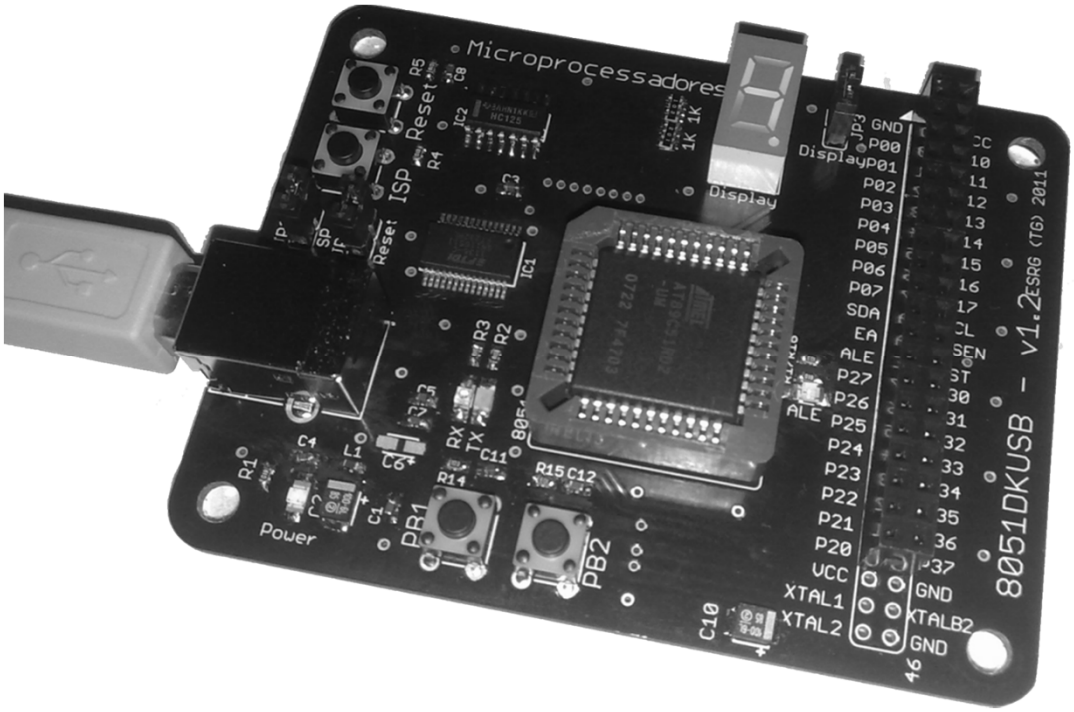
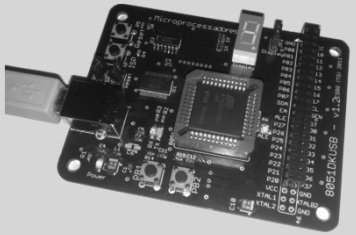
Mestrado Integrado em Eng. Electrónica Industrial e Computadores



