Josiane de Sousa Alves

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.

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.

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

//Código para acionar 6 leds usando apenas 3 pinos, os leds vão acender 1 por vez mas eu verei todos acesos, então não faz diferença a ordem que eles acendem.
//Mas se eu quiser que acenda numa ordem específica, deve-se zerar o P1OUT antes de cada passagem.

```
#include <msp430g2553.h>
#define CHPX1 BIT0
#define CHPX2 BIT1
#define CHPX3 BIT2

int main (void)
{
 WDTCTL = WDTPW|WDTHOLD;
 while (1)
 {
   P1DIR = CHPX1 + CHPX2;
   P1OUT = CPX1;
   P1OUT = CHPX2;
   P1DIR = CHPX2 + CHPX3;
   P1OUT = CHPX2;
   P1OUT = CHPX3;
   P1DIR = CHPX1 + CHPX3;
   P1OUT = CHPX1;
   P1OUT = CHPX3;
 }
return 1;
}
```

//Código para ligar os leds na ordem

```
#include <msp430g2553.h>
#define CHPX1 BIT0
#define CHPX2 BIT1
#define CHPX3 BIT2
#define CHPXS (CHPX1 + CHPX2 + CHPX3)
```

```
void charlie_on (char CHPX_OUT, char CHPX_ON)
{
  P1OUT &=~CHPXS;
  P1DIR &=~CHPXS;
  P1DIR |=CHPX_OUT;
  P1OUT |=CHPX_ON;
}

int main (void)

 WDTCTL = WDTPW|WDTHOLD;
 while (1)
 {

   charlie_on(CHPX1 + CHPX2, CHPX1);
   charlie_on(CHPX1 + CHPX2, CHPX2);
   charlie_on(CHPX2 + CHPX3, CHPX2);
   charlie_on(CHPX2 + CHPX3, CHPX3);
   charlie_on(CHPX1 + CHPX3, CHPX1);
   charlie_on(CHPX1 + CHPX3, CHPX3);
  }

return 0;

}

// Para não chamar a função 6x, podemos fazer um laço for, assim:

#include <msp430g2553.h>
#define CHPX1 BIT0
#define CHPX2 BIT1
#define CHPX3 BIT2
#define CHPXS (CHPX1 + CHPX2 + CHPX3)

void charlie_on (char CHPX_OUT, char CHPX_ON)
{
  P1OUT &=~CHPXS;
  P1DIR &=~CHPXS;
  P1DIR |=CHPX_OUT;
  P1OUT |=CHPX_ON;
}

int main (void)
  {
   char outs [] = {CHPX1 + CHPX2, CHPX1 + CHPX2, CHPX2 + CHPX3, CHPX2 + CHPX3, CHPX1 + CHPX3, CHPX1
+ CHPX3};
   char ons  [] = {CHPX1, CHPX2, CHPX2, CHPX3, CHPX1, CHPX3};
```

```
  char e;
 WDTCTL = WDTPW|WDTHOLD;
 while (1)
 {
  for (e = 0; e<6; e++)
   charlie_on (outs[e], ons[e]);
 }
 return 0;
}
```

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.

```
int main (void)
{ unsigned int i;
  WDTCTL = WDTPW|WDTHOLD;
 while (1)
 {
for (i=0, i<0xFFFF; i++)
 {
   charlie_on (CHPX1 + CHPX2, CHPX1);
   charlie_on (CHPX1 + CHPX2, CHPX2);
  }
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;
}
```

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.

