UFSC - Universidade Federal de Santa Catarina
Centro Tecnológico de Joinville
Giulia Denise Kujat Vieira
EMB5642-04605 (20221) - Microcontroladores

Relatório Prova 1

Esse documento mostra a realização do código para uma calculadora que converte números decimais e hexadecimais, ou hexadecimais em decimais. A conversão acontecerá ao pressionar dois botões simultaneamente, sendo 3 e B para conversão de decimal para hexadecimal e B e F para conversão de hexadecimal para decimal.

Ao digitar mais do que quatro dígitos no display aparecerá a mensagem de erro.

Quando a conversão exigir cinco dígitos para a representação, aparecerá a mensagem de erro.

Ordem da apresentação do código:

- Definição de variáveis
- Configuração na main dos pinos da matriz.
- Matriz de botões
- Funções auxiliares para otimização
- Display de 7 segmentos
- Interrupção para conversão
- Funções de conversão

Definição de variáveis

```
// VARIAVEIS GLOBAIS
uint8_t Valores[18] = {sseg_0, sseg_1, sseg_2, sseg_3, sseg_4, sseg_5, sseg_6, sseg_7, sseg_8,
                       sseg_9, sseg_A, sseg_B, sseg_C, sseg_D, sseg_E,  sseg_F, sseg_V, sseg_W};
uint32_t portal_linha[4] = { GPIOPortF_base, GPIOPortB_base , GPIOPortB_base , GPIOPortB_base };
uint32_t portal_coluna[4] = { GPIOPortF_base, GPIOPortF_base , GPIOPortF_base , GPIOPortF_base };
uint32_t pino_linha[4] = { Pino_4, Pino_0, Pino_1, Pino_5 };
uint32_t pino_coluna[4] = { Pino_0, Pino_1, Pino_2, Pino_3 };
uint32_t numero[4][4] = { 0, 1, 2, 3,
                          4, 5, 6, 7,
                          8, 9,10,11,
                         12,13,14,15};

int i=0,j=0,k=0, num;
int counter = 0, aux_counter = 0, uni = 17, dez = 17, cen = 17, mil = 17,number_grande = 0,delaym;
```

- Para melhor otimização do programa, foram criados vetores com os valores dos segmentos, portais e números recebidos da matriz.
- Para representar os dígitos, foi definido variáveis para casa de milhar, centena, dezena e unidade (mil, cen, dez, uni).
- Para conversão foi definida a variável “number_grande” que seria o valor real inserido.
- Restante valores para iterações e auxílio da lógica aplicada.

```
#define GPIOPortA_base              0x40004000
#define GPIOPortB_base              0x40005000
#define GPIOPortC_base              0x40006000
#define GPIOPortD_base              0x40007000
#define GPIOPortE_base              0x40024000
#define GPIOPortF_base              0x40025000

//defines Funcoes
//HabilitaPortal
#define HabPortalA                  0x01
#define HabPortalB                  0x02
#define HabPortalC                  0x04
#define HabPortalD                  0x08
#define HabPortalE                  0x10
#define HabPortalF                  0x20

//ConfiguraPinoSaida
#define Pino_0                      0x01
#define Pino_1                      0x02
#define Pino_2                      0x04
#define Pino_3                      0x08
#define Pino_4                      0x10
#define Pino_5                      0x20
#define Pino_6                      0x40
#define Pino_7                      0x80

//Valores Display 7seg
#define sseg_0                      0b00111111
#define sseg_1                      0b00000110
#define sseg_2                      0b01011011
#define sseg_3                      0b01001111
#define sseg_4                      0b01100110
#define sseg_5                      0b01101101
#define sseg_6                      0b01111101
#define sseg_7                      0b00000111
#define sseg_8                      0b01111111
#define sseg_9                      0b01101111
#define sseg_A                      0b01110111
#define sseg_B                      0b01111100
#define sseg_C                      0b00111001
#define sseg_D                      0b01011110
#define sseg_E                      0b01111001
#define sseg_F                      0b01110001
#define sseg_V                      0b00111110
#define sseg_W                      0b00000000
```

Configuração na main dos pinos da matriz.

```
// Habilita F0, F1, F2, F3 COLUNAS COMO ENTRADA
ConfiguraPinoEntrada (GPIOPortF_base, Pino_4);
ConfiguraPinoPullUp(GPIOPortF_base, Pino_4);

ConfiguraPinoEntrada (GPIOPortB_base, Pino_0|Pino_1|Pino_5);
ConfiguraPinoPullUp(GPIOPortB_base, Pino_0|Pino_1|Pino_5);

DestravaPino(GPIOPortF_base, Pino_0);

// Habilita o pino 3 do portal F, configura como saída digital
ConfiguraPinoSaida (GPIOPortF_base, Pino_0|Pino_1|Pino_2|Pino_3);
ConfiguraPinoSaida (GPIOPortE_base, Pino_0|Pino_1|Pino_2|Pino_3);
ConfiguraPinoSaida (GPIOPortD_base, Pino_2|Pino_3);
ConfiguraPinoSaida (GPIOPortC_base, Pino_4|Pino_5|Pino_6|Pino_7);
ConfiguraPinoSaida (GPIOPortB_base, Pino_6|Pino_7);


GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_INT_PIN_0|GPIO_INT_PIN_4,GPIO_FALLING_EDGE);
IntEnable(INT_GPIOF);
GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_0|GPIO_INT_PIN_4);
IntMasterEnable();
```

- Configura (Linhas) F4, B0, B1, B5 como Entrada com Pull Up
- Destrava pino F0
- Configura pinos de coluna da matriz e pinos do display como saída.
- Configuração da interrupção para reconhecer na borda de descida.