Unidades Contadoras e/ou Temporizadoras

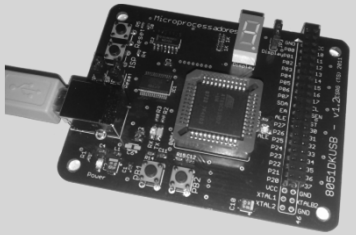
Microcontroladores

2º Ano – A08



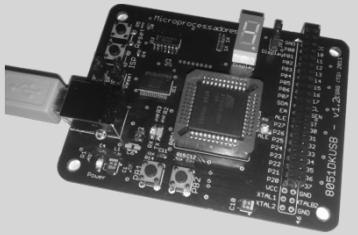
Directivas

Categoria	Directiva	Síntaxe			Função
Controlo do estado	**ORG**		**ORG**	expressão	Especifica um valor para contador de localização do segmento activo
	END		**END**		Indica ao *assembler* o fim do programa fonte
	USING		**USING**	expressão	Indica ao *assembler* o banco de registo usado no código que vem a seguir à directiva. Repare que a comutação do banco de registo deve ser efectuada usando apenas instruções do 8051
Definição de símbolos	**SEGMENT**	Símbolo	**SEGMENT**	tipo_de_segmento	Declara um símbolo como sendo um segmento *relocatable* de um dado tipo. Para começar a usar o segmento, deve-se usar a directiva RSEG
	EQU	Símbolo	**EQU**	expressão	Atribuí um valor a um símbolo
	SET	Símbolo	**SET**	expressão	Igual ao EQU, exceptuando o facto de permitir a redefinição o símbolo
	DATA	Símbolo	**DATA**	expressão	Atribui ao símbolo um endereço directo da RAM interna
	IDATA	Símbolo	**IDATA**	expressão	Atribui um endereço da RAM interna indirectamente endereçável ao símbolo
	XDATA	Símbolo	**XDATA**	expressão	Atribui ao símbolo um endereço da memória externa
	BIT	Símbolo	**BIT**	expressão	Atribuí um endereço directo da área de memória endereçável ao bit a um símbolo
	CODE	Símbolo	**CODE**	expressão	Atribuí um endereço da memória de código ao símbolo



Directivas

Categoria	Directiva	Síntaxe			Função
Incialização e reserva de armazenamento	**DS**	[LABEL:]	**DS**	expressão	Reserva espaços em múltiplos de *bytes*. Não pode ser utilizado com segmento do tipo BIT. O valor da expressão deve ser conhecida pelo *assembler*
	DBIT	[LABEL:]	**DBIT**	expressão	Reserva espaços em múltiplos de bits. O valor da expressão deve ser conhecida pelo *assembler*
	DB/DW	[LABEL:]	**DB/DW**	expressão	Inicializa a memória de código com valores do tipo byte/word
Program linkage	**PUBLIC**		**PUBLIC**	Símbolo [, símbolo] [...]	Define uma lista de símbolos que tornam visíveis e utilizáveis a partir de outros módulos
	EXTRN		**EXTRN**	Tipo_segmento(símbolo [,símbolo] [...], ...)	Informa o *assembler* da lista de símbolos definidos noutros módulos e que vão ser utilizados neste. O tipo de segmento pode ser CODE, DATA, XDATA, IDATA, BIT e um especial designado por NUMBER que especifica um símbolo definido por EQU
	NAME		**NAME**	Nome_do_módulo	
Selecção de Segmentos	**RSEG**		**RSEG**	Nome_do_segmento	Ao encontrar uma directiva de selecção de segmento, o *assembler* direcciona o código
	CSEG		**CSEG**	[**AT** endereço]	ou dado que lhe segue para o segmento
	...		**DSEG**	[**AT** endereço]	seleccionado até que seja seleccionado um
	XSEG		**XSEG**	[**AT** endereço]	outro segmento



Exemplos

```
01  #include <89C51Rx2.inc>
02
03  VAR1     DATA     30H
04  VAR2     DATA     31H
05  VAR3     DATA     32H
06
07  DSEG     AT       40H
08  VAR4:    DS   1
09  VAR5:    DS   1
10  VAR6:    DS   1
11
12  VARI1    IDATA    80H
13  VARI2    IDATA    81H
14
15  VAREXT   XDATA    100H
21      MOV      A,#55H
22      MOV      VAR1,A
23      INC      A
24      MOV      VAR2,A
25      CPL      A
26      MOV      VAR6,A
27      MOV      R0,#VARI1
28      MOV      @R0,#55H
```

