

Sistemas Microcontrolados

UART

Prof. Guilherme Peron

Introdução

- Interface paralela x interface serial

Introdução

- Para que estudar a interface serial?
 - Há vários dispositivos externos que são interfaceados via comunicação serial
 - GPS
 - DAC
 - ADC
 - LCD
 - OLED
 - Impressoras
 - Outros microcontroladores

UART

- *Universal Asynchronous Receiver-Transmitter*
- É um modo de transmissão serial muito utilizado em microcontroladores
- Transmite dados de um microcontrolador para outro ou para um computador **podendo utilizar apenas dois fios** (RX/TX)
- É um sistema de comunicação **Full-Duplex**
- É muito utilizada para comunicação entre periféricos
- Fácil de utilizar

Funcionamento



- Quando parado, o pino de saída está no estado lógico **ALTO**;
- Cada transmissão de dados começa com um bit START, que é sempre estado lógico **BAIXO**;
- Cada pacote de dados tem 8 ou 9 bits de tamanho, onde o LSB é sempre o primeiro a ser transferido;
- Cada transmissão de dados termina com um bit de STOP, que tem sempre estado lógico **ALTO**.

Funcionamento



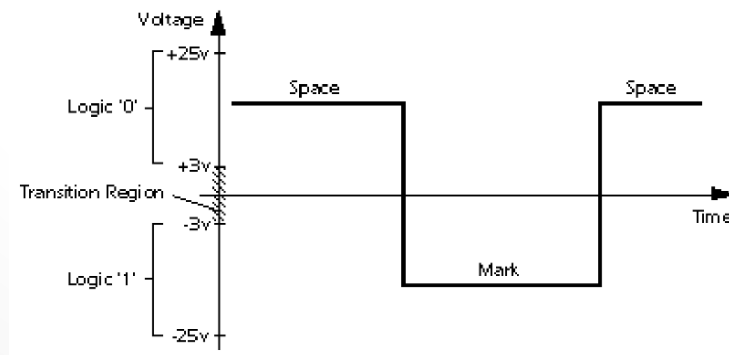
- Quando **não** há dados sendo transmitidos, a linha fica em nível lógico 1.
- O receptor reconhece a borda de descida do *Start Bit* e sincroniza seu *clock*.
- Após 1 ciclo e meio começa a fazer a leitura dos demais bits a cada *clock*.
- Se as frequências do transmissor e receptor estiverem perfeitamente sincronizadas, as leituras serão efetuadas exatamente no **meio** de cada ciclo.

Paridade

- Bit acrescentado ao dado, destinado a detecção de erros. A paridade simples detecta 1 erro, mas não corrige.
 - Paridade par: número **par** de bits no estado 1, incluindo o bit de paridade
 - Paridade ímpar: número **ímpar** de bits no estado 1, incluindo o bit de paridade
 - Exemplo: caractere ASCII 'A' é 0x41: 01000001b tem 2 bits em 1
 - Se for usada a paridade ímpar: acrescenta-se mais um bit '1' (3 bits '1' é ímpar): 01000001**1**b
 - Se for usada a paridade par: acrescenta-se mais um bit 0 (2 bits 1 é par): 01000001**0**b