Matriz de botões

```
279  void matriz_botao(void)
280  {
281
282      for(i = 0; i < 4; i++)
283      {
284          EscritaPinosPortal(portal_coluna[i], pino_coluna[0] | pino_coluna[1] | pino_coluna[2] | pino_coluna[3], ~(pino_coluna[i]));
285          for(j = 0; j < 4; j++)
286          {
287              delay(1000);
288              if(LeituraPinosPortal(portal_linha[j], pino_linha[j]) == 0x00 && counter != 4)
289              {
290
291                  aux_counter++;
292                  if (aux_counter > 15 && counter!=5){
293                      aux_counter = 0;
294
295                      mil = cen;
296                      cen = dez;
297                      dez = uni;
298                      uni = numero[j][i];
299
300                      counter++;
301                      if(counter==5){erro();}
302                  }
303              }
304
305          }
306      }
307  }
```

Escrita dos valores nos portais de coluna e leitura nos portais de linha para reconhecer qual botão está sendo pressionado. Para evitar que seja reconhecido o mesmo valor mais vezes fiz um **if** para reconhecer apenas a cada 15 somas no “aux_counter” (debouncer). Counter ter a quantidade de dígitos lidos e quando 5 mostra mensagem de erro.

Valor inserido é designado para o dígito da unidade, unidade para dezena, dezena para centena e centena para milhar. Obs: para correta conversão, é necessário inserir zero à esquerda.

Funções auxiliares para otimização

```
void HabilitaPortal (uint8_t HabPortalX)
{
    ESC_REG(SYSCTL_RCGCGPIO) |= HabPortalX;

}

void ConfiguraPinoSaida (uint32_t PortalBase, uint8_t Pino)
{
    ESC_REG(PortalBase + GPIO_OS_DIR) |= Pino;
    ESC_REG(PortalBase + GPIO_OS_DEN) |= Pino;
    ESC_REG(PortalBase + GPIO_OS_DR2R) |= Pino;
    ESC_REG(PortalBase + GPIO_OS_DR4R) &= ~(Pino);
    ESC_REG(PortalBase + GPIO_OS_DR8R) &= ~(Pino);
    ESC_REG(PortalBase + GPIO_OS_SLR) &= ~(Pino);

}

void ConfiguraPinoEntrada (uint32_t PortalBase, uint8_t Pino)
{
    ESC_REG(PortalBase + GPIO_OS_DIR) &= ~(Pino);
    ESC_REG(PortalBase + GPIO_OS_DEN) |= Pino;
    ESC_REG(PortalBase + GPIO_OS_DR2R) |= Pino;
    ESC_REG(PortalBase + GPIO_OS_DR4R) &= ~(Pino);
    ESC_REG(PortalBase + GPIO_OS_DR8R) &= ~(Pino);
    ESC_REG(PortalBase + GPIO_OS_SLR) &= ~(Pino);

}

int32_t LeituraPinosPortal (uint32_t PortalBase, uint8_t Pino)
{
    return(ESC_REG(PortalBase + (GPIO_OS_DATA + (Pino << 2))));
}

void EscritaPinosPortal (uint32_t PortalBase, uint8_t Pino, uint8_t Valor)
{
    ESC_REG(PortalBase + (GPIO_OS_DATA + (Pino << 2))) = Valor;
}
```

```
void ConfiguraPinoPullUp(uint32_t portal_base, uint8_t pino)
{
    ESC_REG(portal_base + GPIO_OS_PULLUP)  |= pino;
    //acessa o GPIO de Pull Up e ativa o resistor pull up para o pino passado como parametros
}

void DestravaPino(uint32_t portal_base, uint8_t pino)
{
    ESC_REG(portal_base + GPIO_O_GPIOLOCK) |= 0x4C4F434B;
    ESC_REG(portal_base + GPIO_O_GPIOCR ) |= pino;
}
```