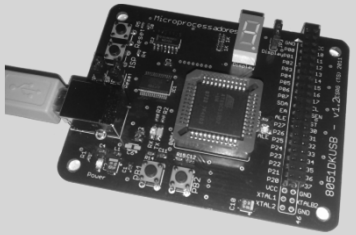
Watch 1

Name
VARI1
VAREXT
<double-click or F2 to add>

```
01  #include <89C51Rx2.inc>
02
03  BSEG     AT   0H
04  BITON:       DBIT     1
05  LEDVERDE:    DBIT     1
06  FLAG3:       DBIT     1
07  MOTORESQ:    DBIT     1
08
09  XSEG     AT   300H
10  VAR1:    DS   1
11
12  CSEG     AT   0H
13      JMP MAIN
14  CSEG     AT   50H
15  MAIN:
16      MOV      A,#55H
17      MOV      20H,A
18      SETB     LEDVERDE
19      CLR      FLAG3
20      MOV      A,20H
```

Watch 1

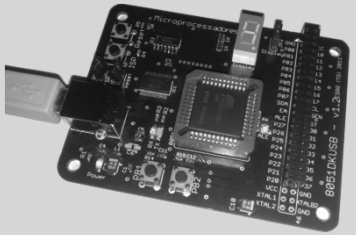
Name
LEDVERDE
BITON
FLAG3
<double-click or F2 to add>



Stack Pointer Register

SP

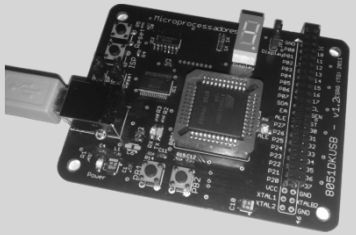
- *Registo de 8-bit (endereço 81H SFR RAM interna)*
- *Contém o endereço do item colocado no topo da pilha (*stack*). Valor após reset, 07H;*
- *Duas instruções permitem manipular a stack: PUSH e POP*
  - *A operação de PUSH coloca um dado na stack, enquanto a operação de POP retira o dado*



Stack Pointer Register

- *A stack cresce no sentido ascendente da memória*
- *Após o estado de reset este registo aponta para a posição 07H*

Pergunta:
Ao fazer o primeiro PUSH, onde é colocado o dado?

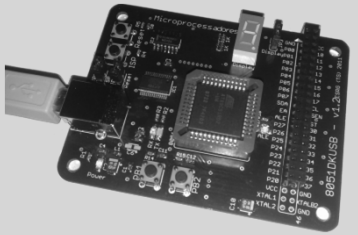


Stack Pointer Register

Atenção:

- *Como a stack após o reset está localizada na posição 7H, que corresponde a zona dos bancos de registos, é conveniente mudar a sua localização caso queiramos usar os bancos.*

Por ex.: MOV SP, #7FH



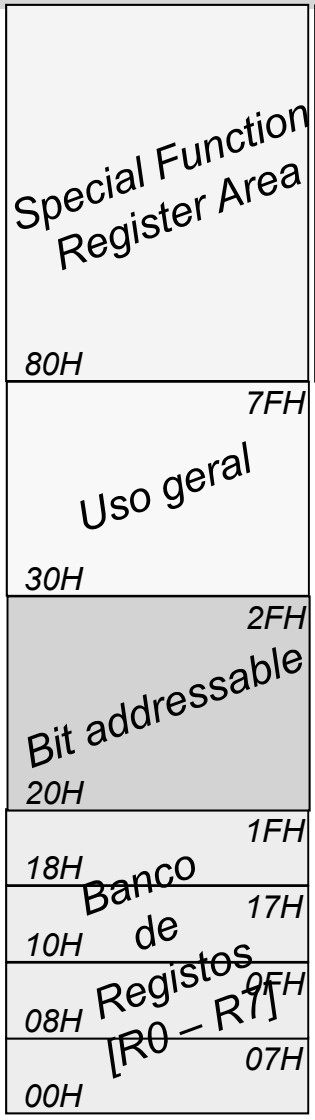
Stack Pointer Register

Atenção: Exemplo

Nota #2:
A pilha pode ser inicializada para a posição 7FH.
Como o acesso é indirecto através do SP, a pilha é implementada na IDATA e o espaço de SFR está salvaguardado.

