

# Experimento 5

## Cronômetro utilizando MSP430

Mattos, A. Guilherme  
Universidade de Brasília  
Faculdade Gama  
Brasília, Brasil  
guilherme.mattos.gam@gmail.com

Souza, Paulo Augusto M. F. de  
Universidade de Brasília  
Faculdade Gama  
Brasília, Brasil  
pauloaugustomiguelfonseca@gmail.com

**Abstract**— *The microcontroller also has a system clock , thus it has a working time. The Timer\_A is a timer that uses the clock MSP , and also the most complete. It has a register, TAR 16 bits, which allows the choice of the system clock . After the selected clock then is made that the division of the clock frequency , which can be 1,2,4,8 times and then immediately the counter mode. This is all done so that there is better use depending on the application .*

**Keywords**— *Multiplexação, clock, Timer\_A*

### I. OBJETIVOS

Utilizar o microcontrolador MSP430 para projetar um cronômetro com precisão de 0.1s, utilizando 4 displays de sete segmentos multiplexados piscando em uma frequência de 100 Hz.

### II. INTRODUÇÃO

Como há limite para o número de pinos no MSP, isso limita sempre a quantidade de entrada ou saídas. Na lounchpad usada, possui 14 pinos de I/O. Para muitas aplicações, podem ser usados sempre mais do que esses 14, para isso usa-se o recurso da multiplexação. Isso pode ser feito a partir da programação, fazendo com que diminua o número de pinos usados. No caso desse experimento por exemplo, como cada display tem oito pinos, e são quatro ao todo, seriam usados 32 pinos sem multiplexar, somente para ascender os displays. Com a multiplexação, foi feito com que cada display recebesse sempre o mesmo número, assim foi escolhido qual display ascenderia por vez, dando a impressão que cada um recebesse um número diferente. Isso pode ser feito, para que se reduza significativamente o hardware, e assim economizando pinos no MSP.

O microcontrolador tem também um sistema de clock, fazendo assim que ele tenha um tempo de funcionamento. O Timer\_A é um timer que usa o clock do MSP, e também o mais completo. Possui um registrador, TAR, de 16 bits, o que permite a escolha do sistema de clock. Após escolhido o

clock, então é feita a divisão da frequência desse clock, que pode ser de 1,2,4,8 vezes e logo em seguida o modo de contagem. Isso tudo é feito para que haja um melhor uso, dependendo da aplicação.

### III. DESENVOLVIMENTO

#### A. Descrição do Software

O código para realização deste experimento consiste em 3 funções, uma de atraso, uma para selecionar como ligar o display de acordo com o numeral que se queira, e a função principal, 2 interrupções pelo timer\_A, uma usada a cada 0,1s e outra a cada 400 Hz, para garantir as especificações de mostrar os quatro displays piscando a cada 100hz.

A função de atraso é apenas para utilizar no debounce do botão, caso não utilizasse o botão seria considerado pressionado várias vezes mesmo sendo pressionado apenas uma vez, pausando ou ligando a contagem, sendo assim o controle para pausar ou não a contagem não seria tão eficaz, por isso foi utilizada essa função.

Foram declaradas também algumas variáveis globais que serão utilizadas nas interrupções e na função DISPLAY. Essa função liga os displays de acordo com o número que estiver na contagem, como são números de 0 a 9 foi criado um switch case para cada caso, considerando que os displays sejam ligados pelos BIT0-BIT7, com exceção do BIT3 que é a entrada, e que correspondem aos terminais a-g dos displays. Ao final da função os bits que serão ligados são colocados na variável “display” e a função retorna esse valor, com os bits que serão ligados.

Na função principal são definidas as saídas na P1 como BIT0,1,2,4,5,6,7, e entrada o BIT3, definido no início do programa como “BTN”. Na porta P2 as saídas são BIT0,1,2,4 que serão utilizadas para multiplexação dos displays. Essas duas instruções “BCSCTL1=CALBC1\_1MHZ; DCOCTL = CALDCO\_1MHZ;” definem que o clock utilizado possui a frequência de 1MHz. Utilizando dois canais diferentes do timer\_A foram definidos cada um pra fazer uma contagem, o TA0 conta em modo up/down, ou seja ele conta até o valor definido, que no caso é 6250, e depois vai diminuindo até

A interrupção do TAO, como já foi dito, é chamada a cada 0.1 segundos e utilizando variável “DECSEG” a cada interrupção ela é incrementada, indicando que a cada décimo de segundo essa variável recebe mais um, e é exatamente o que o experimento consiste, um cronômetro contando a cada décimo de segundo, logo após utilizando ifs se essa variável chegar ao valor 10, ela é zerada e incrementada a variável “SEG\_UNI” que representa os segundos que serão colocados no segundo display, da direita pra esquerda, a mesma lógica serve para todas as outras variáveis, quando chegar a 10 a unidade dos segundos, ela é zerada e incrementada na dezena de segundos, chamada “SEG\_DEZ”. Ao contar até 999.9 o cronômetro vai ser zerado, se não aparecerão valores que não estão especificados no último display.

