Microcontroladores EL66A

UART

Prof. Guilherme Peron

Introdução

- Interface paralela x interface serial

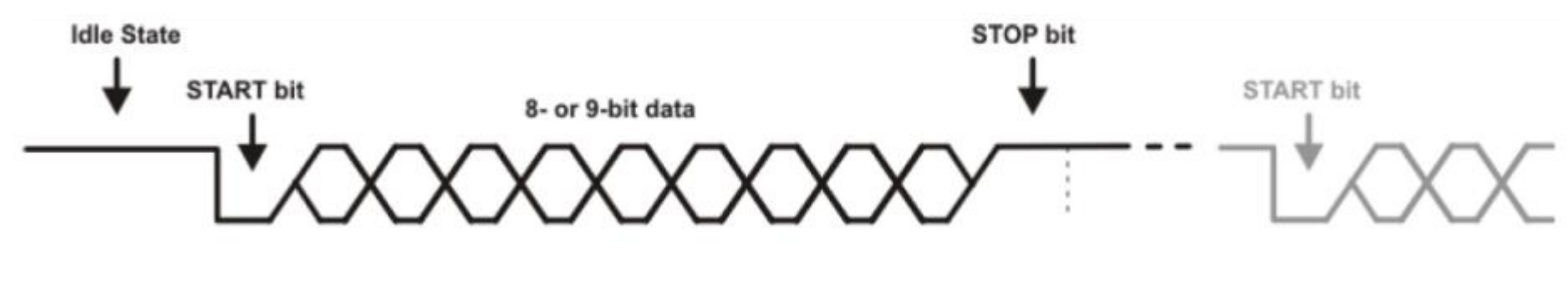
Introdução

- 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

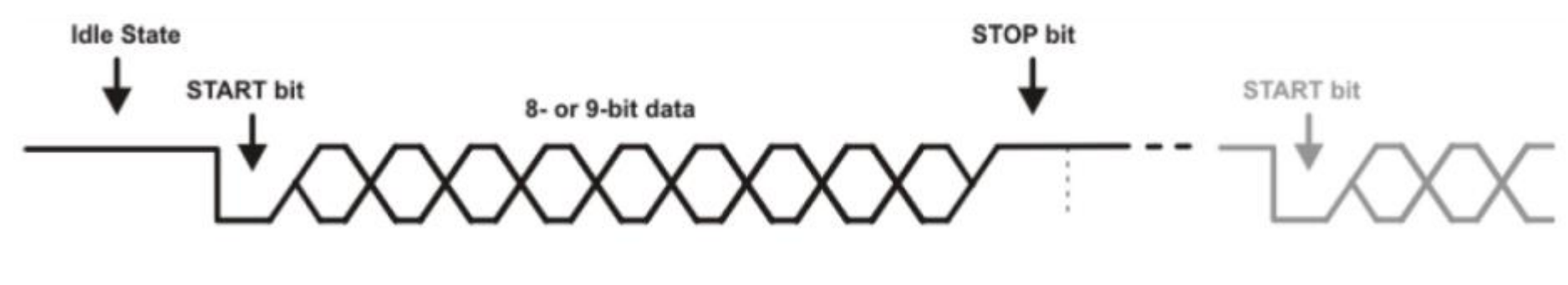
- *Universal Asynchronous Receiver-Transmitter*
- É um modo de transmissão serial muito utilizado em microcontroladores
- Transmite dados de um microcontrolador para outro ou para um computador podendo utilizar apenas dois fios (RX/TX)
- É um sistema de comunicação *Full-Duplex*
- É muito utilizada para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



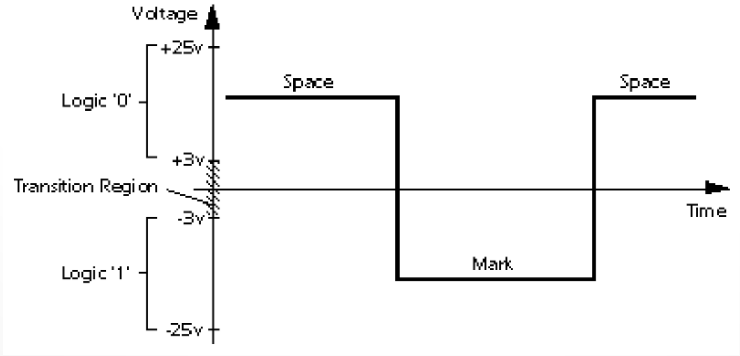
- Quando não há dados sendo transmitidos, a linha fica em nível lógico 1.
- O receptor reconhece a borda de descida do *Start Bit* e sincroniza seu *clock*.
- Após 1 ciclo e meio começa a fazer a leitura dos demais bits a cada *clock*.
- Se as frequências do transmissor e receptor estiverem perfeitamente 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): 010000011b
    - Se for usada a paridade par: acrescenta-se mais um bit 0 (2 bits 1 é par): 010000010b

