

## Aula prática N.º 5

### Objetivos

- Implementar um sistema de visualização com dois *displays* de 7 segmentos.
- Mostrar informação nos dois *displays* de 7 segmentos, controlando a frequência de refresco do sistema de visualização.

### Introdução

A Figura 1 mostra a representação esquemática do *display* duplo de 7 segmentos, bem como a forma como está ligado aos portos do PIC32 na placa DETPIC32.

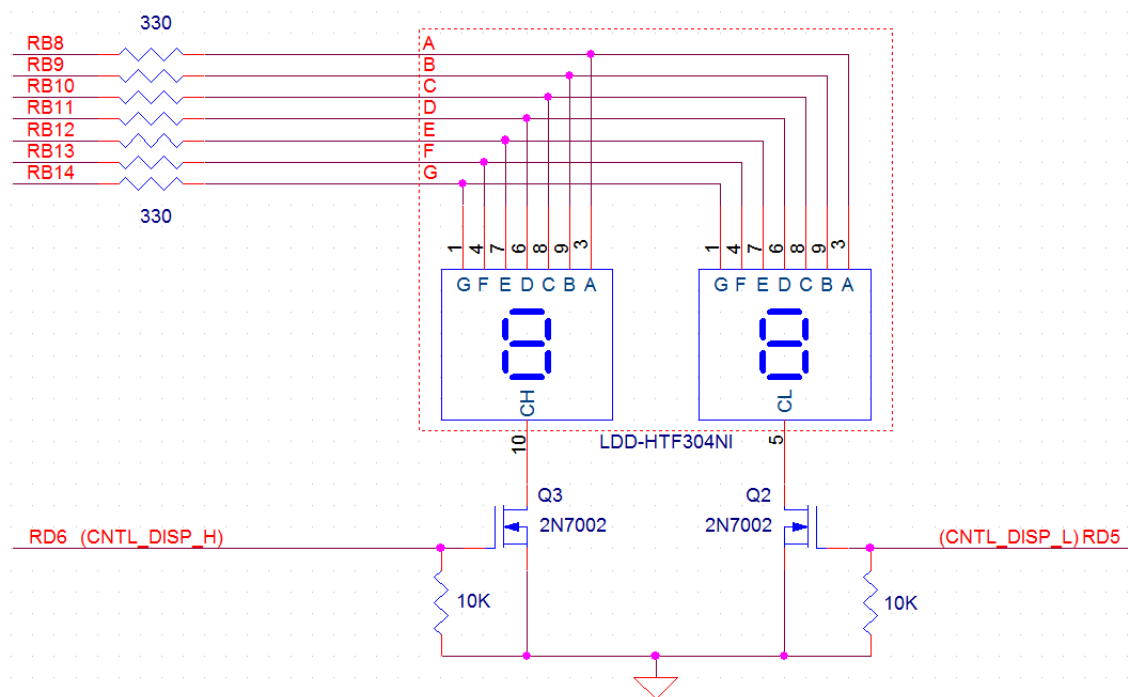


Figura 1. Ligação de dois *displays* de 7 segmentos ao porto B do PIC32.

A seleção/ativação de cada um dos *displays* é feita através dos portos **RD6** e **RD5**, configurados como saída: a ativação do porto **RD6** seleciona o *display high* e a ativação do porto **RD5** seleciona o *display low*.

### Trabalho a realizar

#### Parte I

1. O programa desenvolvido na aula anterior permite enviar 4 bits (um carácter hexadecimal) para um dos *displays*. Escreva agora uma função que envie um byte (8 bits) ou seja dois algarismos hexadecimais para os dois *displays*, fazendo corresponder os 4 bits menos significativos ao *display low* e os 4 bits mais significativos ao *display high*.

```
void send2displays(unsigned char value)
{
    static const char display7Scodes[] = {0x3F, 0x06, 0x5b, ...};
    // select display high
    // send digit_high (dh) to display:    dh = value >> 4
    // select display low
    // send digit_low (dl) to display:     dl = value & 0x0F
}
```

2. Escreva a função **main()** para teste da função **send2displays()**. A função **main()** deverá ter a seguinte estrutura:

```
int main(void)
{
    // configure RB8–RB14 as outputs
    // configure RD5–RD6 as outputs
    while(1)
    {
        send2displays(0x15);
        // wait 0.2s
    }
}
```

Teste o programa na placa.

Como pode observar, o sistema de visualização não apresenta o comportamento esperado (apenas 1 dos displays mostra o valor, estando o outro apagado).