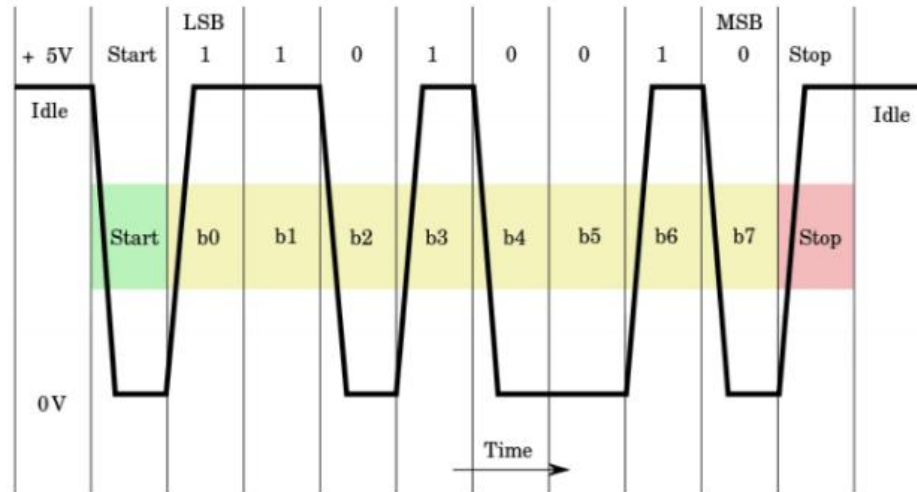
RS232-C

- O padrão da UART TTL opera de 0 a +5V
 - 0V: Nível lógico 0
 - 5V: Nível lógico 1
- Há, no entanto, o padrão RS232-C adotado pelos computadores que operam com tensões
 - Nível lógico 0: Entre +3V e +25V (usualmente +12V)
 - Nível lógico 1: Entre -3V e -25V (usualmente -12V)