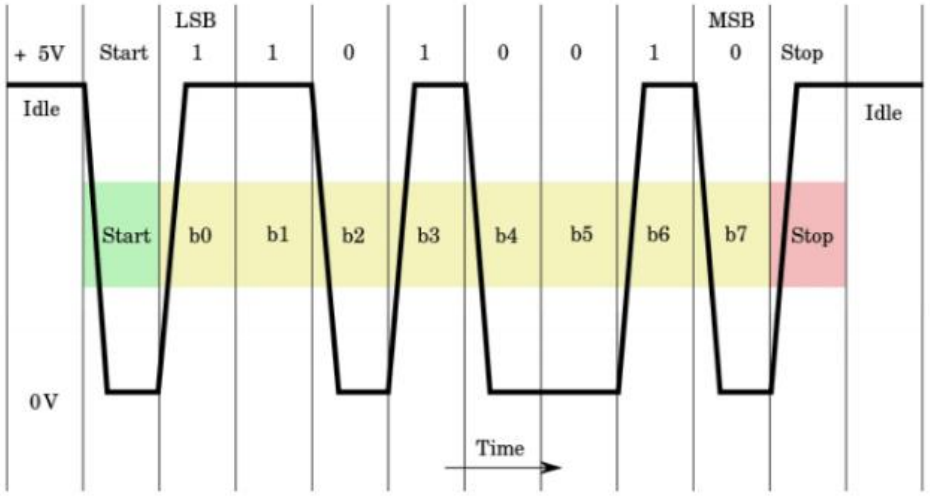
RS232-C

- O padrão da UART TTL opera de 0 a +5V
  - 0V: Nível lógico 0
  - 3,3V: 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)

