Disciplina: Eletrônica Embarcada **Código:** 120871 **Turma:** A
Professor: Diogo Caetano Garcia
Aluno/Matrícula: Fábio Barbosa Pinto – 11/0116356

Questionário: 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.

```
void Atraso(volatile unsigned int x)
{
   TACCR0 = 1000-1; //capture mode ativo para pegar o timer em 999
   TACTL |= TACLR; // limpar o timer A
   TACTL = TASSEL_2 + ID_0 + MC_1; //TASSEL_2 escolhe SMCLK como o clock do TAR
                           //ID_0 significa que o clock sera dividido por 1
                           //MC_1 significa que o timer segue o modo UP e vai contar até o valor setado
em TACCR0
   while(x>0)
   {
     x--;
     while((TACTL&TAIFG)==0);
           TACTL &= ~TAIFG;
   }
   TACTL = MC_0; //timer no modo stop

}

int main(void) {
   WDTCTL = WDTPW | WDTHOLD;     // Stop watchdog timer
   BCSCTL1 = CALBC1_1MHZ;      //MCLK e SMCLK @ 1MHz
   DCOCTL = CALDCO_1MHZ;       //MCLK e SMCLK @ 1MHz
        return 0;
}
```

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

```
#include <msp430g2553.h>
#include <intrinsics.h>

#define LED1 BIT0
#define LED2 BIT6
```

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

   BCSCTL1 = CALBC1_1MHZ;      //MCLK e SMCLK @ 1MHz
   DCOCTL = CALDCO_1MHZ;       //MCLK e SMCLK @ 1MHz
   P1OUT &= ~(LED1+LED2);
   P1DIR |= LED1 + LED2;
   TA0CCR0 = 5000-1;
   TA0CTL = TASSEL_2 + ID_0 + MC_1 + TAIE;
   _BIS_SR(LPM0_bits+GIE);
   return 0;
}

#pragma vector = TIMER0_A1_VECTOR
__interrupt void TIMER0_TA0_ISR(void)
{
   P1OUT ^= LED1 + LED2;
   TA0CTL &= ~TAIFG;
}
```

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

```
#include <msp430g2553.h>
#include <intrinsics.h>

#define LED1 BIT0
#define LED2 BIT6

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

   BCSCTL1 = CALBC1_1MHZ;      //MCLK e SMCLK @ 1MHz
   DCOCTL = CALDCO_1MHZ;       //MCLK e SMCLK @ 1MHz
   P1OUT &= ~(LED1+LED2);
   P1DIR |= LED1 + LED2;
   TA0CCR0 = 25000-1;
   TA0CTL = TASSEL_2 + ID_0 + MC_1 + TAIE;
   _BIS_SR(LPM0_bits+GIE);
   return 0;
}
```

```
#pragma vector = TIMER0_A1_VECTOR
__interrupt void TIMER0_TA0_ISR(void)
{
   P1OUT ^= LED1 + LED2;
   TA0CTL &= ~TAIFG;
}
```

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

```
#include <msp430g2553.h>
#include <intrinsics.h>

#define LED1 BIT0
#define LED2 BIT6

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

   BCSCTL1 = CALBC1_1MHZ;      //MCLK e SMCLK @ 1MHz
   DCOCTL = CALDCO_1MHZ;       //MCLK e SMCLK @ 1MHz
   P1OUT &= ~(LED1+LED2);
   P1DIR |= LED1 + LED2;
   TA0CCR0 = 62500-1; //10000-1;
   TA0CTL = TASSEL_2 + ID_3 + MC_1 + TAIE;
   _BIS_SR(LPM0_bits+GIE);
   return 0;
}

#pragma vector = TIMER0_A1_VECTOR
__interrupt void TIMER0_TA0_ISR(void)
{
   P1OUT ^= LED1 + LED2;
   TA0CTL &= ~TAIFG;
}
```

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

```
#include <msp430g2553.h>
#include <intrinsics.h>

#define LED1 BIT0
```

```
#define LED2 BIT6

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

   BCSCTL1 = CALBC1_1MHZ;      //MCLK e SMCLK @ 1MHz
   DCOCTL = CALDCO_1MHZ;       //MCLK e SMCLK @ 1MHz
   P1OUT &= ~(LED1+LED2);
   P1DIR |= LED1 + LED2;
   TA0CCR0 = 62500-1; //10000-1;
   TA0CTL = TASSEL_2 + ID_3 + MC_3 + TAIE;
   _BIS_SR(LPM0_bits+GIE);
   return 0;
}

#pragma vector = TIMER0_A1_VECTOR
__interrupt void TIMER0_TA0_ISR(void)
{
   P1OUT ^= LED1 + LED2;
   TA0CTL &= ~TAIFG;
}
```

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

```
#include <msp430.h>
#define LED1 BIT0
#define LED2 BIT6
#define LEDS (BIT0 + BIT6)


int main (void){
   WDTCTL = WDTPW + WDTHOLD;


   BCSCTL1 = CALBC1_1MHZ;
   DCOCTL = CALDCO_1MHZ;
   P1OUT&=~LEDS;
   P1DIR|=LEDS;
   TA0CCR0=5000-1;
   TA0CTL = TASSEL_2 + ID_3+ MC_1
+TAIE;
   _BIS_SR(LPM0_bits+GIE);
   return 0;}
```

