11_Timer_A

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.**

```
#include <msp430g2553.h>

#define LED BIT0
#define LED1 BIT6

void atraso(volatile unsigned int x);

{
        WDTCTL = WDTPW + WDTHOLD;       // Stop WDT

        BCSCTL1 = CALBC1_1MHZ;          //MCLK e SMCLK @ 1MHz
        DCOCTL = CALDCO_1MHZ;           //MCLK e SMCLK @ 1MHz
        t = 8/1000000;
        P1OUT &= ~LED;
        P1DIR |= LED;
        TA0CCR0 = x/(1000 * t)-1;
        TA0CTL = TASSEL_2 + ID_3 + MC_1;
        while(1)
        {
                while((TA0CTL & TAIFG)==0);
                P1OUT ^= (LED + LED1);
                TA0CTL &= ~TAIFG;
        }
        return 0;
}
```

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

X = 1/100 = 0,01 s = 10 ms

TA0CCR0 = 10/(1000*t)-1 = 1250

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

   X = 1/20 = 0,05 s = 50 ms

   TA0CCR0 = 50/(1000*t)-1 = 62500- 1

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

```
#include <msp430g2553.h>

#define LED BIT0
#define LED1 BIT6
#define TIMER 2

int main(void)
{
	int time = 0;
	  WDTCTL = WDTPW + WDTHOLD;		// Stop WDT

	  BCSCTL1 = CALBC1_1MHZ;		//MCLK e SMCLK @ 1MHz
	  DCOCTL = CALDCO_1MHZ;		//MCLK e SMCLK @ 1MHz
	  P1OUT &= ~LED;
	  P1OUT |= LED1;
	  P1DIR |= (LED+LED1);
	  TA0CCR0 = 62500-1; //10000-1;
	  TA0CTL = TASSEL_2 + ID_3 + MC_1;
	  while(1)
	  {
		while((TA0CTL & TAIFG)==0);
	    TA0CTL &= ~TAIFG;
		if(time == (TIMER))
	    {
		time=0;
		P1OUT ^= (LED + LED1);
```

```
            }
        time++;
    }
    return 0;
}
```

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

```
#include <msp430g2553.h>

#define LED BIT0
#define LED1 BIT6
#define TIMER 4

int main(void)
{
    int time = 0;
      WDTCTL = WDTPW + WDTHOLD;        // Stop WDT

      BCSCTL1 = CALBC1_1MHZ;           //MCLK e SMCLK @ 1MHz
      DCOCTL = CALDCO_1MHZ;            //MCLK e SMCLK @ 1MHz
      P1OUT &= ~LED;
      P1OUT |= LED1;
      P1DIR |= (LED+LED1);
      TA0CCR0 = 62500-1; //10000-1;
      TA0CTL = TASSEL_2 + ID_3 + MC_1;
      while(1)
      {
            while((TA0CTL & TAIFG)==0);
        TA0CTL &= ~TAIFG;
            if(time == (TIMER))
        {
            time=0;
            P1OUT ^= (LED + LED1);

            }
        time++;
```

```
        }
        return 0;
}
```

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

FORMA ORIGINAL:

```
TA0CTL = TASSEL_2 + ID_3 + MC_1;
```

.

.

.

```
        while(1)
        {
              while((TA0CTL & TAIFG)==0);
            TA0CTL &= ~TAIFG;
              if(time == (TIMER))
            {
              time=0;
              P1OUT ^= (LED + LED1);

              }
            Time++;
```

.

.

.

FORMA COM INTERRUPÇÕES:

```
TA0CTL = TASSEL_2 + ID_3 + MC_1 + TAIE;

_BIS_SR(LPM0_bits+GIE);

                                                    .

                                                    .

                                                    .

interrupt(TIMER0_A1_VECTOR) TA0_ISR(void)
{

static int time=0;

        if(time == TIMER)

        {

                time=0;
                P1OUT ^= LED;
                TA0CTL &= ~TAIFG;

        }

        time++;
}                                       .

                                                    .

                                                    .
```

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.**

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. 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.**