18. TEMPORIZADORES/CONTADORES

Uma funcionalidade necessária em muitas aplicações é a contagem de tempo e a contagem de eventos externos. Embora estas acções se possam realizar através de instruções do CPU, sem recurso a dispositivos específicos, tal solução conduz por norma a programas muito complexos. Por esta razão, é usual nos sistemas baseados em microprocessadores dispormos de dispositivos com esta funcionalidade, normalmente denominados por temporizadores/contadores e que não são mais que registadores com função de contagem.

Se o sinal que dá origem à contagem for periódico (*clock*) com valor de T conhecido, recebe a denominação de temporizador, uma vez que se pode inferir um tempo a partir do número de impulsos registados. Se o sinal que dá origem à contagem for um conjunto de impulsos não periódicos, recebe a designação de contador, pois o valor registado é o número de impulsos que ocorreram num determinado intervalo de tempo ou entre dois quaisquer acontecimentos.

Estes dispositivos têm geralmente uma arquitectura programável, no sentido de facilmente se adaptarem às várias funções requeridas pelas aplicações.

18.1 TC (*Timer*/*Counter*)

A título de exemplo consideremos um *Timer* que apresenta a interface mostrada na Figura 18-1 e que denominaremos por TC_V1.

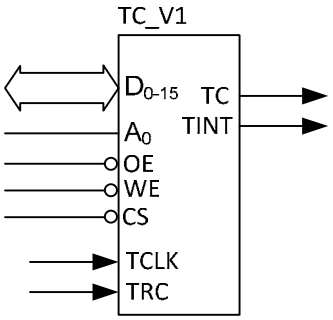


Figura 18-1

18.1.1 Especificação

O TC_V1 apresenta a seguinte especificação:

- Contagem crescente;
- Permite estabelecer um valor inicial de contagem;
- Permite parar e iniciar a contagem por software ou hardware;
- Disponibiliza por software e hardware informação da passagem do contador pelo limite;
- Auto carregamento (atingido o limite de contagem recarrega o valor pré estabelecido);
- Permite a leitura do valor do contador.

O TC_V1 apresenta, do ponto de vista do utilizador, quatro registos com a seguinte função:

Data - 16 bits onde se estabelece o valor inicial de contagem;

CTR - 3 bits para estabelecer o modo de funcionamento e controlo sobre o TC_V1, com a seguinte funcionalidade:

	D_2	D_1	D_0
CTR	autoRL	TR	gate

Gate quando activo permite a paragem e arranque através do sinal externo TRC.
TR paragem e arranque da contagem.
AutoRL configura o contador em modo auto carregamento, ou seja, quando o contador atinge o limite é carregado no registo **count** o valor pré estabelecido no registo **Data**.

Count - 16 bits que contêm o valor corrente de contagem. O facto de a contagem ser crescente, implica que quando se pretenda utilizar a funcionalidade de temporizador, o valor inicial a ser carregado deverá ser a diferença para o limite (0 – *time*);

TF - 1 bit que indica a passagem do contador pelo limite e que é levado a zero quando é lido o valor do registo **Count**.

Na Figura 18-1 podemos identificar os seguintes sinais:

TCLK- (*Timer Clock*) sinal de entrada que a cada transição ascendente promove o incremento do contador;

TRC - (*Timer Run Control*) sinal de entrada que permite inibir e desinibir a contagem. Através deste sinal podemos estabelecer uma janela de contagem e assim medir a sua largura em unidades de períodos de TCLK;

TC - (*Terminal Count*) sinal de saída que fica activo durante a passagem do contador pelo valor limite, permitindo assim utilizar o *Timer* como divisor de frequência programável;

TINT - (*Timer Interrupt*) sinal de saída que fica activo quando o contador atinge o limite. Este sinal só fica novamente desactivo quando o valor do contador for lido pelo CPU.

