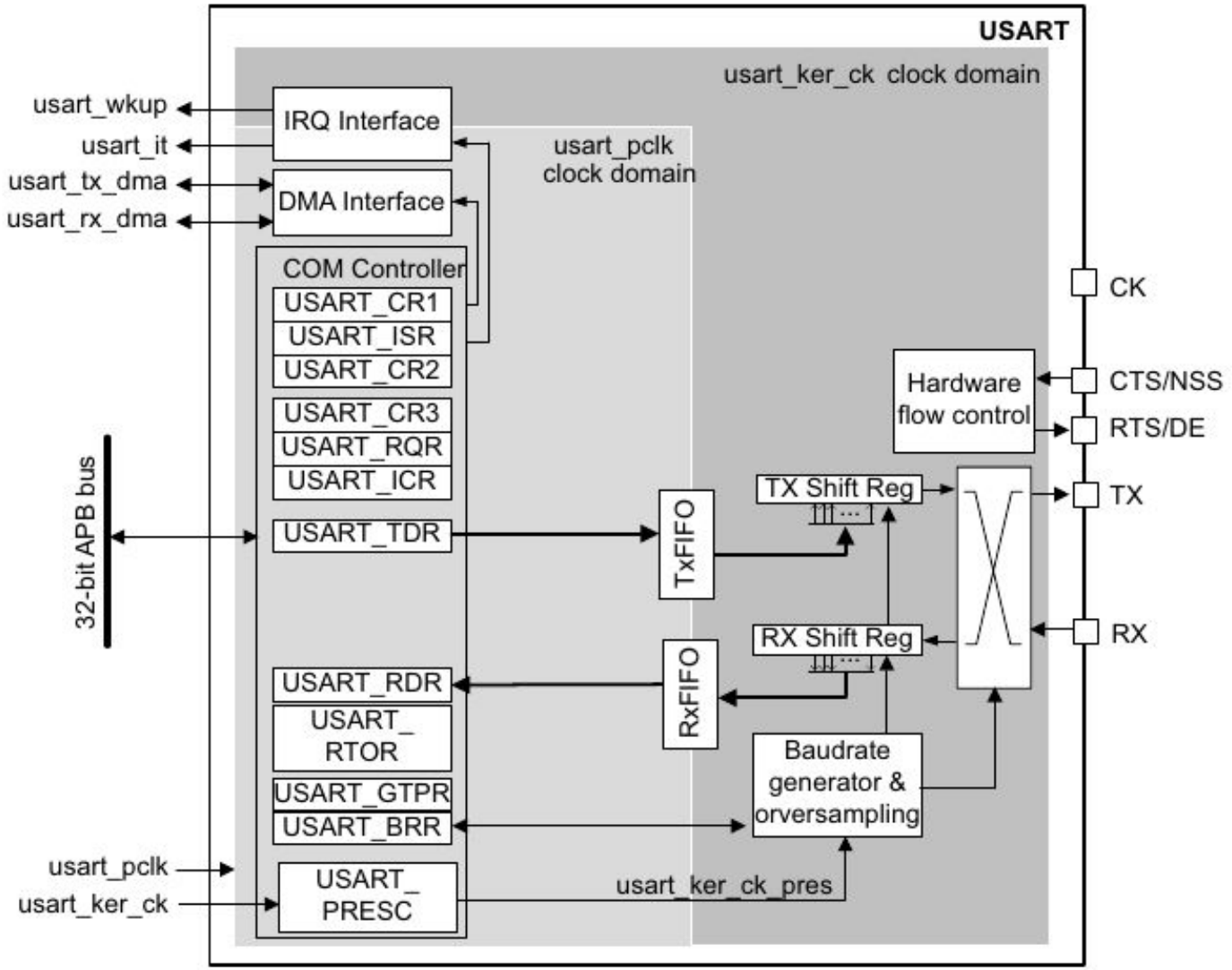


STM32 - Características do UART

- Comunicação Assíncrona *Full-Duplex*.
- Sistema de geração de *baudrate*.
- Duas FIFOs internas.
- Auto Detecção de *baudrate.*
- Tamanho do data de 7,8 e 9.
- Entre outras …



