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

Data: 05/05/2017

1. Projete o hardware necessário para o MSP430 controlar um motor DC de 12V e 4A. Utilize transistores bipolares de junção (TBJ) com Vbe = 0,7 V, beta = 100 e Vce(saturação) = 0,2 V. Além disso, considere que Vcc = 3 V para o MSP430, e que este não pode fornecer mais do que 10 mA por porta digital.

   Utilizando apenas o transistor:

$$I_c = 4V$$

$$I_b = \frac{I_c}{\beta} = \frac{4}{100} = 0.04\ A = 40mA$$

   Logo, o msp430 não conseguiria fornecer a corrente necessária. Nesse caso, utilizando o par Darlington:

$$I_b = \frac{I_c}{\beta^2} = \frac{4}{10000} = 0{,}4\ mA$$

$$R_b = \frac{Vcc - 2 * Vbe}{I_b} = \frac{3 - 2 * 0.7}{0.0004} = 4k\Omega$$

2. Projete o hardware necessário para o MSP430 controlar um motor DC de 10V e 1A. Utilize transistores bipolares de junção (TBJ) com Vbe = 0,7 V e beta = 120. Além disso, considere que Vcc = 3,5 V para o MSP430, e que este não pode fornecer mais do que 10 mA por porta digital.

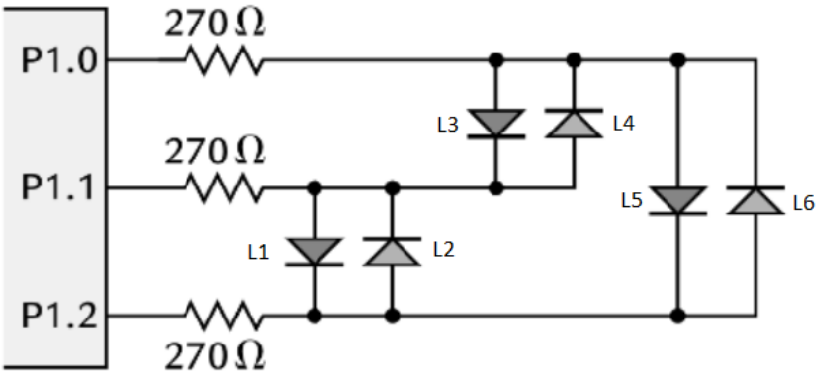
   Utilizando apenas o transistor:

$$I_c = 1\ A$$

$$I_b = \frac{I_c}{\beta} = \frac{1}{120} = 8.3\ mA$$

$$R_b = \frac{Vcc - Vbe}{I_b} = \frac{3.5 - 0.7}{8.33x10^{-3}} = 336\ \Omega$$

3. Projete o hardware utilizado para controlar 6 LEDs utilizando charlieplexing. Apresente os pinos utilizados no MSP430 e os LEDs, nomeados L1-L6.



Com N pinos, o MSP430 é capaz de controlar N(N-1) LEDS utilizando charlieplexing. Portanto, serão necessários apenas 3 pinos: P1.0, P1.1 e P1.2.