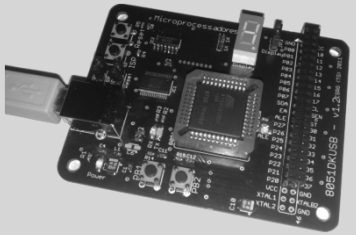
Atenção #1:
Ao ser colocado valores na pilha o segundo banco será afectado

Posição apontada por SP após reset – 07H



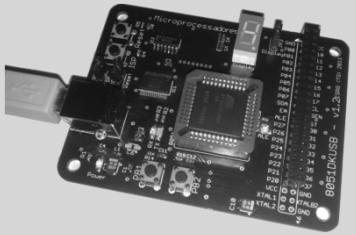
Exemplo:
Inicializamos a SP para

MOV SP, #7FH



Stack Pointer Register

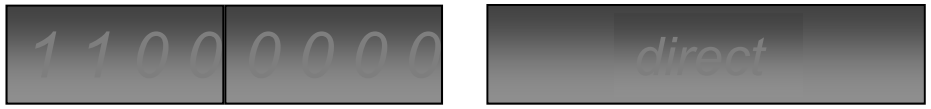
- ***Stack***
  - *O conjunto de instruções do 8051 fornece duas operações para manipulação da stack:*
    - ✓ ***PUSH: insere um dado/valor na stack***
    - ✓ ***POP: retira o último dado/valor inserido na stack***



Stack Pointer Register

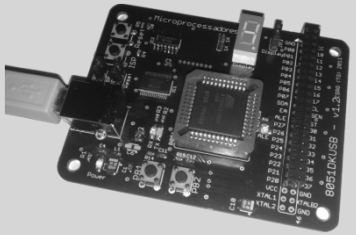
- ***Stack: PUSH***

2 bytes
2 cycles



- *Na operação de PUSH,*
  1. *O valor do stack pointer é incrementado em uma unidade.*
  2. *O conteúdo da posição da RAM indicada como argumento da operação é copiado para a posição apontada pelo stack pointer.*
  3. *Nenhuma flag é afectada.*

$$SP \longleftarrow SP + 1$$
$$(SP) \longleftarrow (directo)$$



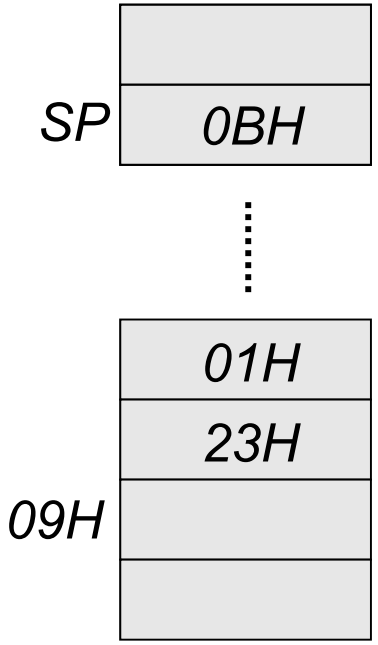
Stack Pointer Register

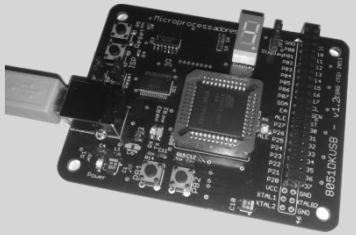
- ***Stack: PUSH - exemplo***

Um fragmento de programa tem o registo DPTR inicializado à 0123H e a stack pointer aponta para a posição 09H.
Explique qual é o estado da stack após o push do registo DPTR.