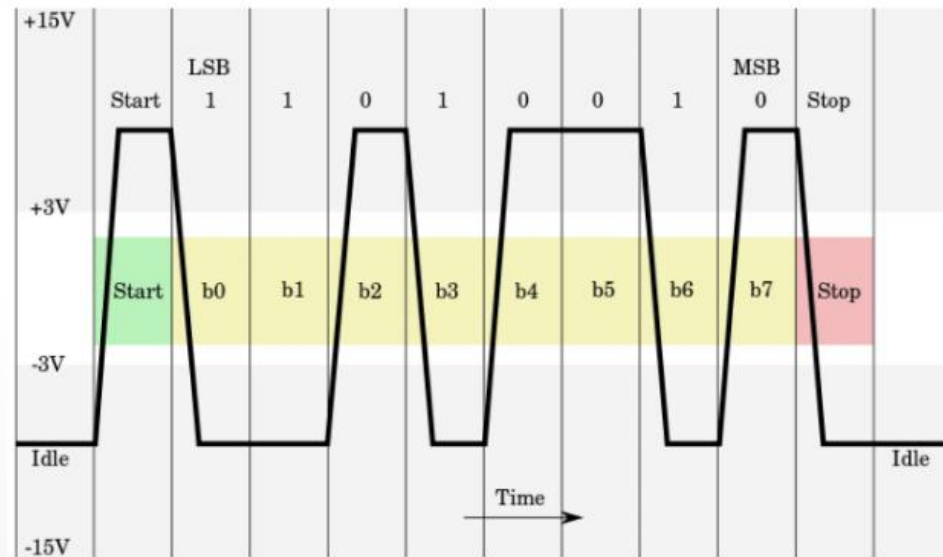


TTL x RS232

TTL



RS232



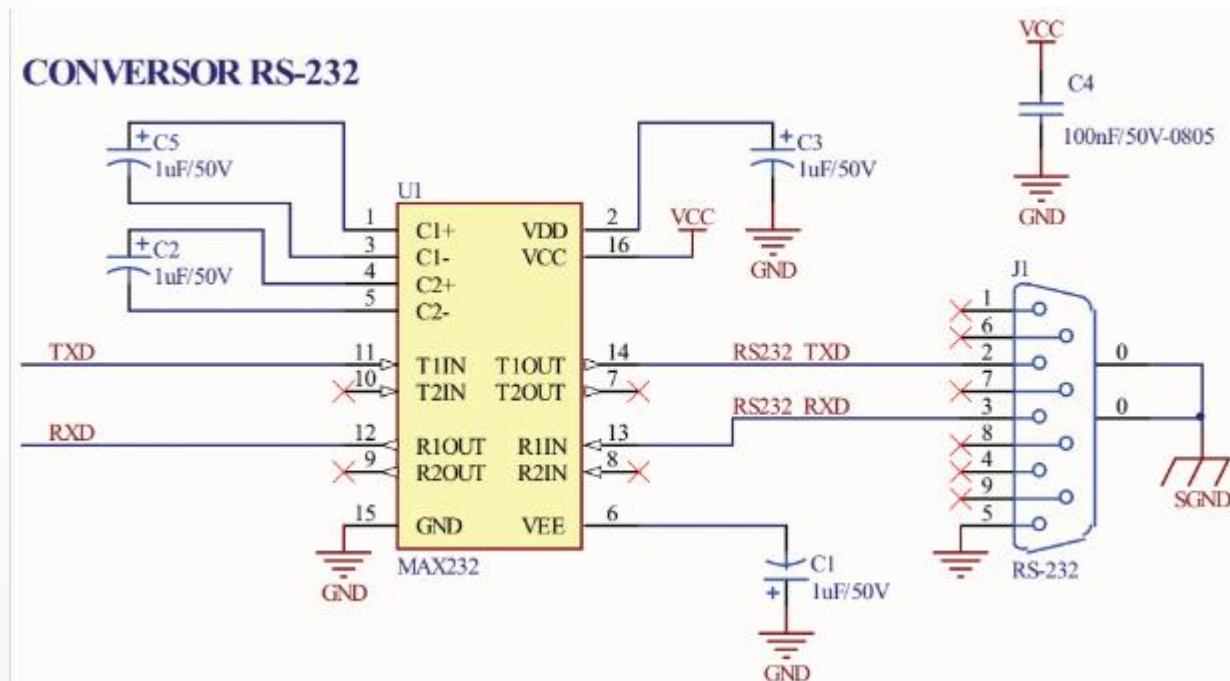
UART

Padrões de Config

- Velocidade (*Baud-Rate*) [bits/s]
 - 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Número de bits
 - 5, 6, 7 ou 8
- Paridade
 - Par, ímpar ou sem paridade
- Stop Bits
 - 1 ou 2

Camada Física

- Para conversão entre TTL e RS232, utiliza-se o circuito MAX232



Substituições

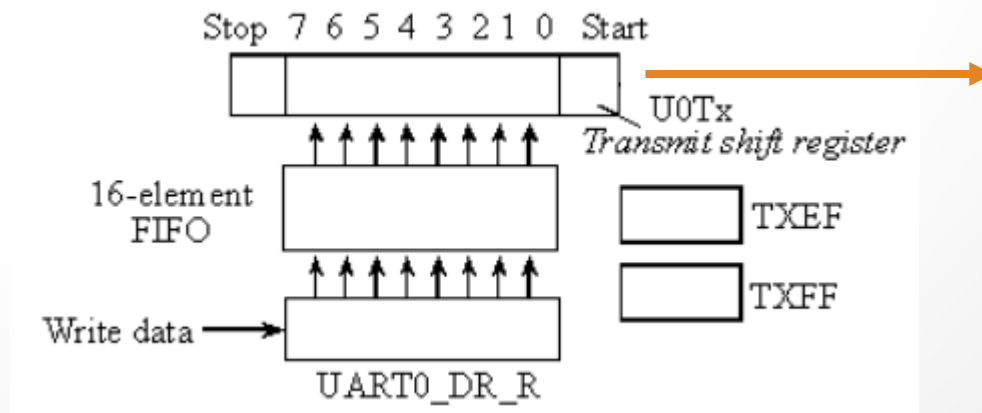
- Os novos computadores não possuem mais porta serial
- Utiliza-se conversores:
 - serial → USB
 - serial → bluetooth



