# Teste – B2 – 04/11/2021

*Pontuação: 3,0 pontos e 25% da frequência*

**Setup**

**A prova tem duas questões e ambas devem ser feitas em cima do código main.c fornecido. Esse código tem duas funções, clockInit() e pmmVCore(), que não devem ser modificadas. Essas funções inicializam os clocks como:**

| | |
|---|---|
| **ACLK** | **16.384 Hz** |
| **SMCLK** | **4 MHz** |
| **MCLK** | **25 MHz** |

**É recomendado colocar todas as funções no mesmo arquivo main.c .**

**QUESTÃO 1 – UART**

A principal tarefa dessa questão é escrever um transmissor e um receptor na UART. A primeira parte envolve o transmissor enquanto a segunda parte envolve o receptor.

Em todos os casos, a configuração do UART deve ser:

| |
|---|
| Baud Rate 57600 bps |
| LSB First |
| Paridade Ímpar |
| 8 bits de dados |
| 2 stop bits |

O transmissor **DEVE** utilizar o UCA0 (onde P3.3 é o TX). O código deve enviar 3 bytes (0xE7, 0x49 e 0x07) quando o botão esquerdo for pressionado e (0xB2, 0x38 e 0xFA) quando o botão direito for pressionado.

| Ação | 1º byte | 2º byte | 3º byte |
|---|---|---|---|
| P2.1 (esquerdo) | 0xE7 | 0x49 | 0x07 |
| P1.1 (direito) | 0xB2 | 0x38 | 0xFA |

O receptor **DEVE** ser configurado no UCA1 (no pino P4.2 para o RX) . Quando a sequência (0xE7, 0x49 e 0x07) for recebida o receptor deve acender o LED vermelho (sem modificar o LED verde) e quando a sequência (0xB2, 0x38 e 0xFA) for recebida o receptor deve acender o LED verde (sem modificar o LED vermelho). Qualquer outra sequência de 3 bytes deve apagar os LEDs. Para configurar o UCA1 você deve utilizar :

```
P4SEL |= BIT2; //Disponibilizar P4.2
PMAPKEYID = 0X02D52; //Liberar mapeamento de P4
P4MAP2 = PM_UCA1RXD; //P4.2 = RXD
```

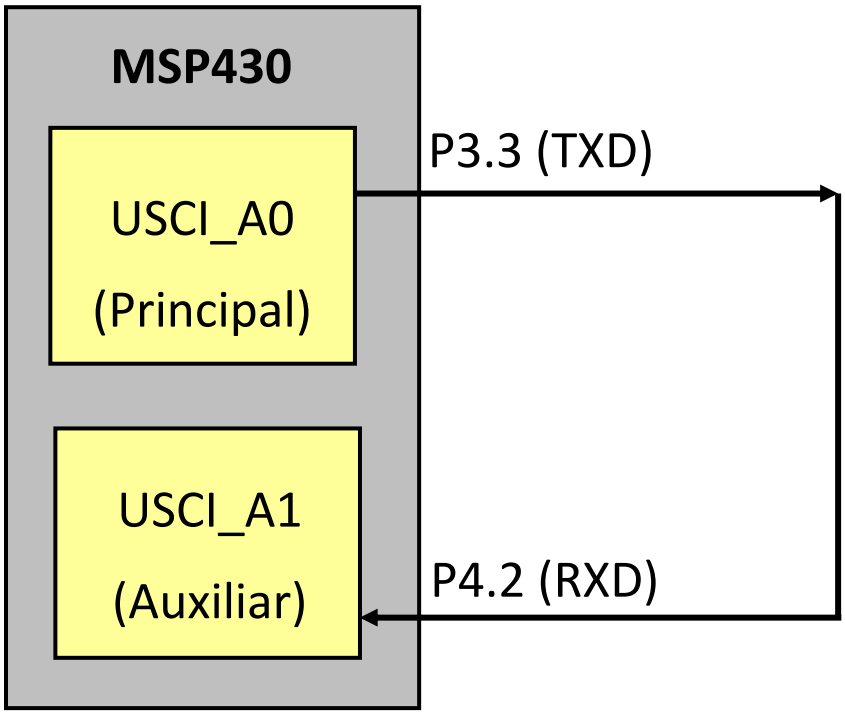
A entrega deve ser feita em 3 programas:

Questão 1 a (3 pontos): O programa deve ser entregue completo, com o transmissor e receptor implementados pelo aluno no mesmo arquivo.

Questão 1 b (1 ponto): O programa deve ser desmembrado e enviado apenas o transmissor na UCA0. A UCA1 e os LEDs não devem ser inicializados ou utilizados.

Questão 1 c (1 ponto): O programa deve ser desmembrado e enviado apenas o receptor na UCA1, junto com o controle dos LEDs. A UCA0 e as chaves não devem ser inicializadas ou utilizadas.

Um diagrama da ligação dos dados no MSP é mostrada abaixo:



**QUESTÃO 2 – ADC e PWM**

Escreva um programa para fazer uma leitura da tensão no ADC no pino 6.0. O ADC deve ser configurado para fazer uma leitura de 8 bits. O ADC deve fazer 32 leituras por segundo automaticamente, sem nenhuma intervenção, preenchendo um vetor unsigned char data_buffer[32] . Quando o buffer for preenchido (i.e., após a leitura de 32 amostras) o programa deve consolidar a leitura em um único valor unsigned int dado_consolidado.

$$dado_{consolidado} = \frac{\sum_{i=0}^{31} data\_buffer[i]}{32}$$

A variável dado_consolidado só pode ter seu valor modificado uma vez por segundo. Essa variável deve ser utilizada para modular um PWM no LED verde. O PWM deve usar uma frequência base de 400 kHz. Quanto maior o valor lido, mais forte o LED deve acender, de forma linear.

Para que o aluno obtenha a pontuação máxima o gatilho do ADC deve ser automático, via Timer A0. O dado convertido deve ser capturado via interrupção.

Da mesma forma, para obter a pontuação máxima, o PWM deve funcionar sem nenhuma intervenção (a não ser, claro, a sua modulação). O PWM deve ser configurado no Timer B0.1. Para mapear o Timer B0.1 para o pino P4.7 pode ser utilizado o mapeamento:

```
P4SEL |= BIT7; //Disponibilizar P4.7
P4DIR |= BIT7; //P4.7 como saída
PMAPKEYID = 0X02D52; //Liberar mapeamento de P4
P4MAP7 = PM_TB0CCR1A; //P4.7 = TB0.1
```

Pontuação:

Configuração do ADC: 2 pontos
Configuração do PWM: 1 ponto
Funcionamento via Interrupção: 2 pontos.