Para acionar L1: P1.1 = 1; P1.2 = 0; P1.0 = configurado como entrada digital

Para acionar L2: P1.1 = 0; P1.2 = 1; P1.0 = configurado como entrada digital

Para acionar L3: P1.0 = 1; P1.1 = 0; P1.2 = configurado como entrada digital

Para acionar L4: P1.0 = 0; P1.1 = 1; P1.2 = configurado como entrada digital

Para acionar L5: P1.0 = 1; P1.2 = 0; P1.1 = configurado como entrada digital

Para acionar L6: P1.0 = 0; P1.2 = 1; P1.1 = configurado como entrada digital

4. Defina a função void main(void){} para controlar 6 LEDs de uma árvore de natal usando o hardware da questão anterior. Acenda os LEDs de forma que um ser humano veja todos acesos ao mesmo tempo.

```
//programa para piscar os LEDS no Charlieplexing

#include <msp430g2553.h>

#define chpx1 BIT0
#define chpx2 BIT1
#define chpx3 BIT2
```

```
#define chpxs (chpx1+chpx2+chpx3)
//#define chpx_clr P1OUT &=~chpxs; // para limpar o p1out antes de mudar o p1dir

void charlie_on (char chppx_out, char chpx_on)
{
	P1OUT &=~chpxs;
	P1DIR &=~chpxs;
	P1DIR |= chpx_out;
	P1OUT |= chpx_on;
}

int main(void) // Main para utilizar quando se tem a funcao do charlie
{
	WDTCTL = WDTPW | WD1HOLD;
	while (1)
	{
		charlie_on (chpx1+chpx2, chpx1); //aciona d1
		charlie_on (chpx1+chpx2, chpx2); //aciona d2
		charlie_on (chpx2+chpx3, chpx2); //aciona d3
		charlie_on (chpx2+chpx3, chpx3); //aciona d4
		charlie_on (chpx1+chpx3, chpx1); //aciona d5
		charlie_on (chpx1+chpx3, chpx3); //aciona d6
	}
	return 0;
}



int main(void){

	WDTCTL = WDTPW | WDTHOLD;

	while(1)  //laço infinito
	{
```

```
            P1DIR = chpx1 + chpx2; //saida saida entrada
            P1OUT = chpx1; //aciona d1
            P1OUT = chpx2; //aciona d2

            P1DIR = chpx2 + chpx3; //entrada saida saida
            P1OUT = chpx2; //aciona d3
            P1OUT = chpx3; //aciona d4

            P1DIR = chpx1 + chpx3; //saida 1 e 3
            P1OUT = chpx1; //aciona d5
            P1OUT = chpx3; //aciona d6
        }
        return 1; //pois o compilador exige devido a funcao main ser int
    }
```

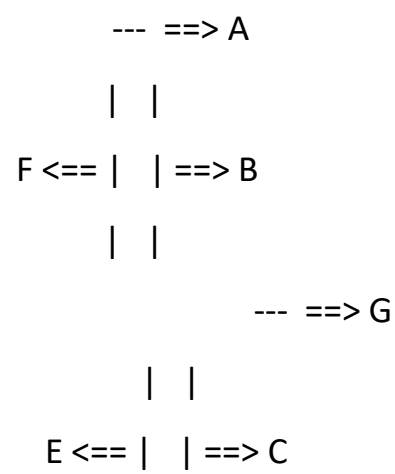
5. Defina a função void main(void){} para controlar 6 LEDs de uma árvore de natal usando o hardware da questão 3. Acenda os LEDs de forma que um ser humano veja os LEDs L1 e L2 acesos juntos por um tempo, depois os LEDs L3 e L4 juntos, e depois os LEDs L5 e L6 juntos.

```
//programa para piscar os LEDS no Charlieplexing dois de cada vez


#include <msp430g2553.h>


#define chpx1 BIT0

#define chpx2 BIT1

#define chpx3 BIT2

#define chpxs (chpx1+chpx2+chpx3)

//#define chpx_clr P1OUT &=~chpxs; // para limpar o p1out antes de mudar o p1dir
```

```
void charlie_on (char chppx_out, char chpx_on)
{
    P1OUT &=~chpxs;
    P1DIR &=~chpxs;
    P1DIR |= chpx_out;
    P1OUT |= chpx_on;
}

int main(void) // Main para utilizar quando se tem a funcao do charlie
{
    unsigned int i;

    WDTCTL = WDTPW | WD1HOLD;
    while(1)
    {
        for (i=0; i<0xFFFF; i++)
        {
            charlie_on(chpx1+chpx2, chpx1);
            charlie_on(chpx1+chpx2, chpx2);                    //fazer isso
em função
        }

        for (i=0; i<0xFFFF; i++)
        {
            charlie_on(chpx2+chpx3, chpx2);
```

```
            charlie_on(chpx2+chpx3, chpx3);
        }

        for (i=0; i<0xFFFF; i++)
        {
            charlie_on(chpx1+chpx3, chpx1);
            charlie_on(chpx1+chpx3, chpx3);
        }
    }

        return 0;
}
```