Programa:

```
PUSH   DPL
PUSH   DPH
```





Stack Pointer Register

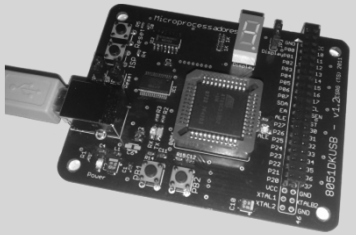
- ***Stack: POP***

2 bytes
2 cycles

1 1 0 1 0 0 0 0	direct

- *Na operação de POP,*
  1. *O conteúdo da posição da RAM interna apontada pelo stack pointer é lido e o valor do stack pointer é decrementado.*
  2. *O valor lido é carregado na posição da RAM indicada como argumento da operação.*
  3. *Nenhuma flag é afectada.*

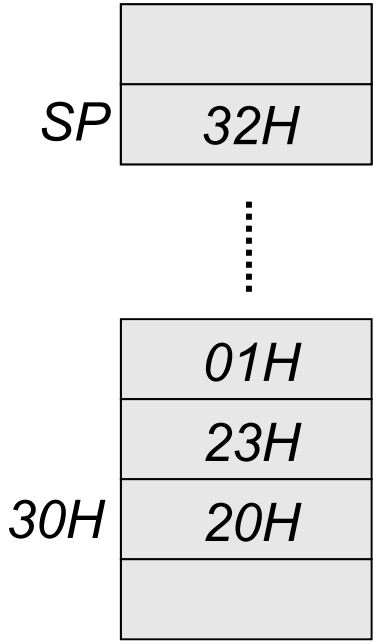
$$(directo) \longleftarrow ((SP))$$
$$(SP) \longleftarrow (SP) - 1$$



Stack Pointer Register

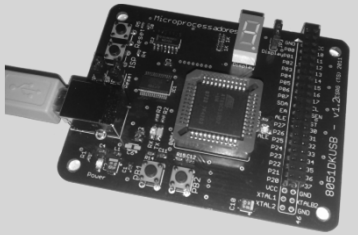
- ***Stack: POP - exemplo***

Analise o seguinte fragmento de programa e explique que valores assumirão os registos DPH, DPL e SP após a sua execução.



Programa:

```
POP   DPH
POP   DPL
POP   SP
```



Stack Pointer Register

- STACK – porquê?

Ao programar necessitamos de utilizar rotinas;

A invocação de rotinas no 8051 é feita utilizando as instruções **ACALL** ou **LCALL**;

O retorno de uma rotina é feito usando a instrução **RET**;

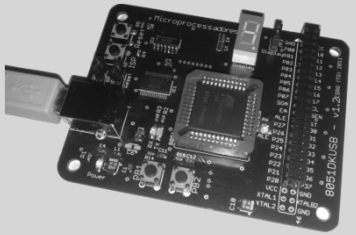
A *stack* pode ser utilizada para passar parâmetros às rotinas;

O que faz o 8051 ao executar uma instrução de CALL?

Guarda na *Stack* o valor do *Program Counter* (PC: PCH e PCL), que é o endereço na memória de código da próxima instrução a executar;

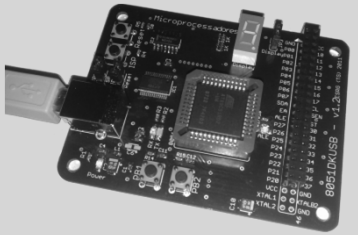
Carrega para o PC o endereço da rotina – começa a executar as instruções da rotina;

Ao executar a instrução RET, carrega para o PC o valor armazenado na *Stack*, ou seja, o endereço da instrução seguinte ao CALL.