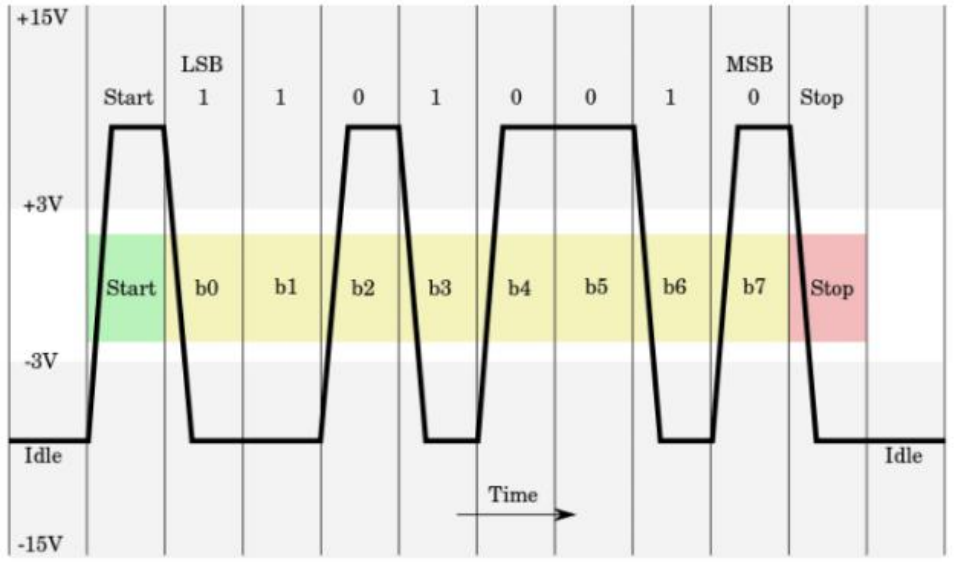


TTL x RS232

TTL



RS232

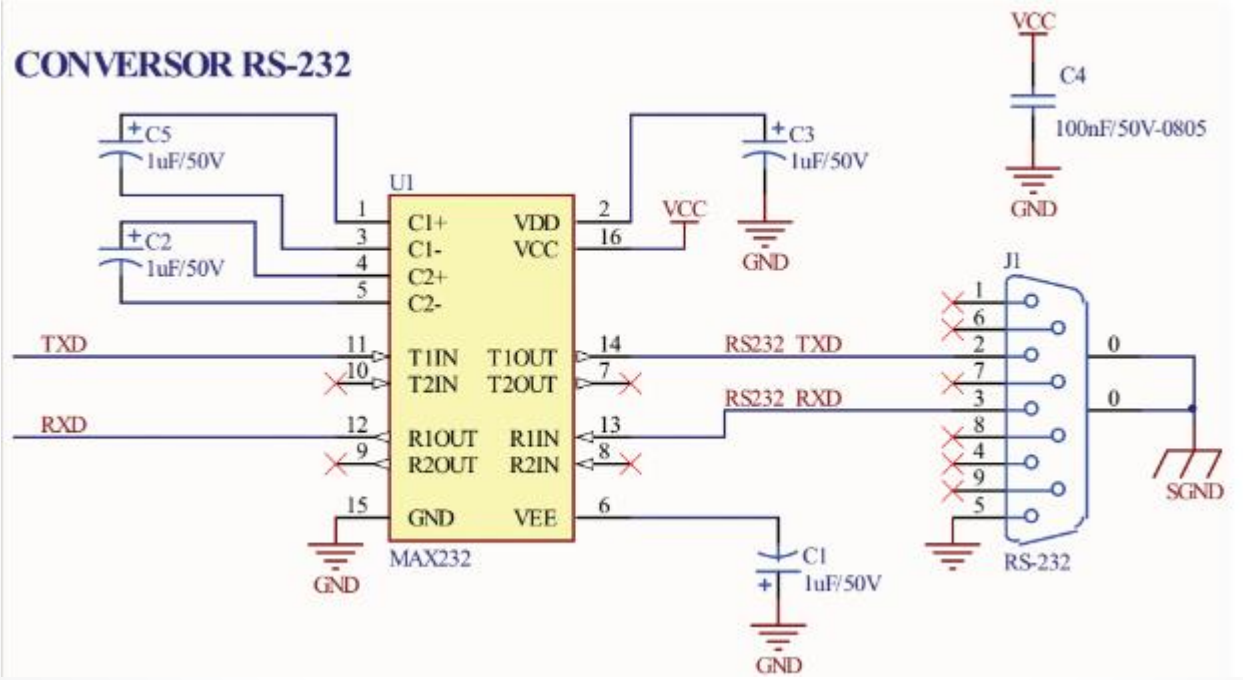


Padrões de Config

- 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



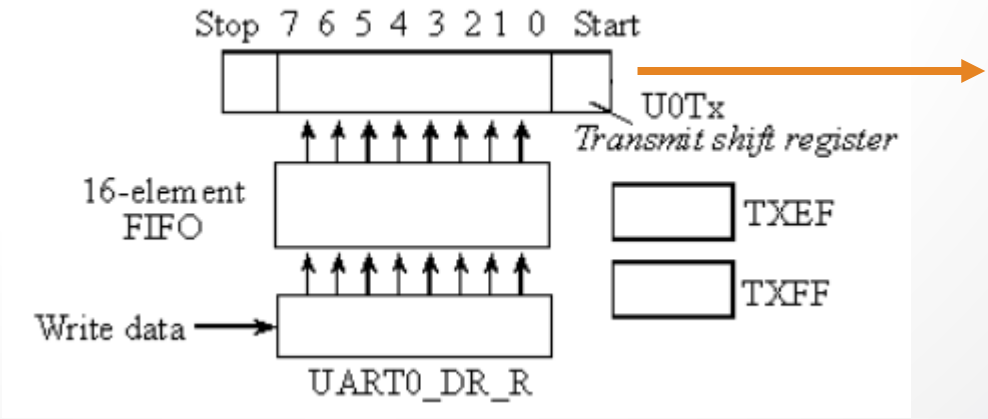
Substituições

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

UART na Tiva

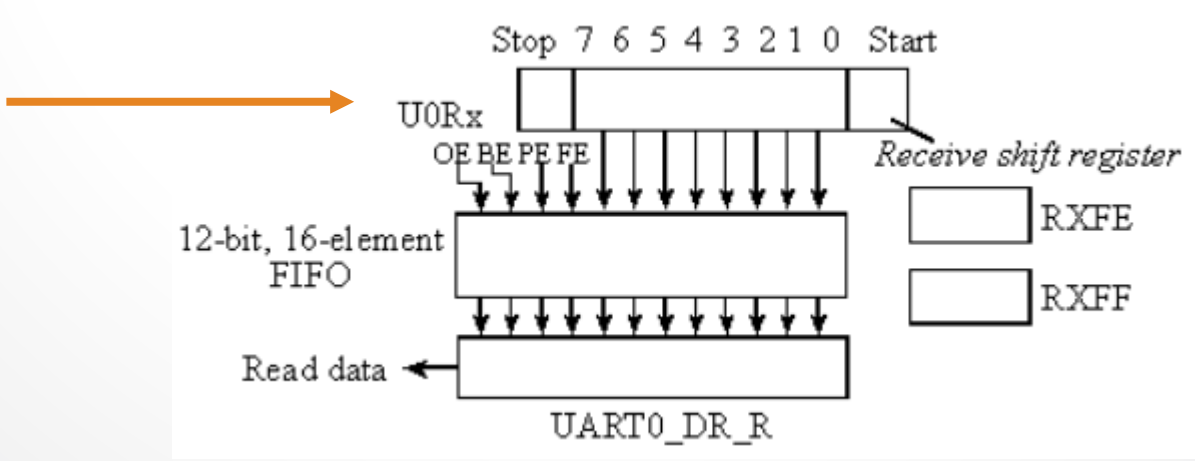
Transmissão UART

- 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.*



