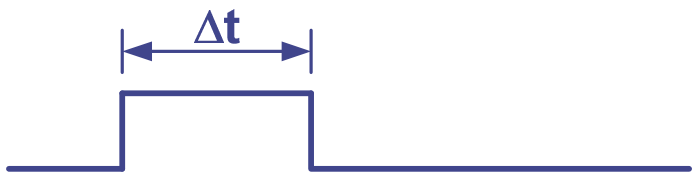
Aula 10

- *Timers*
  - Aplicações
  - Princípio de funcionamento
  - Divisão de frequência e controlo do *duty-cycle*
- *Watchdog timer*

José Luís Azevedo, Bernardo Cunha, Tomás O. Silva, P. Bartolomeu

Introdução

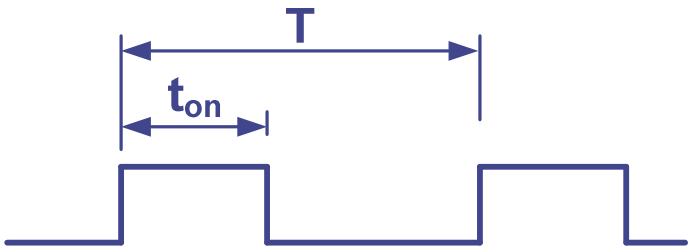
- Um *timer* é um dispositivo periférico de suporte que permite, no essencial, a medição de tempo, partindo de uma referência temporal conhecida
- Alguns exemplos de aplicações típicas de *timers*:
  - geração de um evento com uma duração controlada;
    exemplo: geração de um impulso com uma duração definida, Δt



  - geração de um evento periódico com período controlado;
    exemplo: geração de um impulso com um período definido, T

Introdução

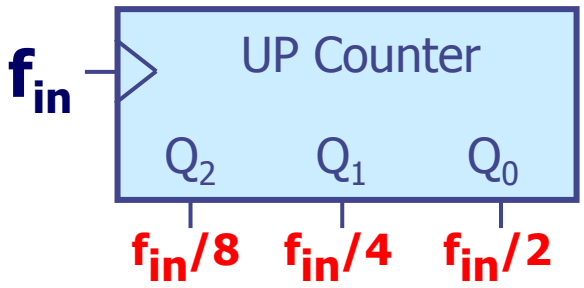
- Geração de um evento periódico com período e duração controlados. Exemplo: geração de um sinal periódico com um período de 10 ms e um "duty-cycle" de 40%:



Duty-cycle **= (t_{on} / T) * 100 [%]**

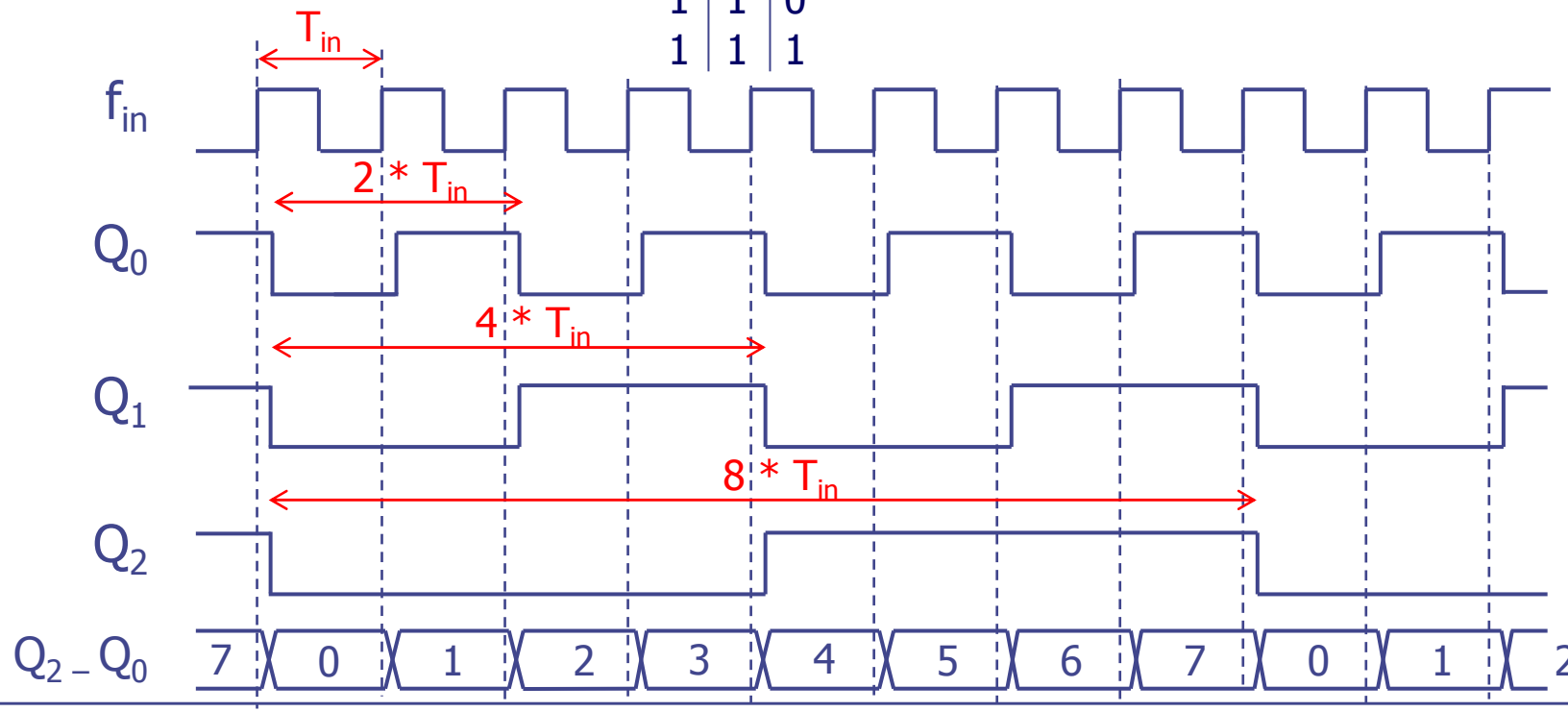
  - "ton" é o tempo durante o qual o sinal está no nível lógico 1, num período
  - a possibilidade de alterar o valor de "ton" sem alterar o valor de T é útil em muitas situações e designa-se por **PWM** (***Pulse Width Modulation*** – modulação por largura de pulso)
- O funcionamento dos *timers* baseia-se sempre na contagem de ciclos de um sinal de relógio com frequência conhecida

Divisão de frequência

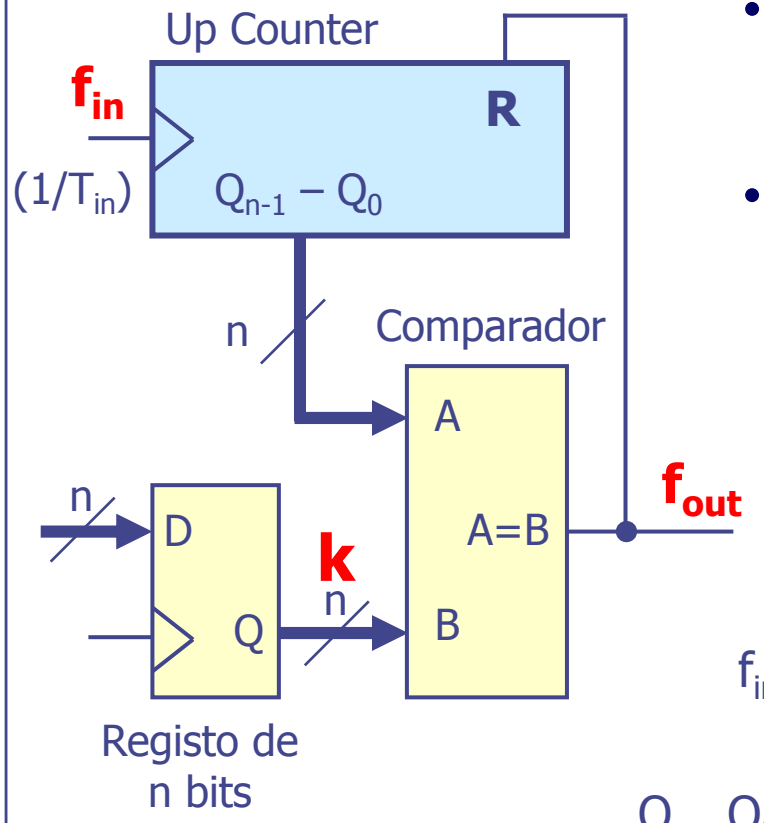


Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

- O fator de divisão é fixo: 2, 4, 8, 16, ... (2^n c/ $n \geq 1$)
- Como dividir a frequência do sinal de entrada por outros valores? ($f_{out} = f_{in} / k$)