6. Defina a função void EscreveDigito(volatile char dig); que escreve um dos dígitos 0x0-0xF em um único display de 7 segmentos via porta P1, baseado na figura abaixo. Considere que em outra parte do código os pinos P1.0-P1.6 já foram configurados para corresponderem aos LEDs A-G, e que estes LEDs possuem resistores externos para limitar a corrente.



```
                    |   |
                            ---  ==> D


#include <msp430.h>

#define LEDA BIT0
#define LEDB BIT1
#define LEDC BIT2
#define LEDD BIT3
#define LEDE BIT4
#define LEDF BIT5
#define LEDG BIT6


void EscreveDigito(volatile char dig)
{

   if(dig=='0')
   {
   //Digito 0
   //LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 e LEDG = 0
   P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF;
   }
   else if(dig=='1')
```

```
{
//Digito 1
//LEDA = 0, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 0, LEDF = 0 E LEDG = 0
P1OUT |= LEDB + LEDC;
}
else if(dig=='2')
{
//Digito 2
//LEDA = 1, LEDB = 1, LEDC = 0, LEDD = 1, LEDE = 1, LEDF = 0 E LEDG = 1
P1OUT |= LEDA + LEDB + LEDE + LEDG;
}
else if(dig=='3')
{
//Digito 3
//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 0, LEDF = 0 E LEDG = 1
P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDG;
}
else if(dig=='4')
{
//Digito 4
//LEDA = 0, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 0, LEDF = 1 E LEDG = 1
P1OUT |= LEDB + LEDC + LEDG;
}
else if(dig=='5')
```

```
{//LEDA = 0, LEDB = 0, LEDC = 0, LEDD = 0, LEDE = 0, LEDF = 0 E LEDG = 0
 //Digito 5
 //LEDA = 1, LEDB = 0, LEDC = 1, LEDD = 1, LEDE = 0, LEDF = 1 E LEDG = 1
 P1OUT |= LEDA + LEDC + LEDD + LEDF + LEDG;
}
else if(dig=='6')
{//LEDA = 1, LEDB = 0, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 E LEDG = 1
 //Digito 6
 P1OUT |= LEDA + LEDC + LEDD + LEDE + LEDF + LEDG;
}
else if(dig=='7')
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 0, LEDF = 0 E LEDG = 0
 P1OUT |= LEDA + LEDC + LEDC;
}
else if(dig=='8')
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 E LEDG = 1
 P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF + LEDG;
}
else if(dig=='9')
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 0, LEDF = 1 E LEDG = 1
 P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDF + LEDG;
}
else if(dig=='A')
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 1, LEDF = 1 E LEDG = 1
```

```
    P1OUT |= LEDA + LEDB + LEDC + LEDE + LEDF + LEDG;
   }
   else if(dig=='B')
   {//LEDA = 0, LEDB = 0, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 E LEDG = 1
    P1OUT = LEDC + LEDD + LEDE + LEDF + LEDG;
   }
   else if(dig=='C')
   {//LEDA = 1, LEDB = 0, LEDC = 0, LEDD = 1, LEDE = 1, LEDF = 1 e LEDG = 0
    P1OUT = LEDA + LEDD + LEDE + LEDF;
   }
   else if(dig=='D')
   {//LEDA = 0, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 0 e LEDG = 1
    P1OUT = LEDB + LEDC + LEDD + LEDE + LEDG;
   }
   else if(dig=='E')
   {//LEDA = 1, LEDB = 0, LEDC = 0, LEDD = 1, LEDE = 1, LEDF = 1 e LEDG = 1
    P1OUT = LEDA + LEDD + LEDE + LEDF + LEDG;
   }
   else if(dig=='F')
   {//LEDA = 1, LEDB = 0, LEDC = 0, LEDD = 0, LEDE = 1, LEDF = 1 e LEDG = 1
    P1OUT = LEDA + LEDE + LEDF + LEDG;
   }
}
int main(void) {
```

```
    WDTCTL = WDTPW | WDTHOLD;     // Stop watchdog timer
    P1DIR |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF + LEDG;
    P1OUT = 0;
        while(1)
        {
           EscreveDigito('F');
        }
        return 0;
}
```