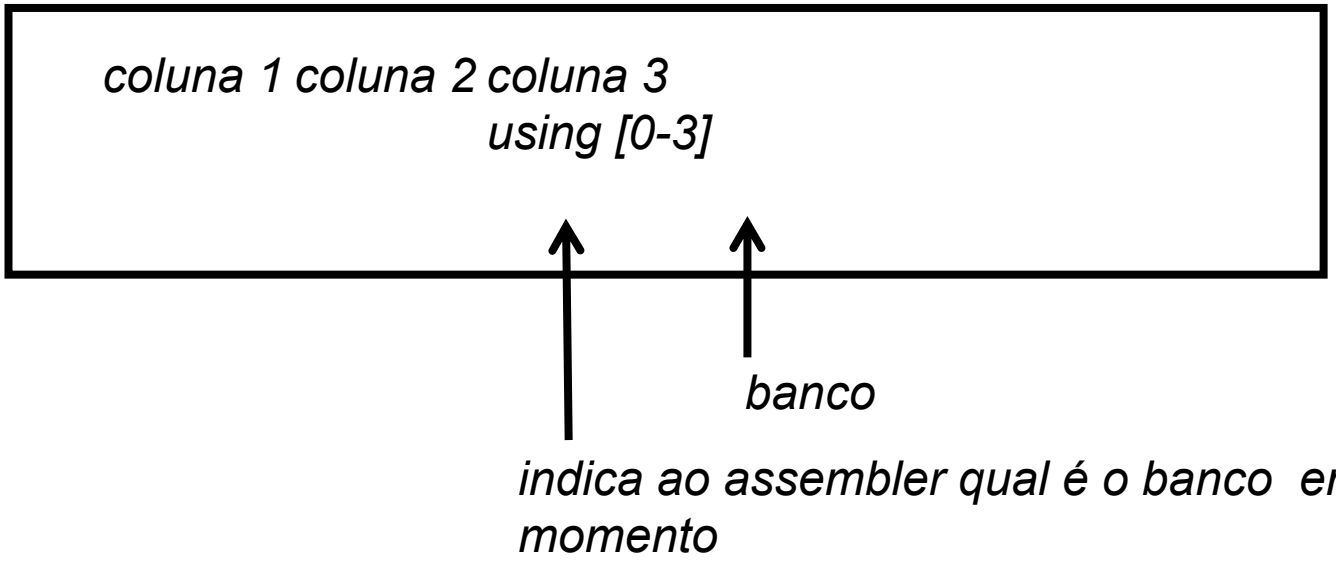
Stack Pointer Register

- ***Directiva 'USING'***
  - *As instruções de PUSH e POP recebem como argumento o endereço da posição da memória interna cujo conteúdo será guardado na* stack*.*
  - *Isto significa que teremos que calcular o endereço dos registos Rn, n* $\in$ *[0-7], quando estivermos a trabalhar com diferentes bancos.*
  - *O assembler do 8051 fornece a directiva 'using' para facilitar a programação.*

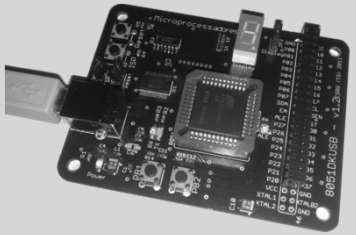


Stack Pointer Register

- ***Directiva 'USING'***



- *No código uso as labels ARn, n ∈ [0 – 7], para os registos Rn, n ∈ [0 – 7], ou ACC para representar o acumulador, A.*



Stack Pointer Register

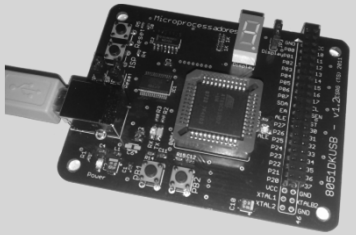
- ***Directiva 'USING': Exemplo***

```
USING 3
PUSH AR0

USING 1
PUSH AR7
```

- *O primeiro push colocará o conteúdo da posição 1FH na stack (endereço de R0 no banco 3).*
- *O segundo push colocará o conteúdo da posição 0FH na stack (endereço de R7 no banco 1).*

Nota:

- *Antes de usar a directiva 'USING' deve-se comutar de banco através da programação do registo PSW.*