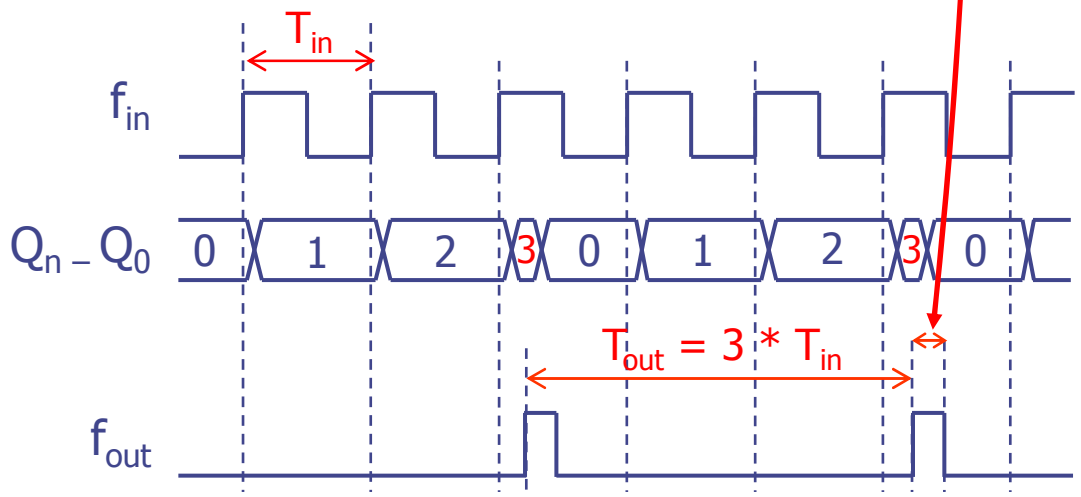


Divisão de frequência (versão 1)



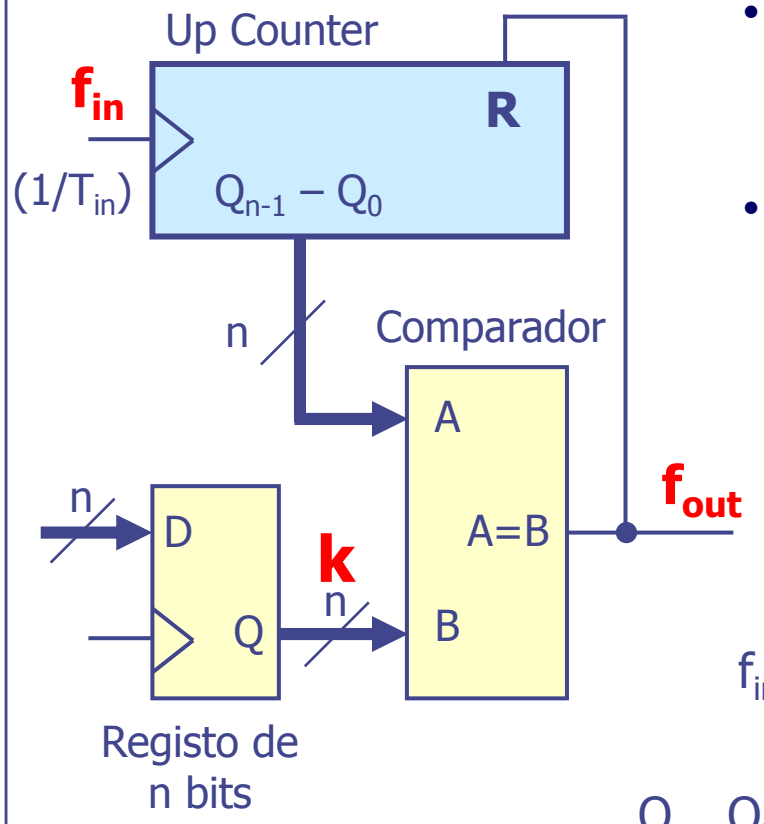
- **Reset do contador assíncrono**: o tempo a 1 do sinal **R** não é controlado (2 atrasos de propagação)
- O período do sinal de saída é: $T_{out} = k * T_{in}$, ou seja:
  - $f_{out} = f_{in} / k$