7. Multiplexe 2 displays de 7 segmentos para apresentar a seguinte sequência em loop:

   00 - 11 - 22 - 33 - 44 - 55 - 66 - 77 - 88 - 99 - AA - BB - CC - DD - EE - FF

```
#include <msp430.h>

#define LEDA BIT0
#define LEDB BIT1
#define LEDC BIT2
#define LEDD BIT3
#define LEDE BIT4
#define LEDF BIT5
#define LEDG BIT6
#define CT1 BIT7 // CT1 - Catodo Comum do Display 1
#define CT2 BIT0 // CT2 - Catodo Comum do Display 2 - Vai no bit 0 da P2OUT
```

```
void EscreveDigito(volatile char dig)
{
   if(dig=='0')
   {
   //Digito 0
   //LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 e LEDG = 0
   P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF;
   }
   else if(dig=='1')
   {
   //Digito 1
   //LEDA = 0, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 0, LEDF = 0 E LEDG = 0
   P1OUT |= LEDB + LEDC;
   }
   else if(dig=='2')
   {
   //Digito 2
   //LEDA = 1, LEDB = 1, LEDC = 0, LEDD = 1, LEDE = 1, LEDF = 0 E LEDG = 1
   P1OUT |= LEDA + LEDB + LEDE + LEDG;
   }
   else if(dig=='3')
   {
   //Digito 3
   //LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 0, LEDF = 0 E LEDG = 1
```

```
P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDG;
}
else if(dig=='4')
{
//Digito 4
//LEDA = 0, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 0, LEDF = 1 E LEDG = 1
P1OUT |= LEDB + LEDC + LEDG;
}
else if(dig=='5')
{//LEDA = 0, LEDB = 0, LEDC = 0, LEDD = 0, LEDE = 0, LEDF = 0 E LEDG = 0
 //Digito 5
 //LEDA = 1, LEDB = 0, LEDC = 1, LEDD = 1, LEDE = 0, LEDF = 1 E LEDG = 1
 P1OUT |= LEDA + LEDC + LEDD + LEDF + LEDG;
}
else if(dig=='6')
{//LEDA = 1, LEDB = 0, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 E LEDG = 1
 //Digito 6
 P1OUT |= LEDA + LEDC + LEDD + LEDE + LEDF + LEDG;
}
else if(dig=='7')
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 0, LEDF = 0 E LEDG = 0
 P1OUT |= LEDA + LEDC + LEDC;
}
else if(dig=='8')
```

```
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 E LEDG = 1
 P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF + LEDG;
}
else if(dig=='9')
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 0, LEDF = 1 E LEDG = 1
 P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDF + LEDG;
}
else if(dig=='A')
{//LEDA = 1, LEDB = 1, LEDC = 1, LEDD = 0, LEDE = 1, LEDF = 1 E LEDG = 1
 P1OUT |= LEDA + LEDB + LEDC + LEDE + LEDF + LEDG;
}
else if(dig=='B')
{//LEDA = 0, LEDB = 0, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 1 E LEDG = 1
 P1OUT = LEDC + LEDD + LEDE + LEDF + LEDG;
}
else if(dig=='C')
{//LEDA = 1, LEDB = 0, LEDC = 0, LEDD = 1, LEDE = 1, LEDF = 1 e LEDG = 0
 P1OUT = LEDA + LEDD + LEDE + LEDF;
}
else if(dig=='D')
{//LEDA = 0, LEDB = 1, LEDC = 1, LEDD = 1, LEDE = 1, LEDF = 0 e LEDG = 1
 P1OUT = LEDB + LEDC + LEDD + LEDE + LEDG;
}
else if(dig=='E')
```

```
    {//LEDA = 1, LEDB = 0, LEDC = 0, LEDD = 1, LEDE = 1, LEDF = 1 e LEDG = 1
     P1OUT = LEDA + LEDD + LEDE + LEDF + LEDG;
    }
    else if(dig=='F')
    {//LEDA = 1, LEDB = 0, LEDC = 0, LEDD = 0, LEDE = 1, LEDF = 1 e LEDG = 1
     P1OUT = LEDA + LEDE + LEDF + LEDG;
    }
}
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;     // Stop watchdog timer
    volatile int i;
    P1DIR |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF + LEDG + CT1;
    P2DIR |= CT2;
    P1OUT = 0;
    P2OUT = 0;
          while(1)
          {
                for (i=0x0; i=0xF; i++)
                {
                      switch (i)
                      {
                            case('0'):
                                  EscreveDigito('0');
                                  P1OUT ^= CT1;
```

```
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('1'):
            EscreveDigito('1');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('2'):
            EscreveDigito('2');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('3'):
            EscreveDigito('3');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
```

```
            break;
        case('4'):
            EscreveDigito('4');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('5'):
            EscreveDigito('5');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('6'):
            EscreveDigito('6');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('7'):
            EscreveDigito('7');
```

```
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('8'):
            EscreveDigito('8');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('9'):
            EscreveDigito('9');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('A'):
            EscreveDigito('A');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
```

```
            P2OUT ^= CT2;
            break;
        case('B'):
            EscreveDigito('B');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('C'):
            EscreveDigito('C');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('D'):
            EscreveDigito('D');
            P1OUT ^= CT1;
            P1OUT ^= CT1;
            P2OUT ^= CT2;
            P2OUT ^= CT2;
            break;
        case('E'):
```

```
                    EscreveDigito('E');
                    P1OUT ^= CT1;
                    P1OUT ^= CT1;
                    P2OUT ^= CT2;
                    P2OUT ^= CT2;
                    break;
                case('F'):
                        EscreveDigito('F');
                        P1OUT ^= CT1;
                        P1OUT ^= CT1;
                        P2OUT ^= CT2;
                        P2OUT ^= CT2;
                        break;
            }
        }
        return 0;

}
```