18.1.2 Diagrama de blocos

Na Figura 18-2 é apresentado o diagrama de blocos de uma possível arquitectura que implementa o TC_V1.
O circuito *Decoder* função de A_0, WE e OE gera quatro sinais: dois para selecção de escrita e dois para selecção de leitura. Descrição do comportamento da estrutura:
Após estabelecido em **CTR** o modo de funcionamento (*Auto Run*/gate), estabelece-se em **Data** o valor inicial de contagem. Esta acção promove simultaneamente a passagem a *Load* do *flip-flop* **LC** (*Load Control*) garantindo desta forma que quando for iniciada a contagem através do sinal **TR** a primeira transição ascendente de **TCLK** promove o carregamento de **Count** com o valor estabelecido em **Data**. O *flip-flop* **LC** fica a zero na primeira transição descendente de **TCLK** passando a partir daqui o registo **Count,** através do conjunto Multiplexer/Somador, a funcionar como contador crescente.
Atingido o limite de contagem, o sinal **TC** (*Terminal count*) fica activo e caso o *Timer* se encontre em modo *Auto Reload* acontece o seguinte: passado meio período de **TCLK** o *flip-flop* **LC** fica novamente a 1 (modo *Load*); na transição ascendente de **TCLK** recarregar o registo **Count** com o valor de **Data**; ao recarregar **Count** o sinal **TC** fica desactivo; passado meio período de **TCLK** o *flip-flop* **LC** passe novamente a zero (modo de contagem).
Caso o *Timer* não se encontre programado em modo *Auto Reload* o contador ao atingir o valor limite continua a contagem a partir do valor zero ficando assinalado o facto no *flip-flop* **TF** (*Terminal Flag*).
A informação de que o contador passou pelo limite, além de ficar disponível em **TINT** (*Timer Interrupt*), pode ser consultada por software pela leitura do registo **TF**. O registo **TF** é posto a zero por software com a leitura do registo **Count**. A leitura do registo **Count**, em modo *timer* pode ser importante, pois permite saber quantas transições do sinal **TCLK** existiram depois do sinal **TF** ter ficado activo.