UART na Tiva

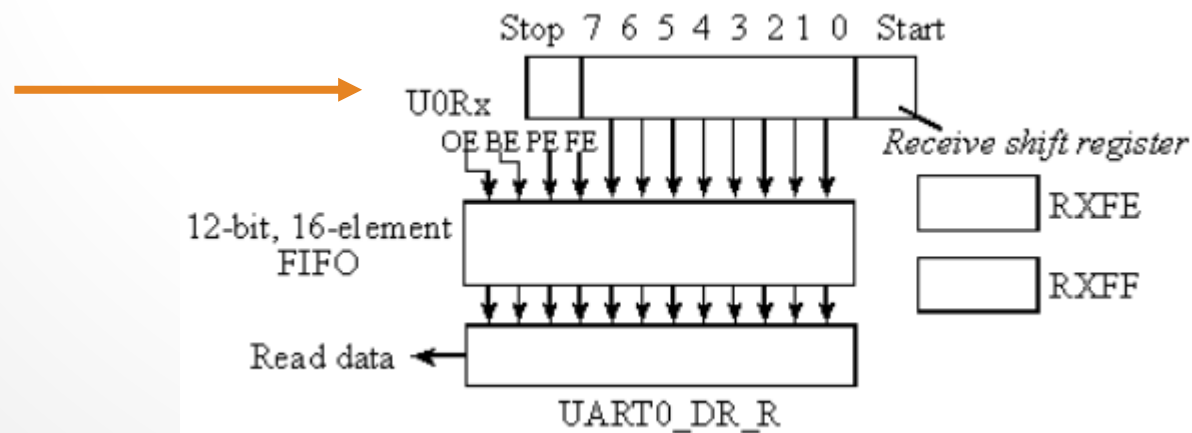
Transmissão UART

- Os transmissores têm uma FIFO de 16 elementos e um *shift register* de 10 bits, que não podem ser acessados diretamente pelo programador
- O SW deve verificar se a fila não está cheia e depois escrever os bits de transmissão no registrador de dados.
- Os bits são “shiftados”:
 - Start, b0, b1, b2, b3, b4, b5, b6, b7 e depois stop
- O registrador de transmissão é *write-only*.



