Aluno: Mateus Alves da Rocha **Matrícula:** 11/0132661

Data: 12/05/2017

1. Defina a função void Atraso(volatile unsigned int x); que fornece um atraso de 'x' milissegundos. Utilize o Timer_A para a contagem de tempo, e assuma que o SMCLK já foi configurado para funcionar a 1 MHz. Esta função poderá ser utilizada diretamente nas outras questões desta prova.

```
void atraso ( volatile unsigned int x)
{
        TACCR0 = 1000-1;
        TACTL |= TACLR;
        TACTL = TASSEL_2 + ID_0 + MC_1;

        while (x--)
        {
                while((TACTL&TAIFG)==0);
                TACTL &= ~TAIFG;

        TACTL = MC_0;
}
```

2. Pisque os LEDs da Launchpad numa frequência de 100 Hz.

```
                int main (void)
{
                WDTCTL = WDTPW | WD1HOLD;

                P1OUT &= ~BIT0;
```

```
        P1DIR |= BIT0;
        TACCR0 = 5000-1;// 100hz
        TACTL = TASSEL_2 + ID_0 + MC_1 + TAIE;
        _BIS_SR(GIE + LMP0 _bits);
        return 0;

}
```

3. Pisque os LEDs da Launchpad numa frequência de 20 Hz.

```
int main (void)
{
    WDTCTL = WDTPW | WD1HOLD;

    P1OUT &= ~BIT0;
    P1DIR |= BIT0;
    TACCR0 = 25000-1;// 20hz - 25000-1
    TACTL = TASSEL_2 + ID_0 + MC_1 + TAIE;
    _BIS_SR(GIE + LMP0 _bits);
    return 0;

}
```

4. Pisque os LEDs da Launchpad numa frequência de 1 Hz.

```
int main (void)
{
        WDTCTL = WDTPW | WD1HOLD;

        P1OUT &= ~BIT0;
        P1DIR |= BIT0;
        TACCR0 =625000-1; // 1hz -- 62500-1
        TACTL = TASSEL_2 + ID_3 + MC_1 + TAIE; //id_3 para 1 hz
        _BIS_SR(GIE + LMP0 _bits);
        return 0;

}
```

5. Pisque os LEDs da Launchpad numa frequência de 0,5 Hz.

```
int main (void)
{
        WDTCTL = WDTPW | WD1HOLD;

        P1OUT &= ~BIT0;
        P1DIR |= BIT0;
        TACCR0 =625000-1; // 1hz -- 62500-1
        TACTL = TASSEL_2 + ID_3 + MC_3 + TAIE;
        _BIS_SR(GIE + LMP0 _bits);
```

```
    return 0;

}
```

6. Repita as questões 2 a 5 usando a interrupção do Timer A para acender ou apagar os LEDs.

7. Defina a função void paralelo_para_serial(void); que lê o byte de entrada via porta P1 e transmite os bits serialmente via pino P2.0. Comece com um bit em nivel alto, depois os bits na ordem P1.0 - P1.1 - … - P1.7 e termine com um bit em nível baixo. Considere um período de 1 ms entre os bits.

```
void atraso ( volatile unsigned int x)
{
    TACCR0 = 1000-1;
    TACTL |= TACLR;
    TACTL = TASSEL_2 + ID_0 + MC_1;

    while (x--)
    {
        while((TACTL&TAIFG)==0);
        TACTL &= ~TAIFG;

    TACTL = MC_0;
}
```

```
P1DIR = 0;
P2DIR |= BIT0;

void paralelo_para_serial(void)
{
    P2OUT |= BIT0;
    atraso (1);
    if (P1IN&BIT0) P2OUT |= BIT0
    else P2OUT &= ~BIT0;
    atraso (1);

    // com o laço

    char b, m;

    m = BIT0;
    for (b=0; b<8; b++)
    {
        if (P1IN&m) P2OUT |= BIT0;
        else P2OUT &= ~BIT0;
        atraso(1);
        m = (m<<1); // m *= 2; (multiplica por 2) desloca a
mascara um bit para a esquerda
    }
```

```
        P2OUT &= ~BIT0;
        atraso(1);
    }
```

8. Faça o programa completo que lê um byte de entrada serialmente via pino P2.0 e transmite este byte via porta P1. O sinal serial começa com um bit em nivel alto, depois os bits na ordem 0-7 e termina com um bit em nível baixo. Os pinos P1.0-P1.7 deverão corresponder aos bits 0-7, respectivamente. Considere um período de 1 ms entre os bits.

9. Defina a função void ConfigPWM(volatile unsigned int freqs, volatile unsigned char ciclo_de_trabalho); para configurar e ligar o Timer_A em modo de comparação

10. Considere que o pino P1.6 já foi anteriormente configurado como saída do canal 1 de comparação do Timer_A, que somente os valores {100, 200, 300, …, 1000} Hz são válidos para a frequência, que somente os valores {0, 25, 50, 75, 100} % são válidos para o ciclo de trabalho, e que o sinal de clock SMCLK do MSP430 está operando a 1 MHz.