STM32G0 - Diagrama da USART



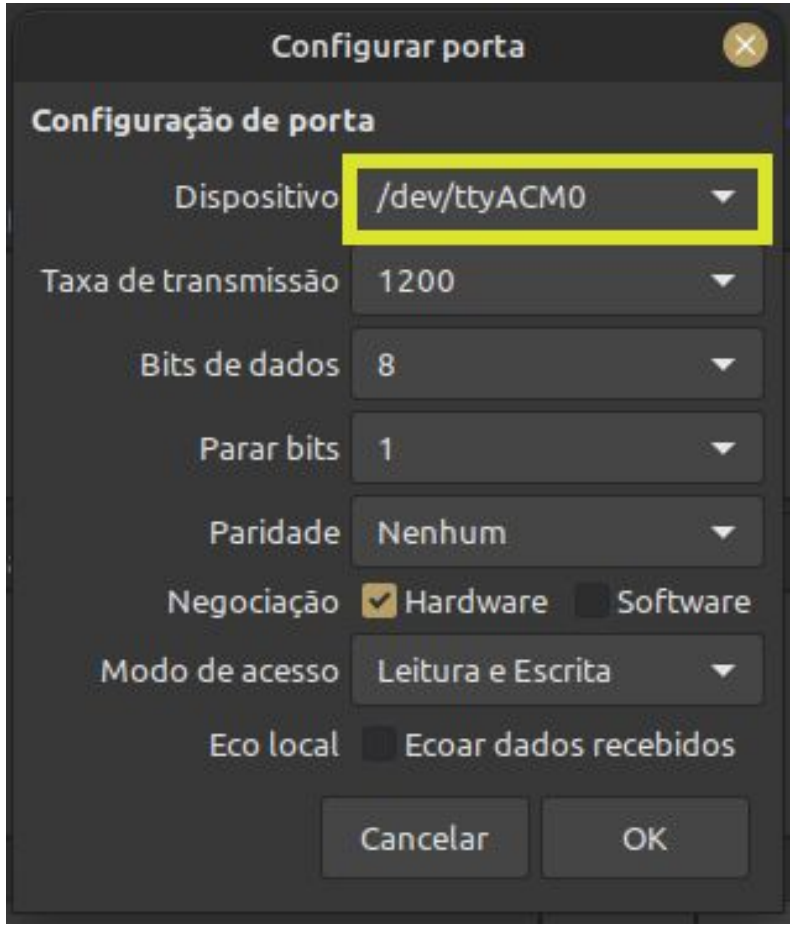
STM32G0 - Localização

Para localizar onde estão conectadas as UARTs, utilize o documento **STM32G0B1xB/xC/xE**, e consulte a **tabela 12**, na **página 47**. Onde é possível observar na coluna *Alternate Functions* onde tem terminais com a USART.

-	-	-	-	-	13	13	E3	15	21	K1	PC0	I/O	FT_a	-	LPTIM1_IN1, LPUART1_RX, LPTIM2_IN1, LPUART2_TX, USART6_TX, I2C3_SCL, COMP3_OUT	COMP3_INM7
-	-	-	-	-	14	14	F2	16	22	J3	PC1	I/O	FT_a	-	LPTIM1_OUT, LPUART1_TX, TIM15_CH1, LPUART2_RX, USART6_RX, I2C3_SDA	COMP3_INP1
-	-	-	-	-	15	15	G2	17	23	K2	PC2	I/O	FT	-	LPTIM1_IN2, SPI2_MISO/I2S2_MCK, TIM15_CH2, FDCAN2_RX, COMP3_OUT	-
-	-	-	-	-	16	16	H1	18	24	L1	PC3	I/O	FT	-	LPTIM1_ETR, SPI2_MOSI/I2S2_SD, LPTIM2_ETR, FDCAN2_TX	-

STM32G0 - Com. com o PC

Para comunicar com o computador, o *debugger* do kit implementa uma porta serial virtual conectada a **UART2**. Para acessar pelo computador, conectamos a porta **ttyACM0**, utilizando programas como *moserial*.



STM32G0B - Funções para UART

```
1   // Envia um array contendo os dados a serem enviados, onde huart refere
2   // ao handle da interface, pData e o ponteiro do array, Size e a
3   // quantidade de bytes de pData e Timeout e o tempo maximo para o
4   // dado ser transmitido em ms
5   HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t
    Size, uint32_t Timeout);
6
7   // Aguarda a recepcao de Size bytes pela serial, ate o tempo Timeout em
    ms
8   HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t
    Size, uint32_t Timeout);
```

STM32G0 - Experimentação

Vamos analisar o sinal da comunicação com o osciloscópio.

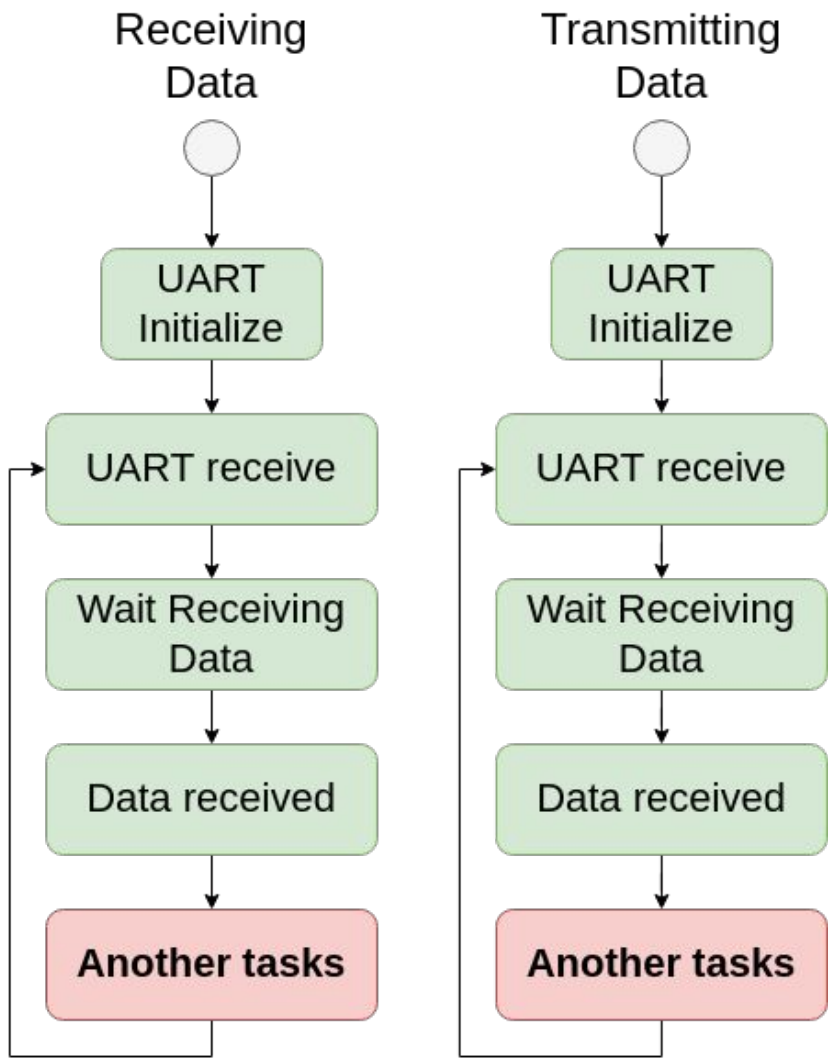
STM32G0 - Esperar?