3. Com a configuração hardware usada no sistema de visualização, é necessário enviar de forma alternada os valores para os dois *displays*. Se o tempo de ativação dos dois *displays* não for igual, o brilho exibido por cada um deles será também diferente (o que está menos tempo ativo terá um brilho inferior). O correto funcionamento desta configuração passa então por garantir que o tempo de ativação dos dois *displays* é aproximadamente o mesmo.

Reescreva a função **send2displays()** de modo a que, sempre que for invocada, envie, de forma alternada, apenas um dos dois dígitos para o sistema de visualização. Isto é, em chamadas sucessivas à função, o comportamento deverá ser: enviar "digit\_low", enviar "digit\_high", enviar "digit\_low", enviar "digit\_high", ...

```
void send2displays(unsigned char value)
{
    static const char display7Scodes[] = {0x3F, 0x06, 0x5b,...};
    static char displayFlag = 0; // static variable: doesn't loose its
                                // value between calls to function

    digit_low = value & 0x0F;
    digit_high = value >> 4;
    // if "displayFlag" is 0 then send digit_low to display_low
    // else send digit_high to display_high
    // toggle "displayFlag" variable
}
```

Teste novamente o programa na placa.

4. Com a solução implementada no ponto anterior o tempo de ativação de cada um dos *displays* ficou equilibrado, mas é visível a alternância entre os dois. Para resolver esse problema é necessário aumentar a frequência de trabalho do processo de visualização (frequência de refrescamento) de modo a que o olho humano não detete a alternância na seleção dos *displays*.

Aumente sucessivamente a frequência de refrescamento do sistema de visualização (usando a função **delay()**) para 20Hz, 50 Hz e 100 Hz, e observe os resultados.

5. Escreva um programa que implemente um contador ascendente módulo 256. O contador deve ser incrementado com uma frequência de 5 Hz e o seu valor deve ser mostrado nos dois *displays* em hexadecimal. A frequência de refrescamento do sistema de visualização deve ser 50 Hz.

A estrutura do programa deverá ser a seguinte:

```

int main(void)
{
    // declare variables
    // initialize ports
    counter = 0;
    while(1)
    {
        i = 0;
        do
        {
            send2displays(counter);
            // wait 20 ms (1/50Hz)
        } while(++i < ??);
        // increment counter (mod 256)
    }
    return 0;
}
ou:
int main(void)
{
    unsigned int i;
    ...
    counter = 0;
    i = 0;
    while(1)
    {
        send2displays(counter);
        // wait 20 ms (1/50Hz)
        i = (i + 1) % ??;
        if(i == 0)
            // increment counter
            (mod 256)
    }
    return 0;
}

```

6. Com a frequência de refrescamento usada no exercício anterior continua a notar-se a comutação entre os dois *displays*, efeito que é comum designar-se por *flicker*. De modo a diminuir, ou mesmo eliminar, o *flicker*, a frequência de refrescamento (*refresh rate*) tem que ser aumentada.

Mantendo a frequência de atualização do contador em 5Hz, altere o programa anterior de forma a aumentar a frequência de refrescamento de 50 Hz (20 ms) para 100 Hz (10 ms), e observe os resultados.

## Parte II

1. Mantendo a frequência de refrescamento em 100 Hz, altere o programa de modo a implementar um contador crescente módulo 60. A frequência de incremento deverá ser 2 Hz e os valores devem ser mostrados nos dois *displays*, em **decimal**, e nos 8 LED ligados o porto E, em BCD. A conversão para BCD pode ser feita, de forma simplificada e desde que o valor de entrada seja representável em BCD com 8 bits, pela seguinte função:

```

unsigned char toBcd(unsigned char value)
{
    return ((value / 10) << 4) + (value % 10);
}

```

2. Altere o programa anterior de modo a que o sistema se comporte como um contador módulo 60 up/down, em que o sentido de contagem seja dependente do valor lógico imposto pelo *dip-switch* 1 (DS1, ligado ao porto **RB0**): ON – contagem ascendente, OFF – contagem descendente.
3. Mantendo a frequência de refrescamento do sistema de visualização em 100 Hz, altere o programa de modo a que, em modo ascendente, a frequência de incremento do contador seja 5 Hz (e em modo descendente seja 2 Hz).
4. Altere o programa, que resultou do ponto anterior, de modo a que quando o contador atinge o seu valor limite (59 ou 00, dependendo do modo de funcionamento) seja ativado, durante 5s, o LED D11 ligado ao porto **RC14**. Durante esse tempo a contagem decorre normalmente.