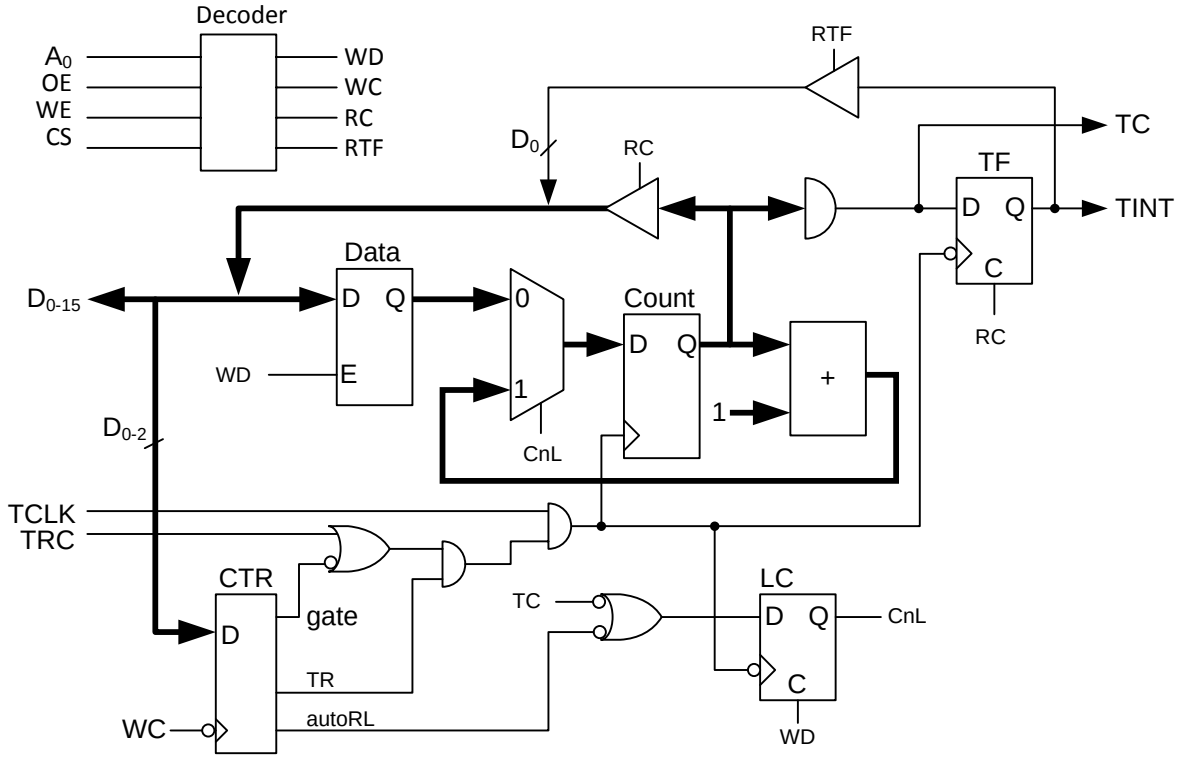


Figura 18-2

18.1.3 Exercícios

EXEMPLO 1:

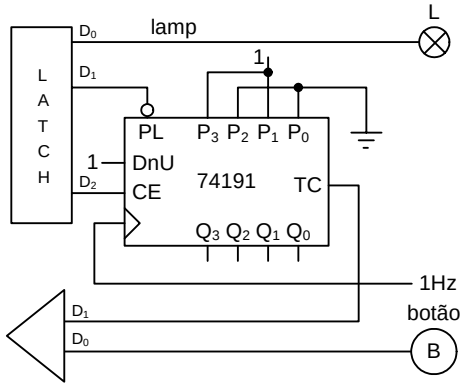
Baseado no sistema SDP16 integrando um PDS16, pretende-se implementar o sistema de controlo de uma lâmpada **L** com temporização accionada por um botão **B** segundo o esquema seguinte:

- ao premir o botão **B** a lâmpada **L** acende-se durante 10 segundos;
- se o botão **B** for novamente activado durante o intervalo de tempo em que a lâmpada **L** se encontra acesa o tempo reinicia-se a partir desse momento.

Para a realização dispõe de um porto paralelo de entrada e outro de saída, um contador 74191 de quatro bits com entrada de PL e CE.

Nota: A solução adoptada utiliza o contador como temporizador de 10s a partir de um sinal periódico com a frequência de 1Hz.

```
for(;;) {
  getTransB(&port, BLOC);
  lamp=TRUE;
  setCount10();
  startCount();
  do {
    if (getTransB(&port, NO_BLOC)
      setCount10();
  while (!TC)
  lamp=FALSE;
}
```




```
      .equ  LAMP, 1
      .equ  PL, 2
      .equ  CE, 4
      .equ PORT_ADDR, 0xff
      .equ B_POS, 1
      .equ TC_POS, 2
      .equ  BLOC,1
      .equ  NO_BLOC,0

      .section start
      .org  0
      jmp   main

      .section my_dada
      .org  2
estado:
      .space      1
```

```
      .section main
      .org  10
getTransB:              ;boolean getTransB(char *port, boolean bloc)
      orl   r1,r1,r1
      jnz   get_block
      ldb   r1,estado
      orl   r1,r1,r1
      jz    wait_trans_up
      ldb   r2,[r0,#0]
      shr   r2,r2,#B_POS,0    ;cy=B
      jc    no_trans          ;no transition
      ldi   r0,#0       ;transition down
      stb   r0,estado
no_trans:
      ldi   r0,#0
      ret
wait_trans_up:
      ldb   r2,[r0,#0]
      shr   r2,r2,#B_POS,0    ;cy=B
      jnc   no_trans          ;no transition
      ldi   r0,#1
      stb   r0,estado
      ldi   r0,#1             ;return true
      ret
get_block:
      ldb   r1,[r0,#0]
      shr   r1,r1,#B_POS,0    ;cy=B
      jc    get_block         ; while(B)
get_block_1:
      ldb   r1,[r0,#0]
      shr   r1,r1,#B_POS,0    ;cy=B
      jnc   get_block_1       ; while(!B)
      ldi   r0,#0             ;return false
      ret

main: ldih  r0,#high(PORT_ADDR)
      ldi   r1,#0             ;lamp=0; PL=0; CE=0
      stb   r1,[r0,#0]
      ldi   r1,#1
      stb   r1,estado         ; inicia estado do botão B
      ldi   r1,#BLOC
      jmpl  getTransB
      ldi   r1,#LAMP+CE+PL    ;lamp=1; PL=1; CE=1;
      stb   r1,[r0,#0]
main_2:
      ldih  r0,#high(POR_ADDR)
      ldi   r1,#NO_BLOC
      jmpl  getTransB
      orl   r0,r0,r0
      jz    main_1
      ldih  r0,#high(POR_ADDR)
      ldi   r1,#LAMP+CE       ;reload timer PL=0
      stb   r1,[r0,#0]
```

```
      ldi   r1,#LAMP+CE+PL
      stb   r1,[r0,#0]
main_1:
      ldih  r0,#high(POR_ADDR)
      ldb   r1,[r0,#0]
      shr   r1,r1,#TC_POS
      jnc   main_2
      jmp   main
```