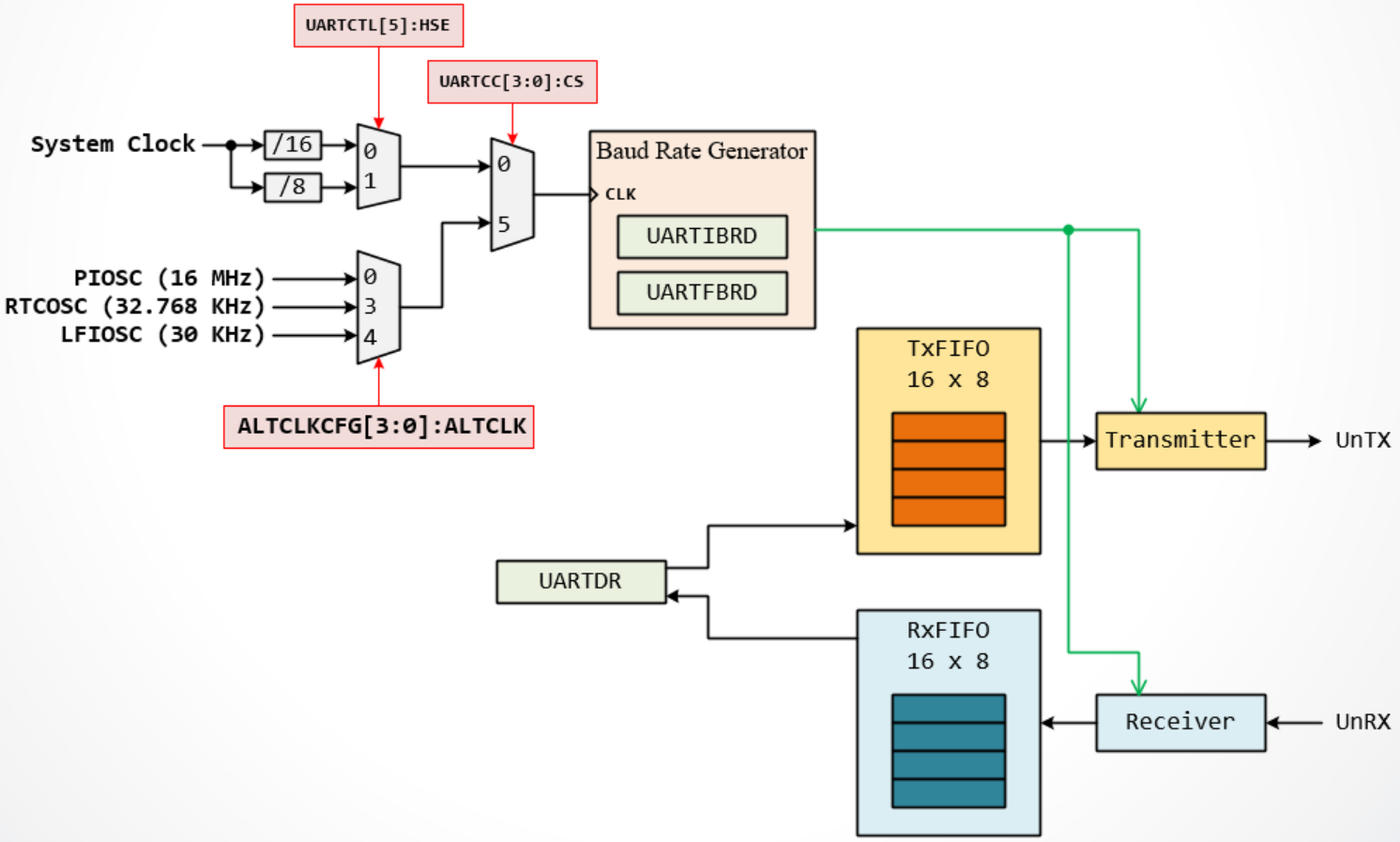
Recepção UART

- Os recpetores 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)*



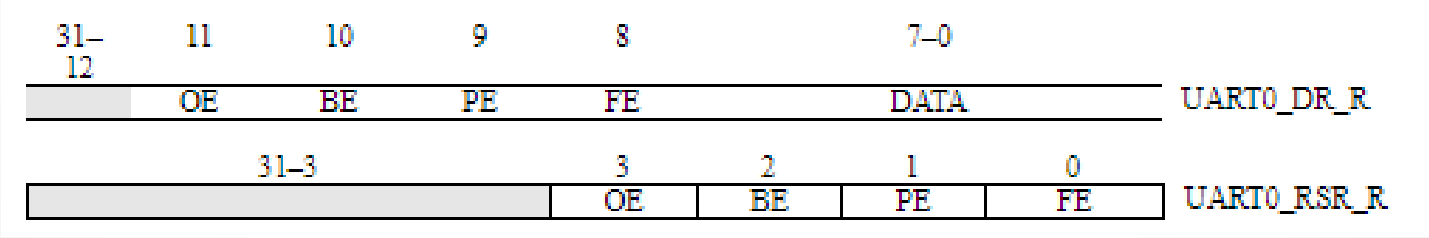
Esquemático UART

- 8 UARTs



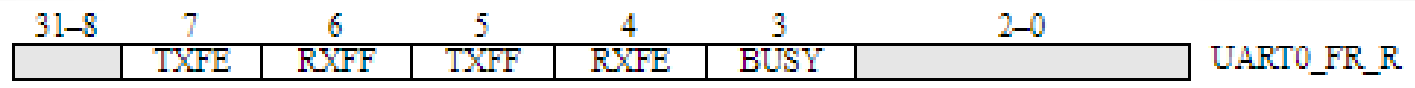
Registradores UART

- Para ativar a UART, o *clock* da respectiva tem que ser ativado no registrador RCGCUART
- Verificar o bit da UART respectiva no registrador PRUART para saber se está pronta para o uso.
- Os *flags* OE, BE, PE e FE podem ser vistos em em cada byte recebido no registrador UARTDR ou no registrador UARTRSR. O *software* pode limpar esses flags escrevendo qualquer valor no registrador UARTRSR



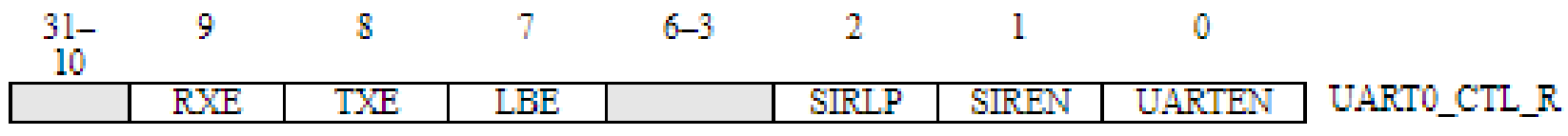
Registradores UART

- 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.