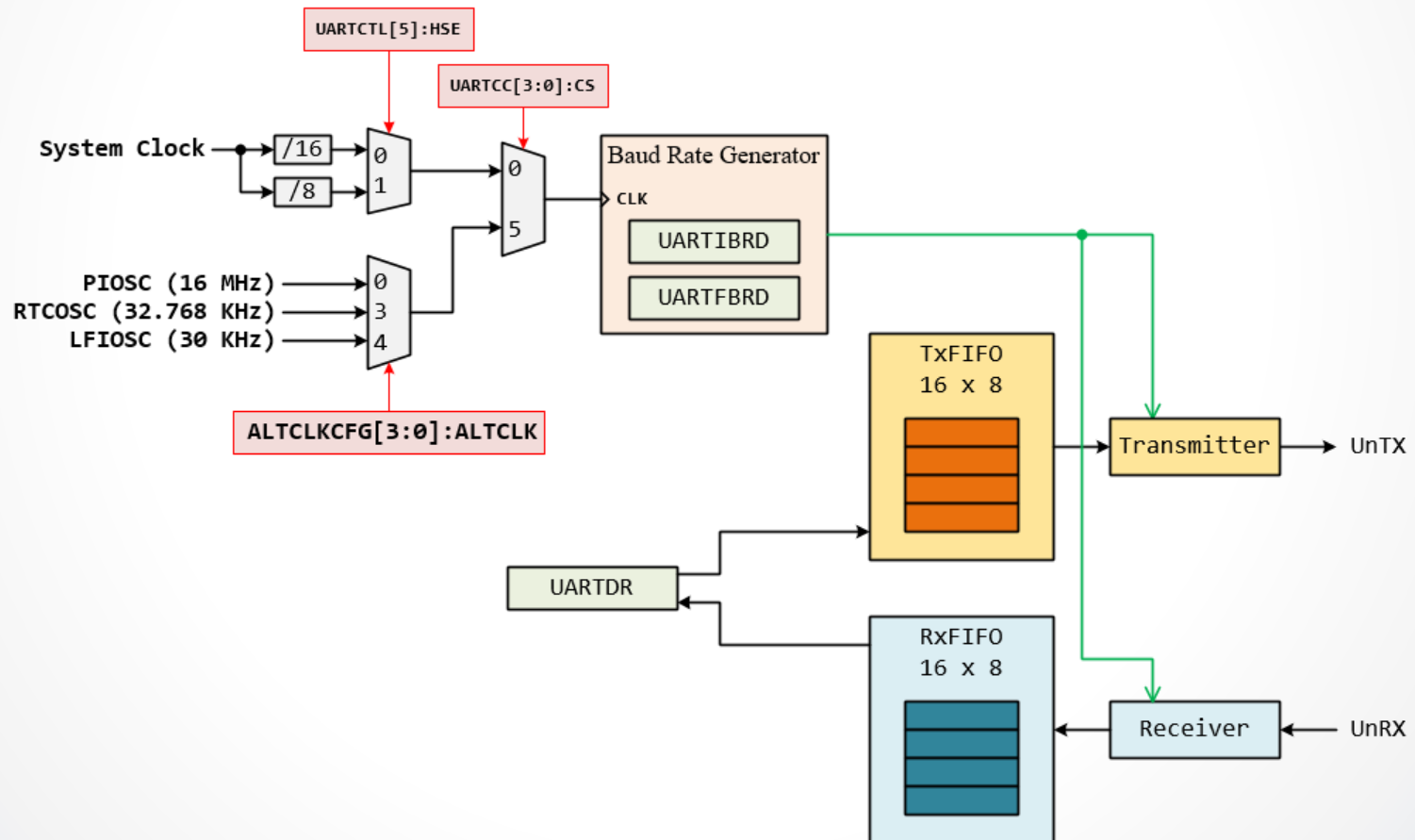
Recepção UART

- Os receptores têm uma FIFO de 16 elementos e um *shift register* de 10 bits, que não podem ser acessados diretamente pelo programador
- O registrador de recepção é *read-only*.
- O receptor reconhece um novo quadro pelo *start bit*
- 6 flags: 2 para a recepção em geral *FIFO empty* e *FIFO full*; 4 para cada elemento da FIFO (*overrun error*, *break error*, *parity error*, *framing error*)



Esquemático UART

- 8 UARTs



Registradores UART

- Para ativar a UART, o *clock* da respectiva UART tem que ser ativado no registrador **RCGCUART**. Cada bit ativa uma UART.
- Verificar o bit da UART respectiva no registrador **PRUART** para saber se está pronta para o uso.

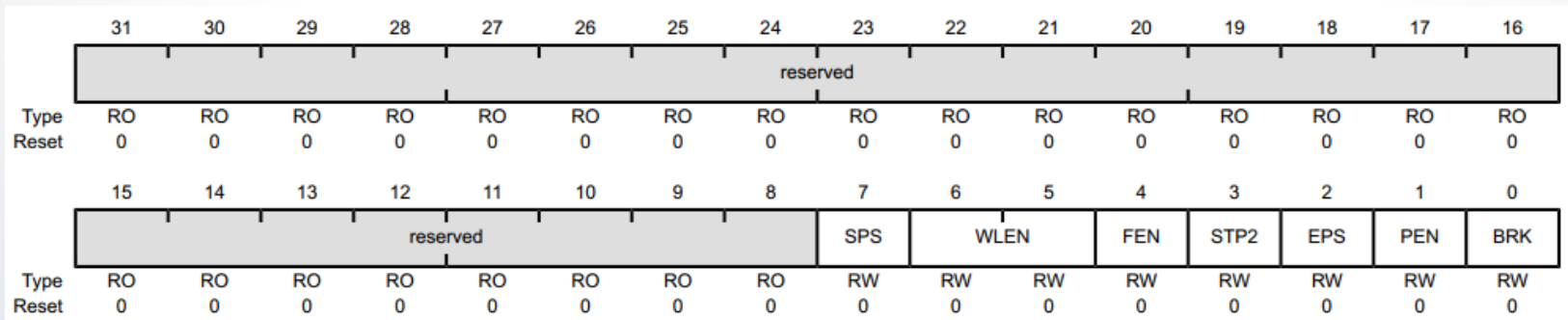
Registadores UART

- O registrador **UARTCTL** controla a UART:
 - **TXE** é o bit para habilitar o transmissor;
 - **RXE** é o bit para habilitar o receptor;
 - **UARTEN** habilita a UART como um todo. **Antes de realizar a configuração deve-se desabilitar este bit.**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CTSEN	RTSEN	reserved		RTS	DTR	RXE	TXE	LBE	reserved	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
Type	RW	RW	RO	RO	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