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.

```
#include <msp430.h>

/*
 * main.c
 */

#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) {
  switch (dig)
  {
  case('0'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF;
  break;

  case('1'):
        P1OUT |= LEDB + LEDC;
  break;

  case('2'):
        P1OUT |= LEDA + LEDB + LEDD + LEDE + LEDG;
  break;

  case('3'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDG;
  break;

  case('4'):
        P1OUT |= LEDB + LEDC + LEDF + LEDG;
  break;

  case('5'):
        P1OUT |= LEDA + LEDC + LEDD + LEDF + LEDG;
  break;

  case('6'):
        P1OUT |= LEDA + LEDC + LEDD + LEDE + LEDF + LEDG;
  break;

  case('7'):
        P1OUT |= LEDA + LEDB + LEDC;
  break;
```

```
  case('8'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF + LEDG;
  break;

  case('9'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDF + LEDG;
  break;

  case('A'):
        P1OUT |= LEDA + LEDB + LEDC + LEDE + LEDF + LEDG;
  break;

  case('B'):
        P1OUT |= LEDC + LEDD + LEDE + LEDF + LEDG;
  break;

  case('C'):
        P1OUT |= LEDA + LEDD + LEDE + LEDF;
  break;

  case('D'):
        P1OUT |= LEDB + LEDC + LEDD + LEDE + LEDG;
  break;

  case('E'):
        P1OUT |= LEDA + LEDD + LEDE + LEDF + LEDG;
  break;

  case('F'):
        P1OUT |= LEDA + LEDE + LEDF + LEDG;
  break;


  }
}

int main(void) {
   WDTCTL = WDTPW | WDTHOLD;         // Stop watchdog timer
   P1DIR = 0xFF;
   for (;;){
      EscreveDigito ('7');
   }

          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>

/*
 * main.c
 */

#define LEDA BIT0
#define LEDB BIT1
#define LEDC BIT2
#define LEDD BIT3
#define LEDE BIT4
#define LEDF BIT5
#define LEDG BIT6
#define CATODO1 BIT7
#define CATODO2 BIT0

void EscreveDigito (volatile char dig) {
  switch (dig)
  {
  case('0'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF;
  break;

  case('1'):
        P1OUT |= LEDB + LEDC;
  break;

  case('2'):
        P1OUT |= LEDA + LEDB + LEDD + LEDE + LEDG;
  break;

  case('3'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDG;
  break;

  case('4'):
        P1OUT |= LEDB + LEDC + LEDF + LEDG;
  break;

  case('5'):
        P1OUT |= LEDA + LEDC + LEDD + LEDF + LEDG;
  break;
```

```
  case('6'):
        P1OUT |= LEDA + LEDC + LEDD + LEDE + LEDF + LEDG;
  break;

  case('7'):
        P1OUT |= LEDA + LEDB + LEDC;
  break;

  case('8'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF + LEDG;
  break;

  case('9'):
        P1OUT |= LEDA + LEDB + LEDC + LEDD + LEDF + LEDG;
  break;

  case('A'):
        P1OUT |= LEDA + LEDB + LEDC + LEDE + LEDF + LEDG;
  break;

  case('B'):
        P1OUT |= LEDC + LEDD + LEDE + LEDF + LEDG;
  break;

  case('C'):
        P1OUT |= LEDA + LEDD + LEDE + LEDF;
  break;

  case('D'):
        P1OUT |= LEDB + LEDC + LEDD + LEDE + LEDG;
  break;

  case('E'):
        P1OUT |= LEDA + LEDD + LEDE + LEDF + LEDG;
  break;

  case('F'):
        P1OUT |= LEDA + LEDE + LEDF + LEDG;
  break;


  }
}

int main(void) {
   WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

   volatile int i;
```

```
  P1DIR |= LEDA + LEDB + LEDC + LEDD + LEDE + LEDF + LEDG + CATODO1;
  P2DIR |= CATODO2;
  P1OUT = 0;
  P2OUT = 0;
  for(;;)
  {
    //Aqui foi utilizado a Tabela ASCII para usar os digitos
    // 0,1,2,3,4,5,6,7,8,9 estão definidos entre 0x30 e 0x39 na tabela
    // A,B,C,D,E,F estão definidos entre 0x41 e 0x46
    for(i=0x30; i<=0x39; i++)
    {
      EscreveDigito(i);
      P1OUT ^= CATODO1;
      P1OUT ^= CATODO1;
      P2OUT ^= CATODO2;
      P2OUT ^= CATODO2;
    }
    for(i=0x41; i<=0x46; i++)
    {
      EscreveDigito(i);
      P1OUT ^= CATODO1;
      P1OUT ^= CATODO1;
      P2OUT ^= CATODO2;
      P2OUT ^= CATODO2;
    }
  }
  return 0;
}
```