PADO Labs

E se não puder esperar?

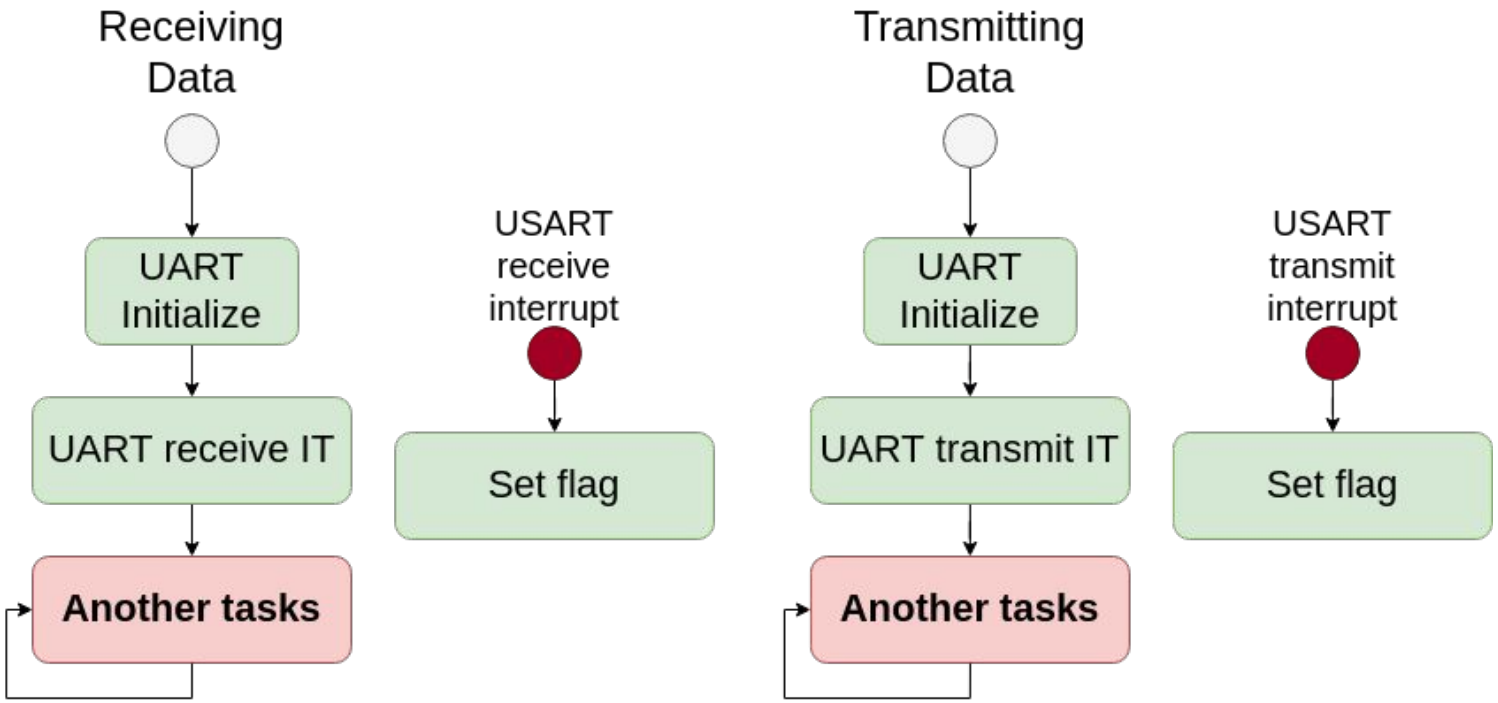
STM32G0 - Interrupção

Quando não utilizamos interrupção, o programa fica ocioso aguardando um recebimento ou transmissão.



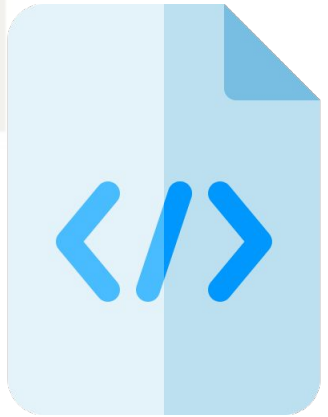
STM32G0 - Interrupção

Quando utilizamos interrupção, o programa pode seguir realizando as outras tarefas enquanto aguarda o recebimento através de uma interrupção.