Display de 7 segmentos

```
void desligaDisplay(){
    EscritaPinosPortal(GPIOPortB_base, Pino_6, 0x40);
    EscritaPinosPortal(GPIOPortB_base, Pino_7, 0x80);
    EscritaPinosPortal(GPIOPortD_base, Pino_2, 0x04);
    EscritaPinosPortal(GPIOPortD_base, Pino_3, 0x08);
}

void desligaDig(int n){
    switch (n){

        case 1:
            EscritaPinosPortal(GPIOPortB_base, Pino_6, 0x40);
            break;
        case 2:
            EscritaPinosPortal(GPIOPortB_base, Pino_7, 0x80);
            break;
        case 3:
            EscritaPinosPortal(GPIOPortD_base, Pino_2, 0x04);
            break;
        case 4:
            EscritaPinosPortal(GPIOPortD_base, Pino_3, 0x08);
            break;

    }
}

void ligaDig(int n){
    switch (n){

        case 1:
            EscritaPinosPortal(GPIOPortB_base, Pino_6, 0x00);
            break;
        case 2:
            EscritaPinosPortal(GPIOPortB_base, Pino_7, 0x00);
            break;
        case 3:
            EscritaPinosPortal(GPIOPortD_base, Pino_2, 0x00);
            break;
        case 4:
            EscritaPinosPortal(GPIOPortD_base, Pino_3, 0x00);
            break;

    }
}

void Display(int dig, uint8_t Valor){
    desligaDisplay();
    ligaDig(dig);
    num = Valores[Valor];
    EscritaPinosPortal(GPIOPortE_base, Pino_0|Pino_1|Pino_2|Pino_3, num);
    EscritaPinosPortal(GPIOPortC_base, Pino_4|Pino_5|Pino_6|Pino_7, num);
}
```

Função Display é a única necessária para imprimir o valor, recebendo o dígito (sendo 1 para milhar, 2 para centena, 3 para dezena e 4 para unidade) e o valor em número (0, 1, 2…).

```
while(1)
{
        //SysCtlDelay(10000);
        matriz_botao();
        delaym = 15000;

        for(ui32Loop = 0; ui32Loop < delaym; ui32Loop++) {  }
        Display(4,uni);

        for(ui32Loop = 0; ui32Loop < delaym; ui32Loop++) {  }
        Display(3,dez);

        for(ui32Loop = 0; ui32Loop < delaym; ui32Loop++) {  }
        Display(2,cen);

        for(ui32Loop = 0; ui32Loop < delaym; ui32Loop++) {  }
        Display(1,mil);


}
```

A impressão dos valores é realizada na mais, com delay definido de forma experimental. Nota-se que é necessário apenas alterar os valores mil, cen, dez e uni.

Portanto a definição das funções de erro e over fica da seguinte forma..

```
void erro(){
    mil = 14;
    cen = 10;
    dez = 10;
    uni = 0;
}

void over(){
    mil = 0;
    cen = 16;
    dez = 14;
    uni = 10;
}
```

Interrupção para conversão

```
void IntPortalF (void){
    uint32_t ui32Period;

    GPIOIntClear(GPIO_PORTB_BASE, Pino_0|Pino_1|Pino_5);

    if(LeituraPinosPortal(GPIOPortB_base, Pino_1) == 0x00 &&
       LeituraPinosPortal(GPIOPortB_base, Pino_5) == 0x00){
        conversao_hexa_decimal();
    }

    if(LeituraPinosPortal(GPIOPortF_base, Pino_4) == 0x00 &&
       LeituraPinosPortal(GPIOPortB_base, Pino_5) == 0x00){
        conversao_decimal_hexa();
    }


    GPIOIntDisable(GPIO_PORTF_BASE, GPIO_INT_PIN_0|GPIO_INT_PIN_4);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_A_ONE_SHOT);
    ui32Period = (SysCtlClockGet() / 1) / 2;
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
    IntEnable(INT_TIMER0A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    TimerEnable(TIMER0_BASE, TIMER_A);

}
```

Reconhecimento dos botões por interrupção, se pressionado B e F converterá de hexa para decimal, se pressionado 3 e F converterá de decimal e hexa.

Funções para conversão.

```
int conversao_hexa_decimal(void) // hexa --> decimal numero max 270F ~ 9999 - B1 e B5
{
    if(mil > 2 || (mil==2 && cen>7) || (mil==2 && cen==7 && dez>0))
    {
        over();
        return 0;
    }

    number_grande = mil*(16*16*16) + cen*(16*16) + dez*(16) + uni;

    mil=number_grande/1000;
    number_grande = number_grande%1000;

    cen=number_grande/100;
    number_grande = number_grande%100;

    dez=number_grande/10;
    number_grande = number_grande%10;

    uni=number_grande;
    return 1;
}

int conversao_decimal_hexa(void) // decimal --> hexa - F4 e B5
{
    if(mil>9 || cen>9 || dez >9 || uni > 9 )
    {
        mil = 0;
        cen = 16;
        dez = 14;
        uni = 10;
        return 0;
    }
    number_grande = 0;
    number_grande = (mil * (1000) + cen * (100) + dez * 10 + uni);
    uni = number_grande % 16;
    dez = (number_grande%(16*16)-uni)/16;
    cen = (number_grande %(16*16*16)-uni-dez)/(16*16);
    mil = (number_grande-uni-cen-dez)/(16*16*16);

    return 1;

}
```