EXEMPLO 2:

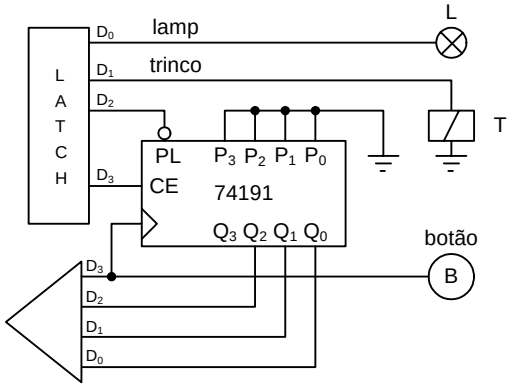
Baseado no sistema SDP16 integrando um PDS16, pretende-se implementar o sistema de controlo de abertura de uma fechadura electrónica accionada por um botão **B** segundo o esquema seguinte:

- ao premir o botão **B** acende-se uma lâmpada **L** durante 5 segundos;
- durante este intervalo de tempo é necessário premir o botão **B** um número n (1 < n < 8) de vezes;
- findo o tempo o trinco **T** é accionado se o número de actuações de **B** coincidir com um valor pré estabelecido.

Para a realização dispõe de um porto paralelo de entrada e outro de saída, um contador 74191 de quatro bits com entrada de PL e CE.

Nota: A solução adoptada utiliza o contador 191 como contador de impulsos do botão **B**

```
for(;;) {
  clearCount();
  getTransB();
  lamp=TRUE;
  startCount();
  delay(5);
  lamp=FALSE;
  if (cont=SECRET_COUNT){
    trinco=TRUE;
    delay(1);
    trinco=FALSE;
  }
}
```




```
      .equ  LAMP, 1
      .equ  TRINCO, 2
      .equ  PL, 4
      .equ  CE, 8
      .equ  secret_count, 5
      .equ PORT_ADDR, 0xff
      .equ B_POS, 4

      .section start
      .org  0
      jmp   main
```

```
      .section main
      .org  10

      .equ  MIL_MS, 62  ; frequência de MCLK 500H
delay:ldi   r1,#MIL_MS
delay_1:
      dec   r1          ;4 * 2ms
      jnz   delay_1     ;4 * 2ms
      dec   r2
      jnz   delay
      ret

getTransB:
      ldb   r1,[r0,#0]
      shr   r1,r1,#B_POS,0    ;cy=B
      jc    getTransB         ; while(B)
getT_1:
      ldb   r1,[r0,#0]
      shr   r1,r1,#B_POS,0    ;cy=B
      jnc   getT_1            ; while(!B)
      ret

main: ldih  r0,#high(PORT_ADDR)
main_1:
      ldi   r1,#0             ;reset contador
      stb   r1,[r0,#0]
      jmpl  getTransB
      ldi   r1,#LAMP+CE+PL    ;lamp=1; trinco=0; PL=1; CE=1;
      stb   r1,[r0,#0]
      ldi   r2,#5
      jmpl  delay
      ldi   r1,#PL            ;lamp=0; trinco=0; PL=1; CE=0;
      stb   r1,[r0,#0]
      ldb   r1,[r0,#0]        ; ler contador
      shl   r1,r1,#12,0
      shr   r1,r1,#12,0
      ldi   r2,#secret_count
      sub   r1,r1,r2
      jne   main_1
      ldi   r1,#TRINCO        ;lamp=0; trinco=1; PL=0; CE=0;
      stb   r1,[r0,#0]
      ldi   r2,#1
      jmpl  delay
      jmp   main_1
```