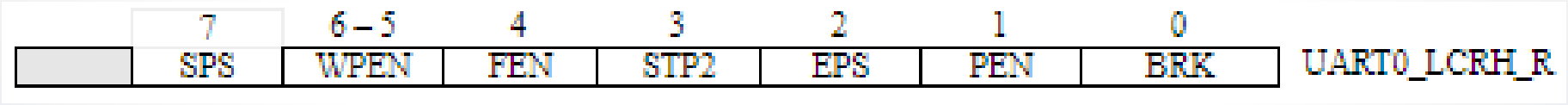
Registradores UART

- 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.**



Registradores UART

- O registrador UARTLCRH:
  - WPEN controla o tamanho da palavra a ser enviada
    - 3 → 8 bits
  - FEN habilita as filas;
  - STP2 habilita 2 *stop bits*;
  - EPS e PEN controla a paridade;



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 $1/(2^6) = 1/64$

BRD = BRDI + BRDF = UARTSysClk / (ClkDiv * *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.

BRD = 80.000.000/(16*19.200) = 260,4167

UARTIBRD = 260

UARTFBRD = 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 UARTEN no registrador UARTCTL (habilitar a recepção, transmissão e a UART)

Passo-a-passo (GPIO)

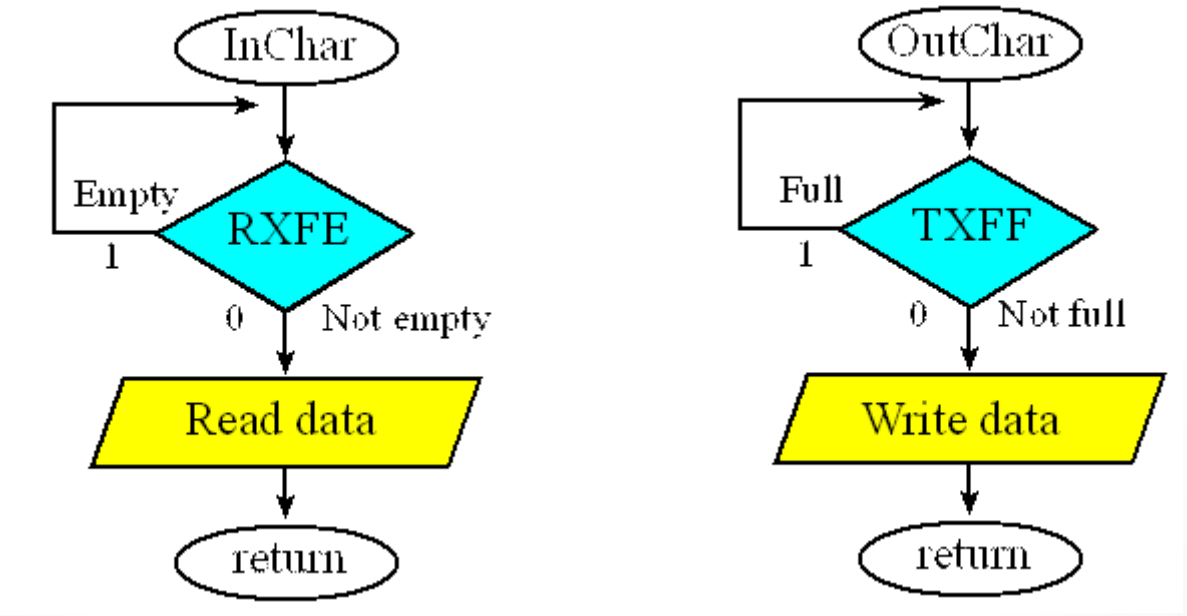
7. Habilitar o clock no módulo GPIO no registrador RCGGPIO (cada bit representa uma GPIO) e esperar até que a respectiva GPIO esteja pronta para ser acessada no registrador PRGPIO (cada bit representa uma GPIO).
8. Desabilitar a funcionalidade analógica no registrador GPIOAMSEL.
9. Escolher a função alternativa dos pinos respectivos TX e RX no registrador GPIOPCTL (verificar a tabela 10-2 no datasheet páginas 743-746)

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
  - Realizar *polling* no bit RXFE *(FIFO empty)* do registrador UARTFR, esperar enquanto for 0;
  - Quando for 1, o conteúdo do registrador UARTDR para uma variável de 8 bits.

- Para fazer a transmissão
  - Realizar *polling* no bit TXFF *(FIFO full)* do registrador UARTFR;
  - Quando for 1, copiar o byte a ser enviado para o registrador UARTDR.

TM4C1294

- Na Tiva, a UART0 está conectada aos pinos PA0 (U0Rx) e PA1 (U1Tx) e que por sua vez já passam por um conversor USB
- Mesma interface do *debugger*