Exemplo com k = 3



$$k = \frac{f_{in}}{f_{out}}$$

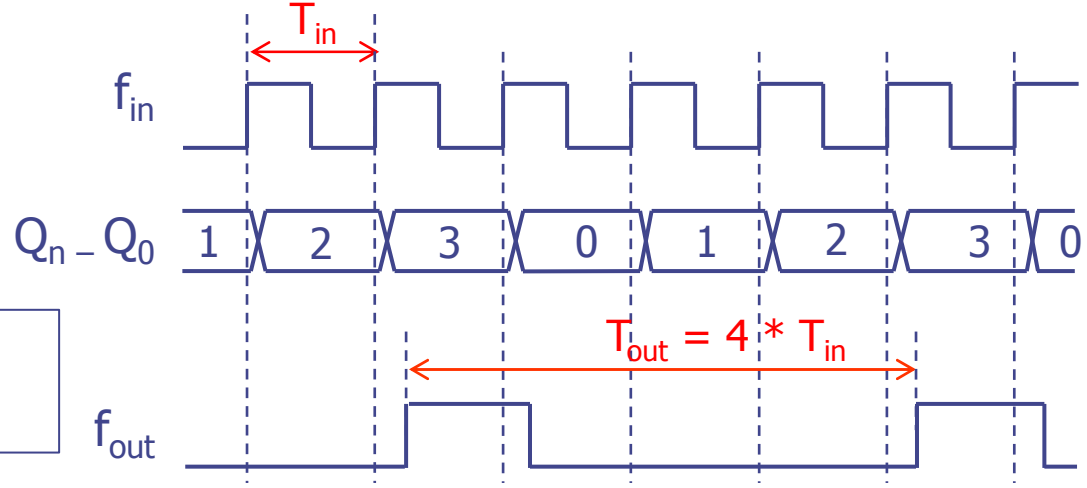
Divisão de frequência (versão 2)



- **Reset do contador síncrono**: o tempo a 1 do sinal de saída (f_{out}) é 1 ciclo de relógio do sinal de entrada (T_{in})
- O período do sinal de saída é:
  $T_{out} = (k+1) * T_{in}$, ou seja:
  - $f_{out} = f_{in} / (k+1)$
  - $k = (f_{in} / f_{out}) - 1$

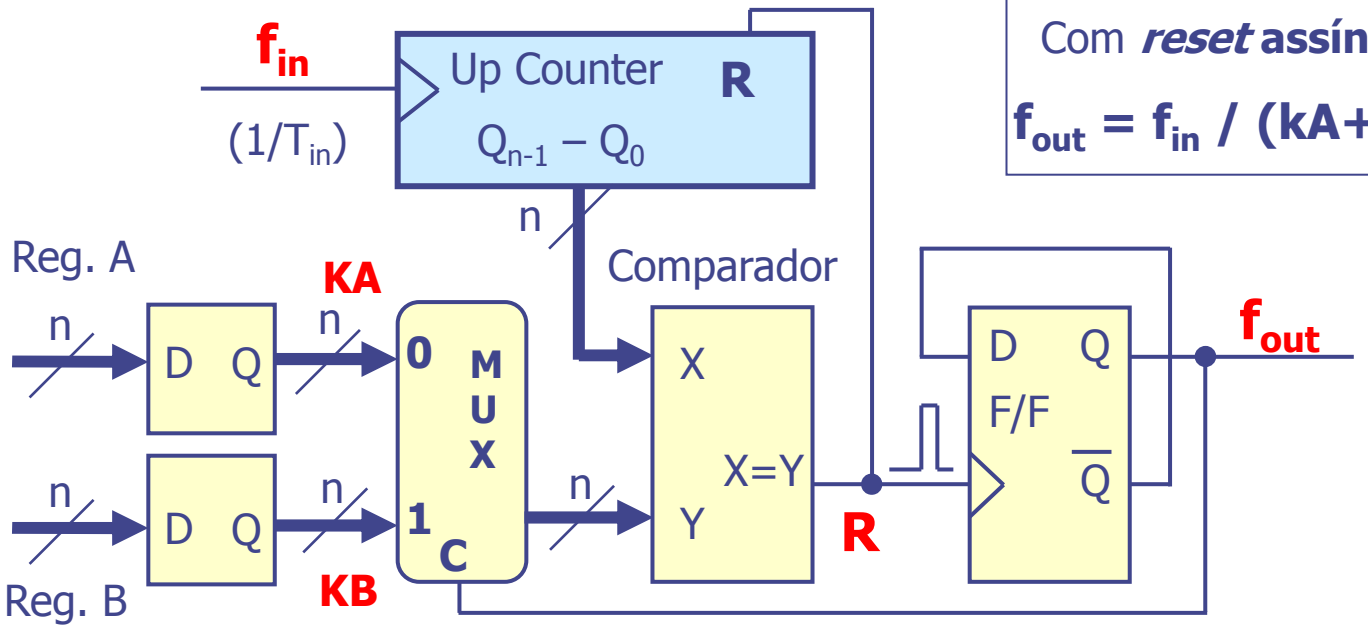
$$k = \frac{f_{in}}{f_{out}} - 1 \quad (k > 0)$$