TA0CTL, para o cronômetro voltar a contar basta pressionar o botão novamente, caindo assim no if colocando o modo de contagem novamente para up/down “MC\_3” e assim sucessivamente. Ao final de todas interrupções é zerada a flag de interrupção, para sair da interrupção e para que ela possa ser habilitada novamente.

Como foram usados quatro displays, não seriam suficientes as portas do MSP, então foram feitas a multiplexação das saídas. Foram usados displays de cátodo comum, ou seja, liga-se o 0 no comum, e os leds dele são ativados em nível alto. Os comuns foram ligados nos pinos P2.0, P2.1, P2.2, P2.4, as cores azul, vermelho, preto e rosa respectivamente, do display um ao quatro. Então utilizando da lógica na programação, leva para nível baixo apenas uma porta e as outras três em nível alto, assim ascendia somente um display por vez.

Agora a saída para os leds, cada segmento do display, foi ligada no P1.0, P1.1, P1.2, P1.4, P1.5, P1.6, P1.7, estes foram ligados aos segmentos, A,B,C,D,E,F,G respectivamente. Como é possível ver no esquemático, cada um foi ligado com uma cor diferente, para uma melhor visualização. Como cada saída foi ligada igualmente em todos os displays, fez-se então a multiplexação dos comuns dos display. Isso fez com que cada display ascendesse por vez, mas como o clock era de 100Hz, a essa velocidade, ao olho humano já não era possível

ver cada um ascendendo por vez. E por fim, o pino P1.3 foi definido como o push-button que tem na placa, para iniciar, parar e retomar a contagem.

#### IV. RESULTADOS

No primeiro momento não foi colocado os resistores de pull-up no botão na interrupção do TA1, imaginando que bastasse colocar na função principal testamos o código e percebeu-se que ao entrar na interrupção o BIT3 não estava mais sendo considerado, estava sempre em zero, como se estivesse sempre pressionado, o que impossibilitou de criar a lógica de pausa da contagem, ao inserir os comandos pra definir ele como entrada, e habilitando o resistor de pull-up no BIT3 na própria interrupção do timer\_A, ele mantém nível alto, e ao ser pressionado vai para nível baixo, o que permitiu pausar ou continuar a contagem.

#### V. CONCLUSÃO

Como visto no experimento, dependendo da velocidade do clock usado, é possível ver ou não os displays piscando. Com o clock a 100Hz era como se eles estivessem acesos continuamente. Quando se fez um teste com 1Hz, era possível ver claramente, como somente um se ascendia por vez. Assim foi possível ver o resultado da multiplexação, todos os números eram mandados para os displays, mas somente um display ascendia por vez, isso dava a impressão que somente um recebia de cada vez os números. Com este experimento além de utilizar um conceito que ajuda muito com a limitação de pinos que se pode utilizar, que é a multiplexação, houve também um grande uso nas interrupções pelo timer\_A, e quão amplo pode ser sua funcionalidade, tanto na utilização da contagem para criar o cronômetro, como na frequência em que se queira ligar ou desligar algo. Essa ferramenta permite uma vasta aplicação e funcionalidade, caso queira receber algum dado após um tempo especificado o MSP se torna uma ferramenta perfeita para este uso, podendo deixar ele em vários modos de consumo onde ainda consiga utilizar algum dos clocks.

#### REFERÊNCIAS

- [1] John H. Davies, MSP430 Microcontroller Basics, Elsevier, 2008.
- [2] Software: fritzing (org).