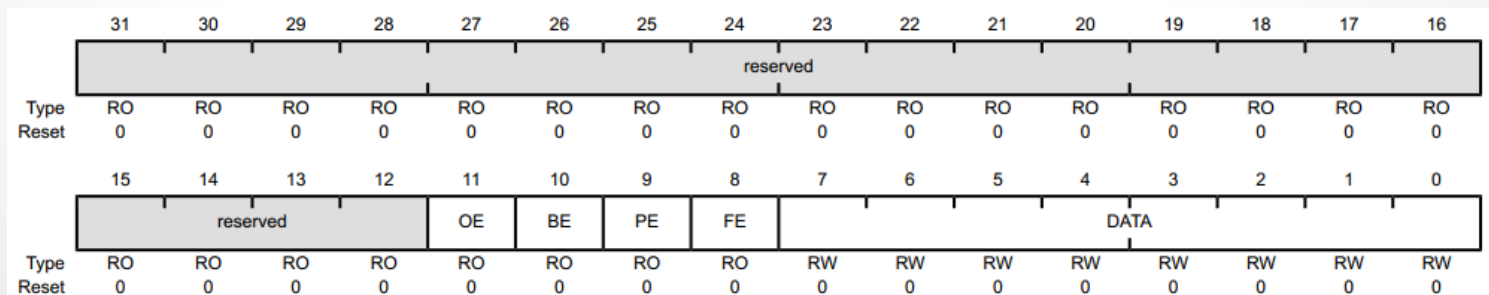
Registadores UART

- O registrador **UARTLCRH**:
 - **WLEN** controla o tamanho da palavra a ser enviada
 - 5 → 8 bits
 - **FEN** habilita as filas;
 - **STP2** habilita 2 *stop bits*;
 - **EPS** e **PEN** controla a paridade;



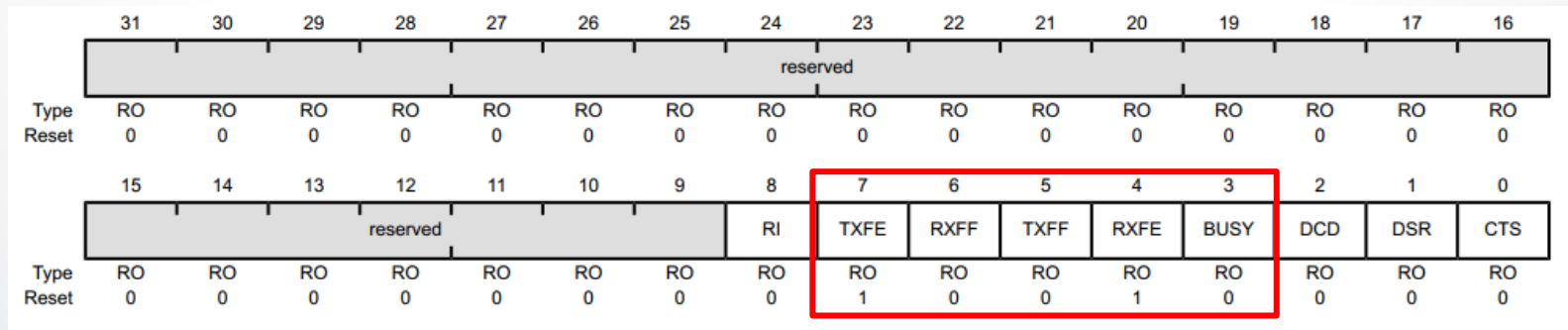
Registadores UART

- O registrador **UARTDR** é usado para transmitir o caractere (5 a 8 bits) ou para receber o caractere.
- Se escrever nele, uma transmissão é iniciada. Se ler dele, uma palavra recebida é lida.
- Ele também contém flags de controle:
 - **OE: *overflow*** → recepção realizada com a FIFO cheia;
 - **BE: *break*** → a linha de recepção ficou presa em 0 por um tempo de recepção completo (start a stop bit);
 - **PE: *parity*** → paridade recebida não corresponde
 - **FE: *frame*** → caractere recebido não tem um stop bit válido.



Registadores UART

- Os *status* das *flags* das FIFOs podem ser vistos no registrador **UARTFR**.
 - BUSY** é setado enquanto o transmissor ainda tiver bits sem enviar, mesmo que o transmissor estiver desabilitado. É limpo quando o último stop bit é enviado.
 - Para uma transmissão com *busy-wait* o bit **BUSY** deve ser testado continuamente para a próxima transmissão ocorrer.



Geração de *Baud-Rate*

- Para o Baud-Rate há dois registradores:

- **UARTIBRD**: 16 bits para a parte inteira;
- **UARTFBRD**: 6 bits para a parte fracionária.
 - Resolução de $1/(2^6) = 1/64$

$$\text{BRD} = \text{BRDI} + \text{BRDF} = \text{UARTSysClk} / (\text{ClkDiv} * \text{Baud Rate})$$

- ClkDiv é 16 se o bit **HSE** do **UARTCTL** for 0, ou 8 se o bit for 1.