Exemplo com k = 3



Controlo de frequência e *duty-cycle*

- Para este tipo de aplicação, o timer tem que permitir o controlo do período do sinal de saída, bem como do tempo durante o qual esse sinal está a 1

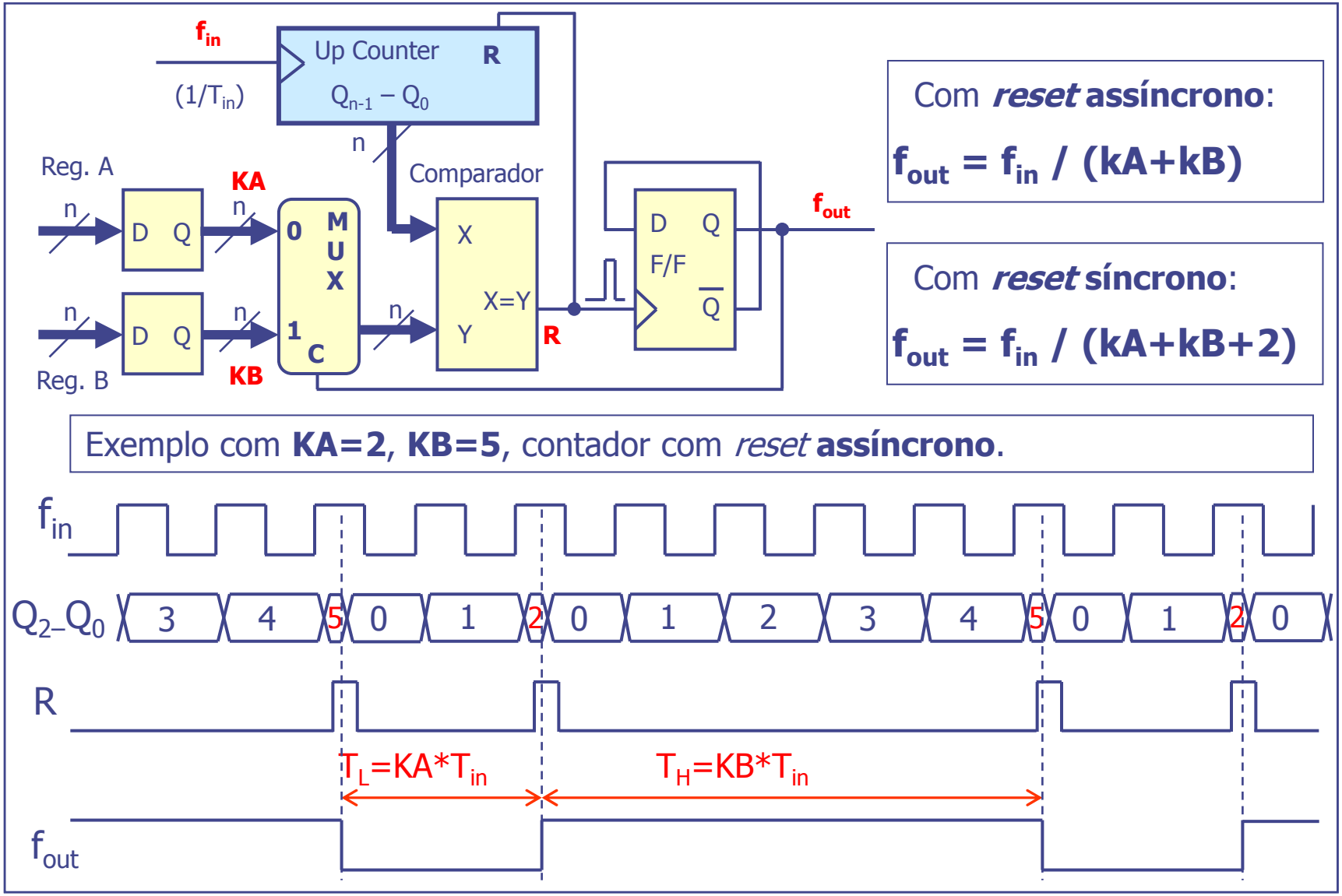


Com ***reset* assíncrono**:

$f_{out} = f_{in} / (kA+kB)$

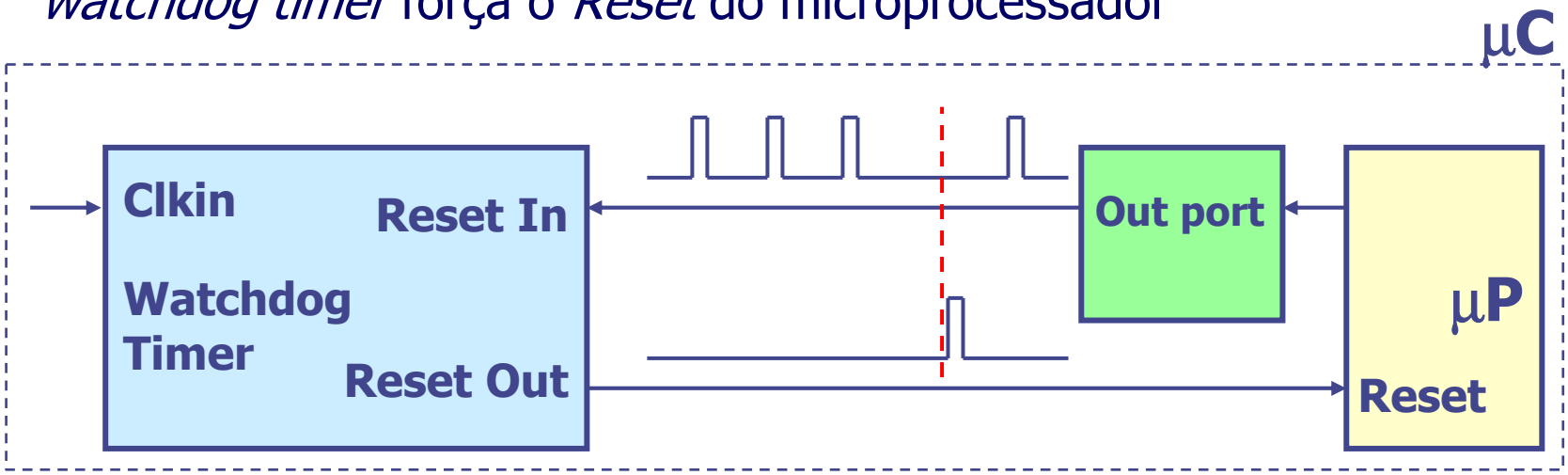
- Quando a saída Q do flip-flop está a 1, a entrada Y do comparador toma o valor de KB; caso contrário, toma o valor de KA. Logo, o tempo durante o qual o sinal de saída está a 1 depende de KB, e durante o qual está a 0 depende de KA