STM32G0 - Funções

```
1  // Envia um array contendo os dados a serem enviados, onde huart refere
2  // ao handle da interface, pData e o ponteiro do array, Size e a
3  // quantidade de bytes de pData em modo de interrupcao
4  HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData,
   uint16_t Size);
5
6  // habilita a recepcao da serial em modo de interrupcao, aguardando o
   Size bytes
7  HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t
   Size);
```

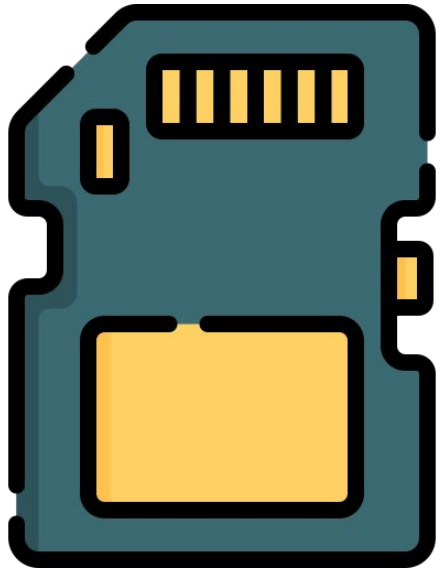


STM32G0 - Funções

```
1   // Callback que indica que o envio de dados foi completado, ou seja,
    todos
2   // os bytes foram enviados
3   void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
4
5   // Callback que indica que todos os bytes foram enviados
6   void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
```

STM32G0 - DMA

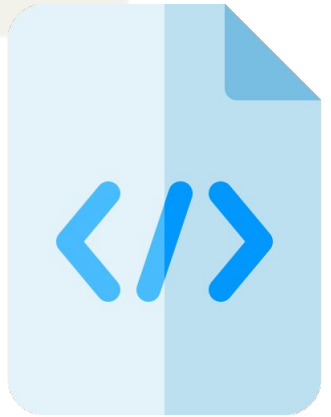
A USART permite também que seja utilizada o DMA para recepção de dados. No entanto, ele funciona melhor quando é necessário receber uma quantidade maior de dados.
O periférico irá gerar uma interrupção quando a quantidade de *bytes* for recebida.



STM32G0 - Funções

```
1  // Envia um array contendo os dados a serem enviados, onde huart refere
2  // ao handle da interface, pData e o ponteiro do array, Size e a
3  // quantidade de bytes de pData por DMA
4  HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, uint8_t *pData,
   uint16_t Size);
5
6  // habilita a recepcao da serial em modo de DMA, aguardando o Size
   bytes
7  HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *pData,
   uint16_t Size);
```

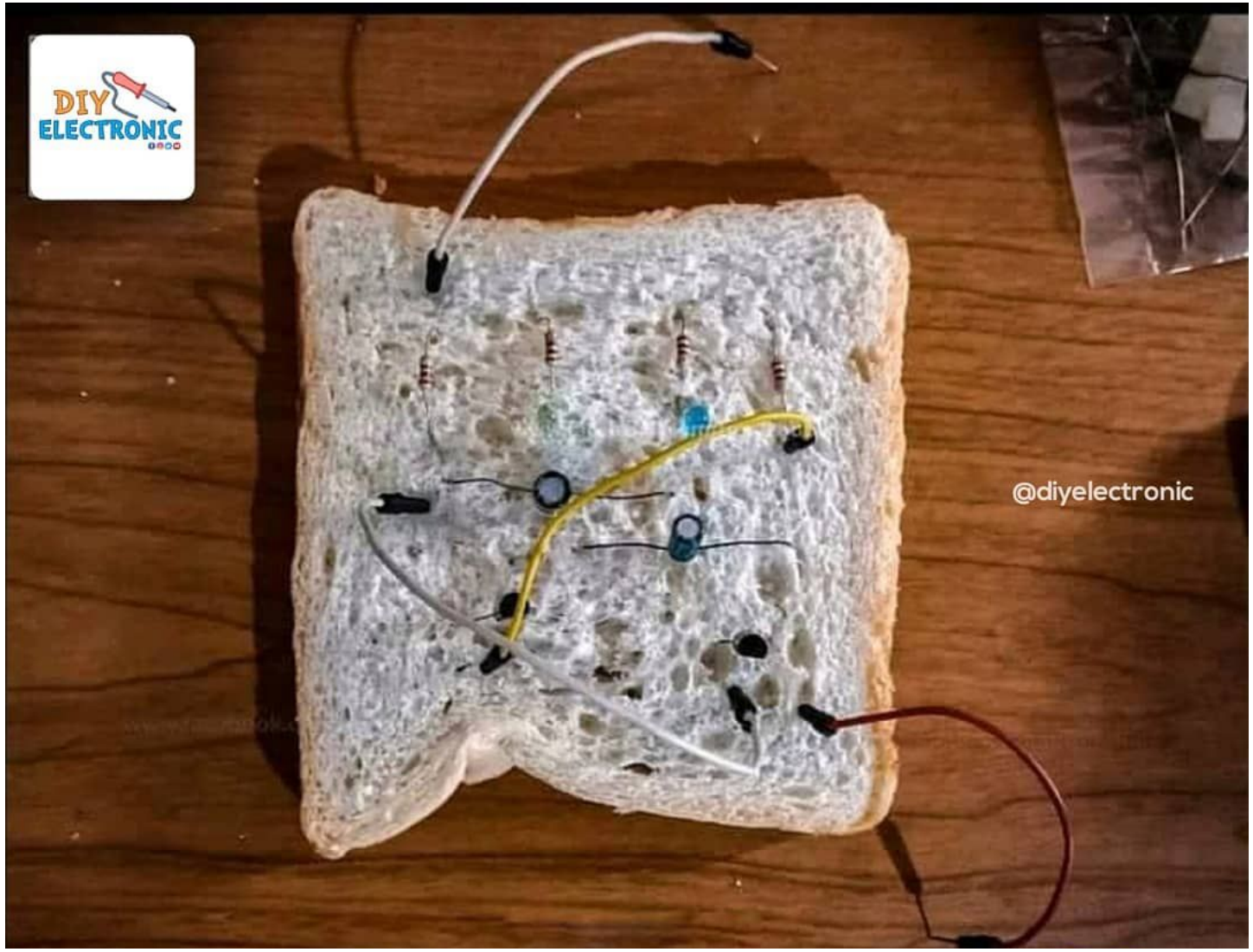
As funções de callback são as mesmas para o caso de interrupção.