Exemplo:

Para um *baud rate* de 19200bps e um clock de 80MHz.

$$\text{BRD} = 80.000.000 / (16 * 19.200) = 260,4167$$

$$\text{UARTIBRD} = 260$$

$$\text{UARTFBRD} = \text{BFDF} * 64 = 0,4167 * 64 = 26,67 = 27$$

Geração de *Baud-Rate*

- **ATENÇÃO:** uma alteração no divisor de baud-rate deve ser seguida por uma escrita no registrador **UARTLCRH** para as alterações fazerem efeito.

Passo-a-passo (UART)

1. Habilitar o clock no módulo UART no registrador **RCGCUART** (cada bit representa uma UART) e esperar até que a respectiva UART esteja pronta para ser acessada no registrador **PRUART** (cada bit representa uma UART).
2. Garantir que a UART esteja desabilitada antes de fazer as alterações (limpar o bit **UARTEN**) no registrador **UARTCTL** (Control).
3. Escrever o *baud-rate* nos registradores **UARTIBRD** e **UARTFBRD**

Passo-a-passo (UART)

4. Configurar o registrador **UARTLCRH** para o número de bits, paridade, stop bits e fila
5. Garantir que a fonte de *clock* seja o *clock* do sistema no registrador **UARTCC** escrevendo 0 (ou escolher qualquer uma das outras fontes de *clock*)
6. Habilitar as flags RXE, TXE e URTEN no registrador **UARTCTL** (habilitar a recepção, transmissão e a UART)

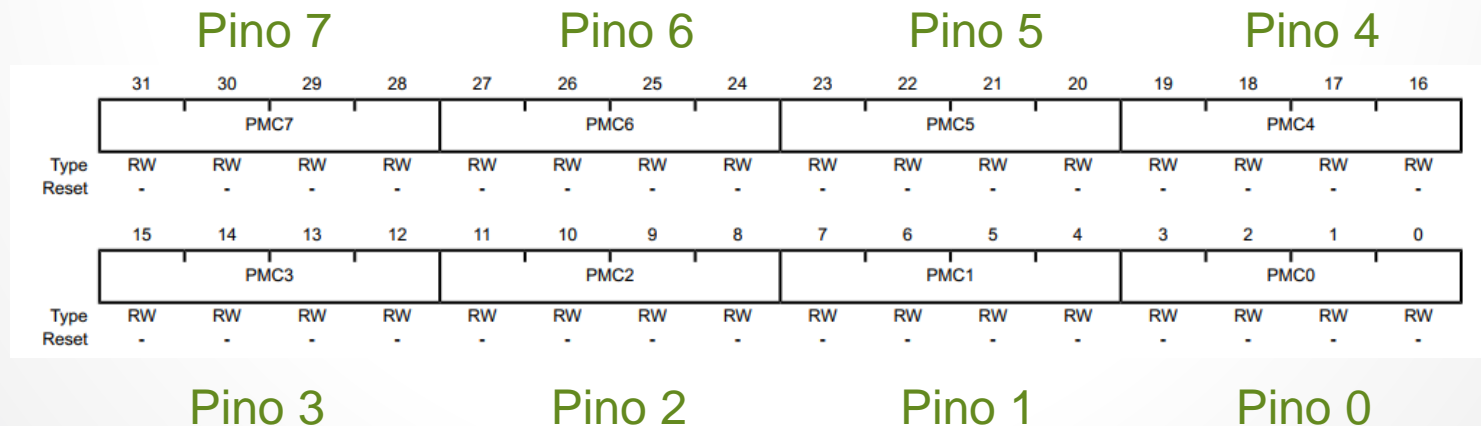
Passo-a-passo (GPIO)

7. Habilitar o clock no módulo GPIO no registrador **RCGGPIO** (cada bit representa uma GPIO) e esperar até que a respectiva GPIO esteja pronta para ser acessada no registrador **PRGPIO** (cada bit representa uma GPIO).
8. Desabilitar a funcionalidade analógica no registrador **GPIOAMSEL**.

Passo-a-passo (GPIO)

- Escolher a função alternativa dos pinos respectivos TX e RX no registrador **GPIOCTL** (verificar a tabela 10-2 no datasheet páginas 743-746)

Lembrando que cada 4 bits deste registrador configura um pino.