```
enable_interrupt();


#pragma vector = TIMER0_A1_VECTOR


__interrupt void TA0_ISR(void)
      {
   P1OUT ^=LEDS;
   TA0CTL &=~TAIFG;
      }
```

Questão 5:

```
#include <msp430.h>
#define LED1 BIT0
#define LED2 BIT6
#define LEDS (BIT0 + BIT6)



int main (void){
   WDTCTL = WDTPW + WDTHOLD;


   BCSCTL1 = CALBC1_1MHZ;
   DCOCTL = CALDCO_1MHZ;
   P1OUT&=~LEDS;
   P1DIR|=LEDS;
   TA0CCR0=62500-1;
   TA0CTL = TASSEL_2 + ID_3+ MC_3
+TAIE;
   _BIS_SR(LPM0_bits+GIE);
   return 0;}


enable_interrupt();


#pragma vector = TIMER0_A1_VECTOR


__interrupt void TA0_ISR(void)
```

```
    {
  P1OUT ^=LEDS;
  TA0CTL &=~TAIFG;
    }
```

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.

```
#include <msp430.h>

/*
 * main.c
 */

void Atraso(volatile unsigned int x)
{
   TACCR0 = 1000-1; //capture mode ativo para pegar o timer em 999
   TACTL |= TACLR; // limpar o timer A
   TACTL = TASSEL_2 + ID_0 + MC_1; //TASSEL_2 escolhe SMCLK como o clock do TAR
                           //ID_0 significa que o clock sera dividido por 1
                           //MC_1 significa que o timer segue o modo UP e vai contar até o valor setado
em TACCR0
   while(x>0)
   {
     x--;
     while((TACTL&TAIFG)==0);
           TACTL &= ~TAIFG;
   }
   TACTL = MC_0; //timer no modo stop

}

void paralelo_para_serial(void)
{

   int i,x;
   x = P1IN;
   P2OUT |= BIT0; //Começando com um bit 1 e mandando ele por 1 ms
   Atraso(1);
   for(i=0;i<8;i++)
   {
```

```
        P2OUT |= (x&BIT0); //Mascarando x para nao aparecer valor espúrio na saída de P2OUT
inteira
        Atraso(1);
        x = (x >> 1); //vai mandando do lsb para o msb de P1IN
    }
    P2OUT &= ~(BIT0); //Manda o bit 0 para terminar a comunicação
    Atraso(1);
}


int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
    BCSCTL1 = CALBC1_1MHZ;      //MCLK e SMCLK @ 1MHz
    DCOCTL = CALDCO_1MHZ;       //MCLK e SMCLK @ 1MHz
    P1DIR = 0; //Definindo toda a porta P1 como entrada
    P2OUT = 0; //Inicializando toda a porta P2 em 0
    P2DIR |= BIT0; //Definindo somente P2.0 definida como saida
    for(;;)
    {
        paralelo_para_serial();
    }
    return 0;
}
```

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.

```
#include <msp430g2553.h>

/*
 * main.c
 */

void Atraso(volatile unsigned int x)
{
    TACCR0 = 1000-1; //capture mode ativo para pegar o timer em 999
    TACTL |= TACLR; // limpar o timer A
    TACTL = TASSEL_2 + ID_0 + MC_1; //TASSEL_2 escolhe SMCLK como o clock do TAR
                                //ID_0 significa que o clock sera dividido por 1
                                //MC_1 significa que o timer segue o modo UP e vai contar até o valor setado
em TACCR0
    while(x>0)
```

```
    {
      x--;
      while((TACTL&TAIFG)==0);
            TACTL &= ~TAIFG;
    }
    TACTL = MC_0; //timer no modo stop

}

void serial_para_paralelo(void)
{
    /*Na entrada vem uma palavra da forma 1 XXXX XXXX 0
     *
     */
    if((P2IN&BIT0) == 1)
    {
        /*Deve ser verificado se o primeiro bit é 1.
         *É meio que a lógica do bit de paridade
         *
         */
        Atraso(1); /*Espera 1 ms pra aparecer a primeira entrada de dados*/
        int x=0,i,y=0;
            for(i=0;i<8;i++)
            {
                x = 0;
                x |= (P2IN&BIT0);//mascarando a porta P2IN para pegar somente o valor de P2.0 e
armazenar em x
                y |= (x << i); //desloca x
                Atraso(1); //Espera 1 ms até vir a próxima entrada
            }
        if((P2IN&BIT0)==0)
            {/*Aqui tem que verificar outro bit na porta P2.0*/
             P1OUT = y;
            }
    }
}


int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
    BCSCTL1 = CALBC1_1MHZ;      //MCLK e SMCLK @ 1MHz
    DCOCTL = CALDCO_1MHZ;       //MCLK e SMCLK @ 1MHz
    P1OUT = 0; //Limpando a porta P1
    P1DIR = 0xFF; //Definindo toda a porta P1 como saida
    P2DIR &= ~(BIT0); //Definindo P2.O como entrada
```

```
    for(;;)
    {
        serial_para_paralelo();
    }
    return 0;
}
```

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.