STM32G0 - Mais Funções

Para consultar mais funções implementadas pelo **HAL**, utilize a documentação **UM2319:Description of STM32G0 HAL and low-layer drivers**, nos capítulos **50** e **51**.

Dúvidas ??



Referências

CAMPBELL, Scott. **Basics of UART Communication**. 2022. https://www.circuitbasics.com/basics-uart-communication/. Acesso em 26 de Janeiro de 2022.

MAGDY, Khaled. **STM32 USART / UART Tutorial**. 2020. https://deepbluembedded.com/stm32-usart-uart-tutorial/. Acesso 30 de Janeiro de 2022.

STMICROELETRONICS. **RM0444 - Reference Manual**. 5. ed. [S.l.], 2020. STM32G0x1 advanced Arm ® -based 32-bit MCUs.

__. **UM2319: Description of STM32G0 HAL and low-layer drivers**. 2. ed. [S.l.], 2020.

__. **STM32G0B1xB/xC/xE**. 2. ed. [S.l.], 2021. Arm ® Cortex ® -M0+ 32-bit MCU, up to 512KB Flash, 144KB RAM, 6x USART, timers, ADC, DAC, comm. I/Fs, 1.7-3.6V.

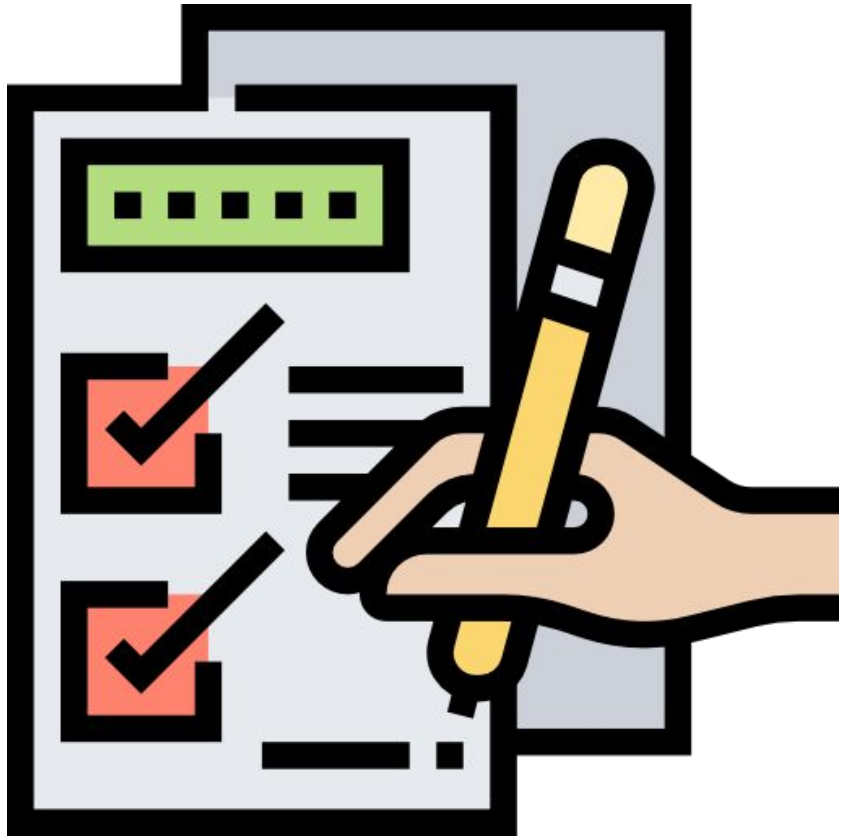
__. **UM2324 - User Manual**. 4. ed. [S.l.], 2021. STM32 Nucleo-64 boards (MB1360).