Passo-a-passo (GPIO)

9. Escolher a função alternativa dos pinos respectivos TX e RX no registrador **GPIOCTL** (verificar a tabela 10-2 no datasheet páginas 743-746)

Por exemplo a UART0 da placa está mapeada nos pinos PA0 e PA1:

IO	Pin	Analog or Special Function ^a	Digital Function (GPIOCTL PMCx Bit Field Encoding) ^b											
			1	2	3	4	5	6	7	8	11	13	14	15
PA0	33	-	U0Rx	I2C9SCL	TOCCP0	-	-	-	CAN0Rx	-	-	-	-	-
PA1	34	-	U0Tx	I2C9SDA	TOCCP1	-	-	-	CAN0Tx	-	-	-	-	-

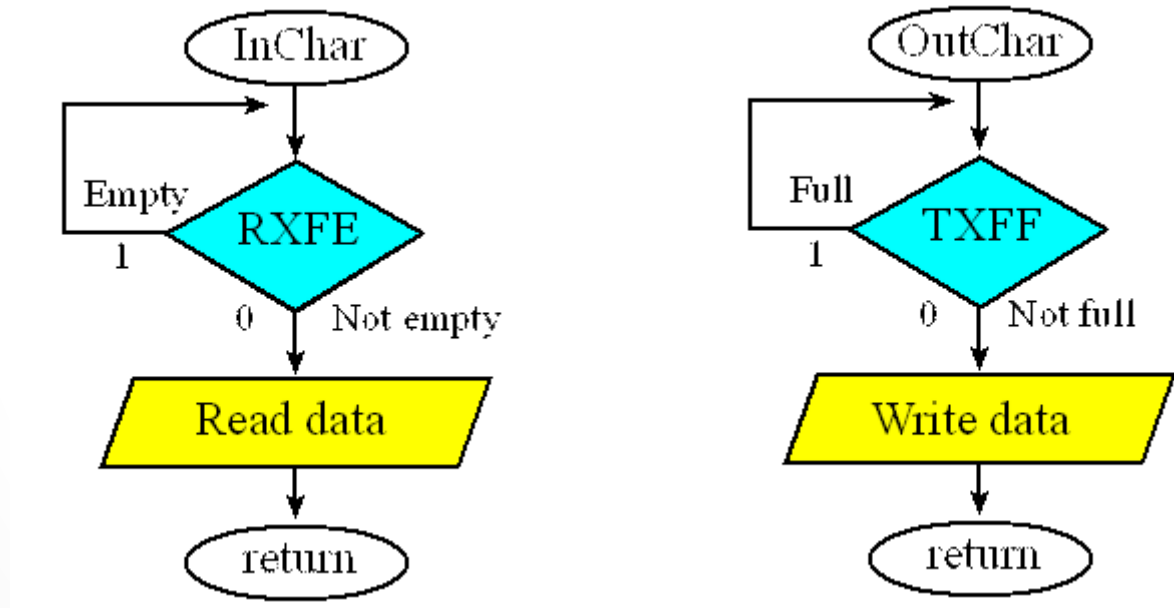
Assim, como a função alternativa do pino 0 e pino 1 é a 1 de cada um então, o registrador **GPIOCTL** do Port A (**GPIO_PORTA_AHB_PCTL_R**) deve receber o valor 0x11 (00010001 em binário).

Passo-a-passo (GPIO)

10. Habilitar os bits de função alternativa no registrador **GPIOAFSEL** nos pinos respectivos à UART.
11. Configurar os pinos como digitais no registrador **GPIODEN**.

Recepção e Transmissão

- Para escrita e leitura da porta serial por *busy-flag*:



Recepção e Transmissão

- Para fazer a recepção, criar uma função que:
 - Realiza *polling* no bit **RXFE (0x10)** (*FIFO empty*) do registrador **UARTFR**;
 - Enquanto for 1, não há dados para serem lidos;
 - Quando for 0, o conteúdo do registrador **UARTDR** para uma variável de 8 bits.
- Para fazer a transmissão, criar uma função que:
 - Realiza *polling* no bit **TXFF (0x20)** (*FIFO full*) do registrador **UARTFR**;
 - Enquanto for 1, não pode transmitir / FIFO cheia;
 - Quando for 0, copiar o byte a ser enviado para o registrador **UARTDR**.

TM4C1294

- Na Tiva, a **UART0** está conectada aos pinos **PA0 (U0Rx)** e **PA1 (U0Tx)** e que por sua vez já passam por um conversor USB
- Mesma interface do *debugger*