## Anexos

```
#include <msp430g2553.h>
#include <legacymsp430.h>
#define BTN BIT3
#define BTN_DLY 10000 // VALOR PARA DEBOUNCE DO BOTÃO

// VARIÁVEIS GLOBAIS
volatile int DECSEG=0;
volatile int SEG_UNI=0;
volatile int SEG_DEZ=0;
volatile int SEG_CEN=0;
volatile int i=0;
volatile int mostra;

// FUNÇÃO DE ATRASO
void delay(volatile unsigned long int t)
{
    volatile unsigned long int i;
    for(i=0;i<t;i++) { }
}

// FUNÇÃO QUE SETA AS SAÍDAS DOS DISPLAYS DE ACORDO COM O NÚMERO CORRESPONDENTE
int DISPLAY(int SEGs){

int display;
switch(SEGs)
{
case 0 :
display = (BIT0 + BIT1 + BIT2 + BIT4 + BIT5 + BIT6);
break;
case 1 :
display = (BIT1 + BIT2);
break;
case 2:
display = (BIT0 +BIT1 + BIT7 + BIT5 +BIT4);
break;
case 3:
display = (BIT0 + BIT1 + BIT7 + BIT2 + BIT4);
break;
case 4:
display = (BIT6 + BIT7 + BIT1 + BIT2);
break;
case 5:
display = (BIT0 + BIT6 + BIT7 + BIT2 + BIT4 );
break;
case 6:
display = (BIT0 + BIT6 + BIT2 + BIT4 + BIT5 + BIT7);
break;
case 7:
display = (BIT0 + BIT1 + BIT2);
break;
case 8:
```

```

display = (BIT0 + BIT1 + BIT2 + BIT4 + BIT5 + BIT6 + BIT7);
break;
case 9:
display = (BIT0 + BIT1 + BIT2 + BIT6 + BIT7 + BIT4 );
break;
}
return display;

// FUNÇÃO PRINCIPAL
}

int main (void)
{
    WDTCTL = WDTPW + WDTHOLD; // PARA WATCHDOG TIMER
    P1SEL = 0; P1SEL2 = 0;
    P1REN |= BTN; // HABILITAR RESISTOR DE PULL-UP/DOWN NO BTN
    P1OUT |= BTN; // RESISTOR BTN NO VCC – PULL-UP

    P1DIR = BIT0+BIT1+BIT2+BIT4+BIT5+BIT6+BIT7; //SAÍDAS
    P2DIR = BIT0+BIT1+BIT2+BIT4; //SAÍDAS
    BCSCTL1=CALBC1_1MHZ; // DEFINE A FREQUÊNCIA COMO 1 MHz
    DCOCTL = CALDCO_1MHZ;
    TA0CCR0= 6250; // 6250*2*8uS= 0,1s
    TA0CTL = TASSEL_2 + MC_3 + ID_3; // SMCLK , UP/DOWN MODE, /8
    TA1CCR0 = 2500-1; // 2500*1uS= 0,0025S = 2,5 mS= 400Hz para cada display, como são 4 displays 100hz pra todos
    TA1CTL = TASSEL_2 + ID_0 + MC_1+TAIE ; // SMCLK , UP MODE, /1

    for(;;){
        _BIS_SR(GIE);
    }

    return 0;
}

// INTERRUPTO QUE CONTA A CADA 0,1 SEGUNDOS
interrupt(TIMERO_A1_VECTOR) TA0_ISR(void)
{

    if(DECSEG<10) DECSEG++; // INCREMENTA O DÉCIMO DE SEGUNDO A CADA CONTAGEM

    if(DECSEG==10) //CASO DÉCIMO DE SEG =10 ZERA E INCREMENTA UNIDADE
    {
        DECSEG=0;
        SEG_UNI++;

        if(SEG_UNI==10) //CASO UNIDADE =10 ZERA UNIDADE E INCREMENTA DEZENA
        {
            SEG_UNI=0;
            SEG_DEZ++;

            if(SEG_DEZ==10) //CASO DEZENA =10 ZERA DEZENA E INCREMENTA CENTENA
            {
                SEG_DEZ=0;
            }
        }
    }
}

```

```

    SEG_CEN++;
    }}}

TA0CTL &= ~TAIFG;
}

// INTERRUPTOÇÃO CHAMADA A CADA 2,5 ms=400Hz
//AO FINAL DE 4 CHAMADAS DESSA INTERRUPTOÇÃO,
//MOSTRAR OS 4 DISPLAYS A CADA 100HZ
interrupt(TIMER1_A1_VECTOR) TA1_ISR(void)
{
    P1REN |= BTN;
    P1OUT |= BTN;
    P1DIR &= ~BTN;
    // SE O BOTÃO FOR PRESSIONADO
    // PAUSA A CONTAGEM OU CONTINUA A CONTAR DEPENDENDO DO VALOR DE i
    if((P1IN&BTN)==0) { TA0CTL &= ~TAIFG;

    if((i%2)==0){ TA0CTL = TASSEL_2 + MC_3 + ID_3+ TAIE;} //i PAR HABILITA CONTAGEM
    else {TA0CTL = TASSEL_2 + MC_0 + ID_3+ TAIE; } // i ÍMPAR PAUSA CONTAGEM
    i++;
    delay(BTN_DLY);}

    if (mostra==0){
        P2OUT=0x1E; // 11110 -> 30 -> 1E SELECIONA O PRIMEIRO DISPLAY
        P1OUT= DISPLAY(DECSEG);
    }

    if(mostra==1){
        P2OUT=0x1D; //11101->29->1D SELECIONA O SEGUNDO DISPLAY
        P1OUT= DISPLAY(SEG_UNI);
    }

    if (mostra==2){
        P2OUT=0x1B; //11011->27->1B SELECIONA O TERCEIRO DISPLAY
        P1OUT= DISPLAY(SEG_DEZ);
    }

    if (mostra==3){
        P2OUT=0x0F; //01111->0F SELECIONA O QUARTO DISPLAY
        P1OUT= DISPLAY(SEG_CEN);
    }

    //LÓGICA PARA SELECIONAR CADA UM DOS DISPLAYS
    // A CADA INTERRUPTOÇÃO LIGA UM E APAGA OS OUTROS TRÊS

    mostra++;
    if (mostra==4){
        mostra = 0;
    }

    TA1CTL &= ~TAIFG;
}

```