WIKIPEDIA. **Universal asynchronous receiver-transmitter**. 2022. https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter. Acesso em 25 de Janeiro de 2022.

Mão na Massa

PADO Labs

Lista de Exercícios #5



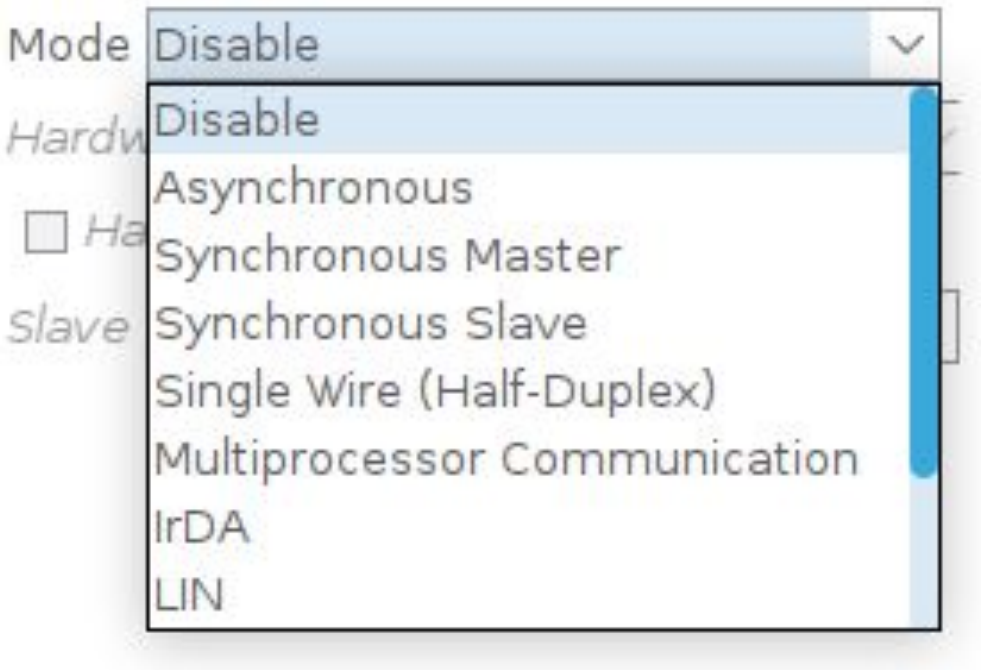
Comunicação UART

UART em modo polling

Selecionamos o periférico de USART na seção *Connectivity.*

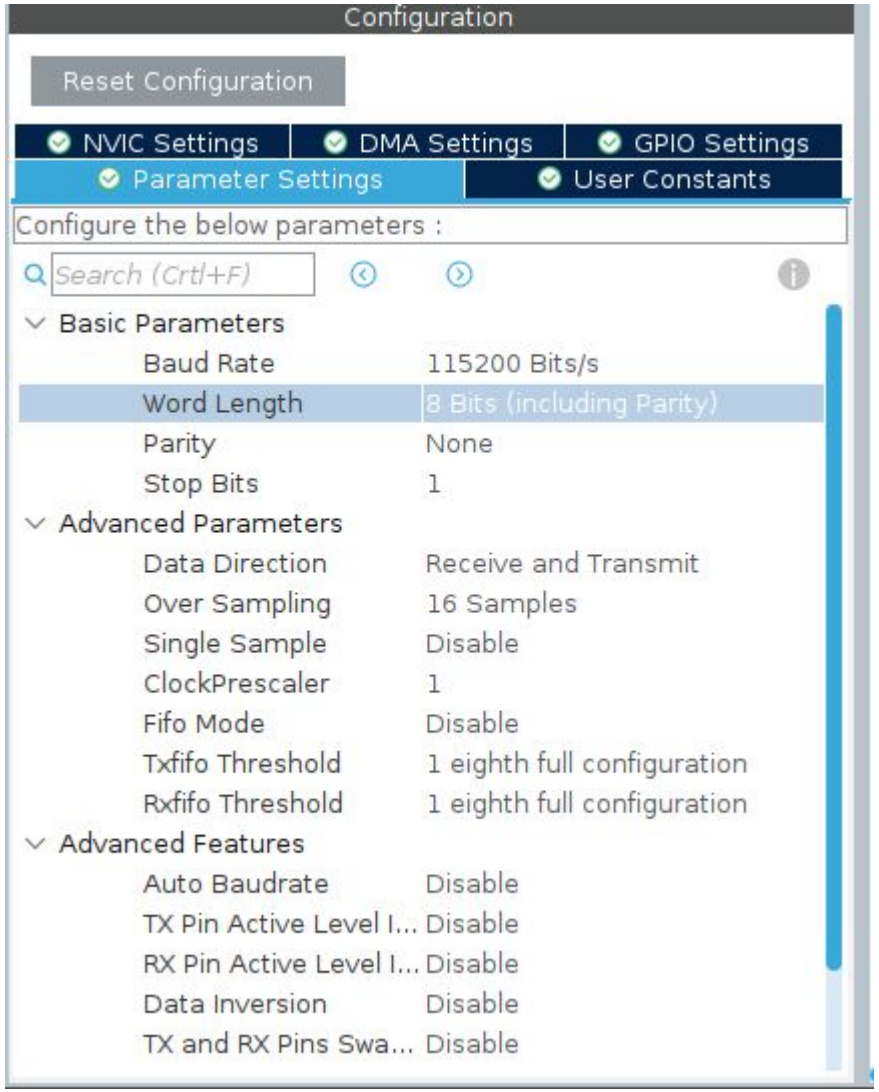
É possível também escolher os terminais pelos pinos do microcontrolador, escolhendo os terminais para UART e habilitando o periférico.