Controlo de frequência e *duty-cycle*

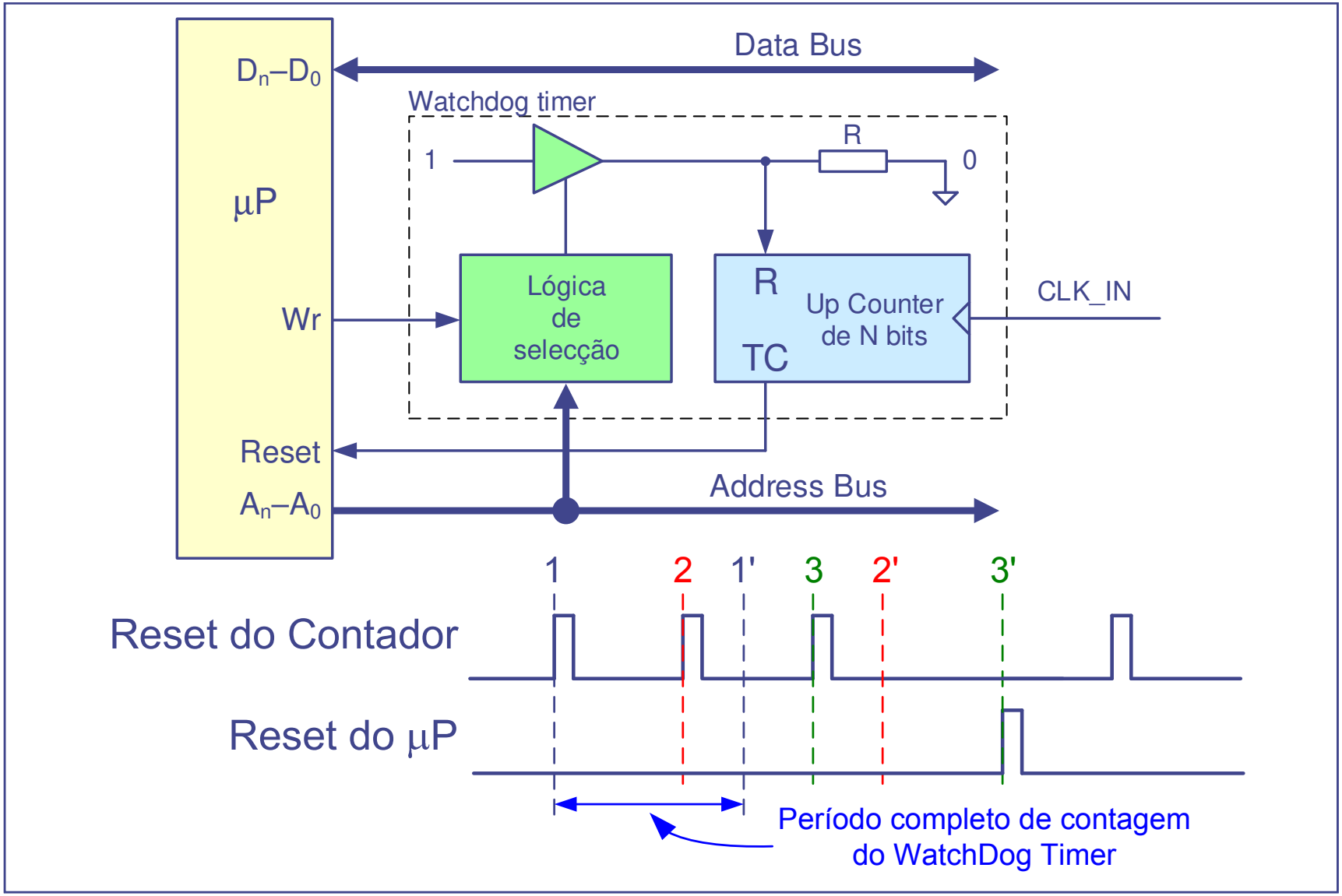


Watchdog Timer (temporizador "cão de guarda")

- Sistemas baseados em microcontrolador podem assegurar funções de controlo críticas que não podem falhar
- Como garantir que um crash do microprocessador não compromete o funcionamento global do sistema?
- Um *watchdog timer* tem como função monitorizar a operação do microprocessador e, em caso de falha, forçar o seu reinício
- Situação mais comum: se o microprocessador não atuou a entrada de *Reset* do *watchdog timer* ao fim de um tempo pré-determinado o *watchdog timer* força o *Reset* do microprocessador



Watchdog Timer – exemplo de implementação



Watchdog Timer – exemplo de utilização

- A aplicação no microcontrolador executa em ciclo infinito
- O *watchdog timer* é ativado quando o programa inicia. O *reset* da sua contagem é feito regularmente no corpo do ciclo (no exemplo, **`clearWatchdogTimer()`**)

```
void main(void)
{
        enableWatchdogTimer();
        (...)
        while(1)
        {
                (...)
                clearWatchdogTimer();
        }
}
```

- Caso haja uma falha no microprocessador que implique a quebra de execução do ciclo, a função "clearWatchdogTimer()" deixa de ser chamada e o *watchdog timer* deixa de ser reiniciado:
  - Quando o contador do *watchdog timer* atinge o valor máximo da contagem força um *reset* ao microprocessador