ADC em modo polling



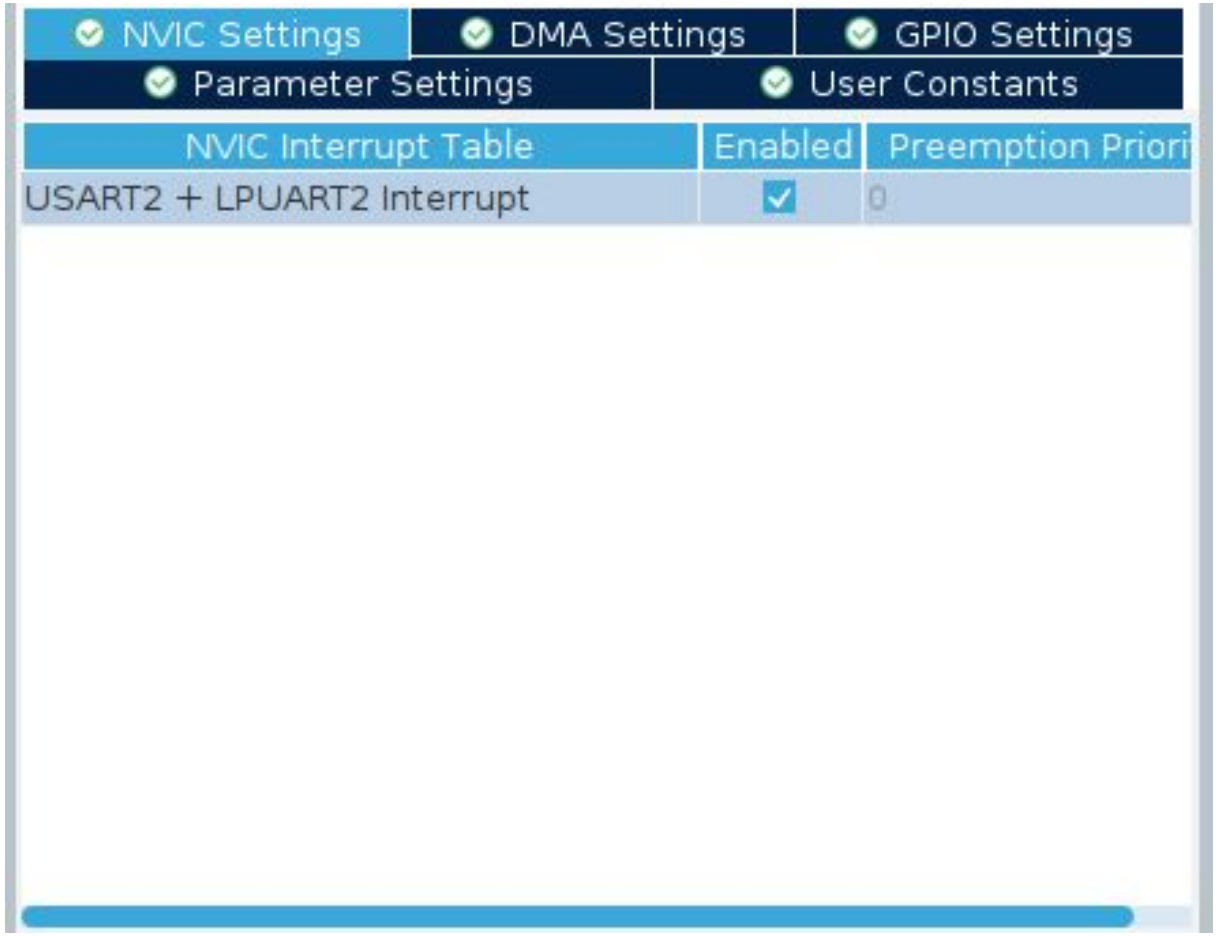
Selecionamento então *Asynchronous*.

ADC em modo polling



Na tela realizamos as configurações do periférico como *baudrate*, paridade, e outras configurações mais específicas.

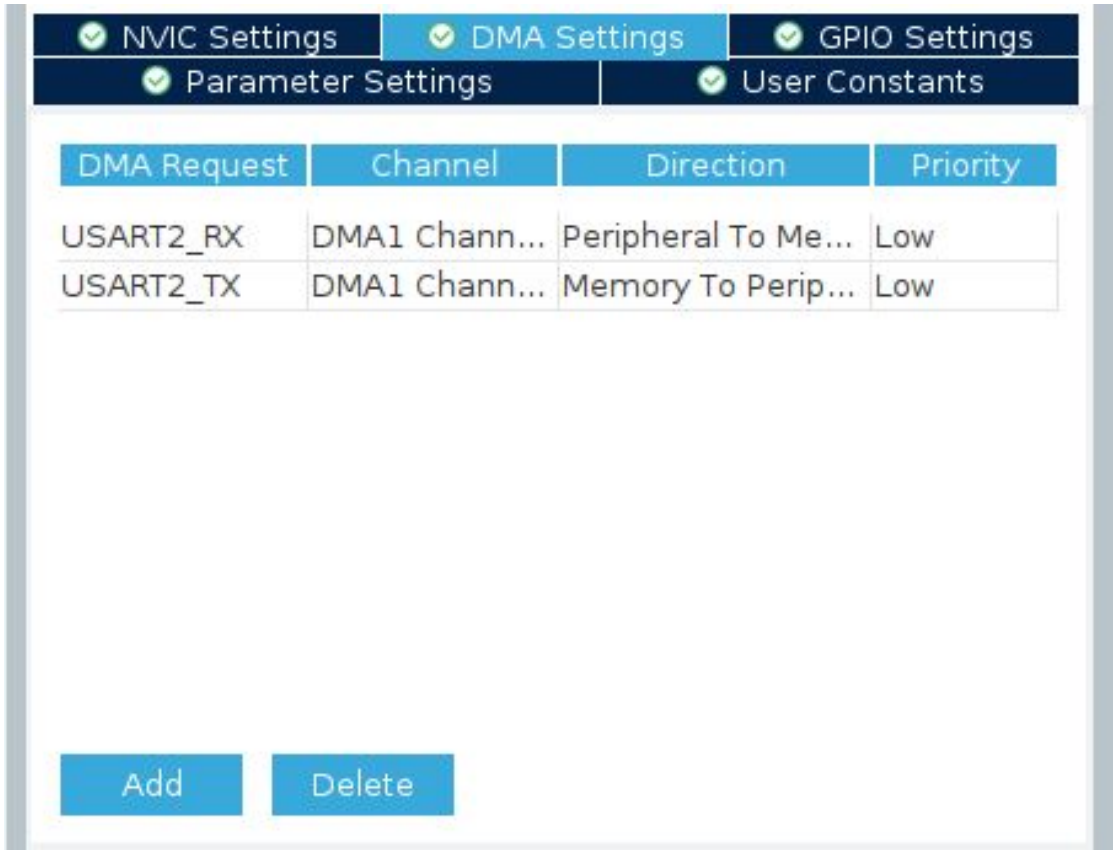
ADC em Interrupção



Após feitas as configurações anterior, vamos até a aba do NVIC Settings.

E habilitamos a interrupção para o USART.

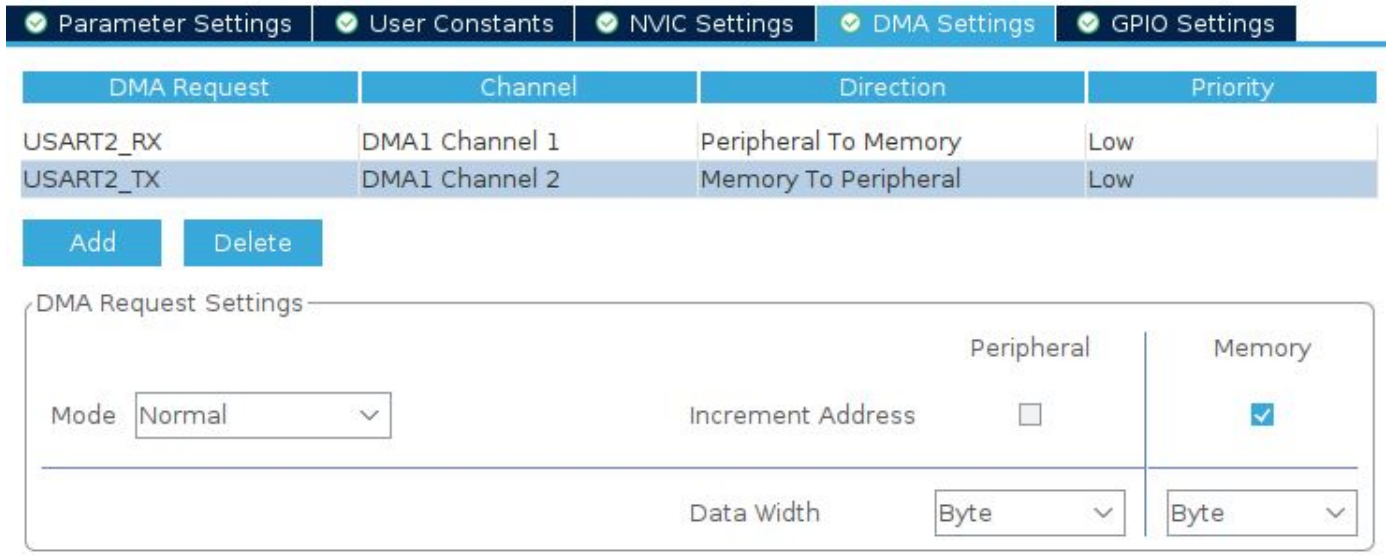
ADC em DMA



Vamos até a aba do DMA Settings.

Clicamos em Add e selecionamos os canais da UART que serão utilizados, sendo *USART2_RX* e/ou *USART_TX*.

ADC em DMA



Configuramos os canais de acordo com a figura ao lado.

PADO

Labs