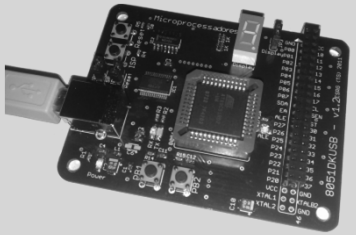
Stack Pointer Register

- ***Directiva 'USING': Exemplo completo***

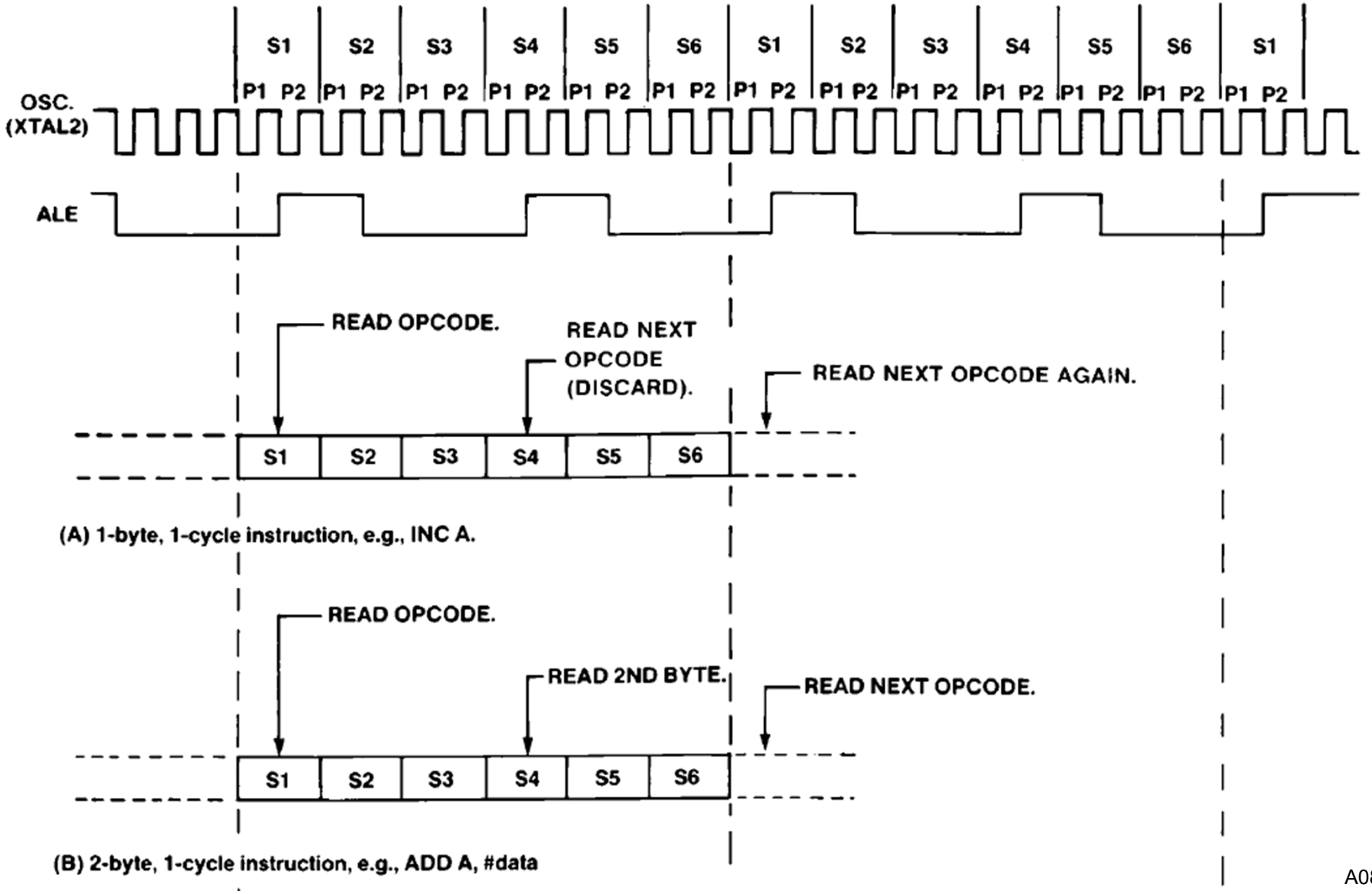
```
MOV PSW, #00011000B    ; Comuta para o banco 3
USING 3
PUSH AR0

MOV PSW, #00001000B    ; Comuta para o banco 1
USING 1
PUSH AR7
```

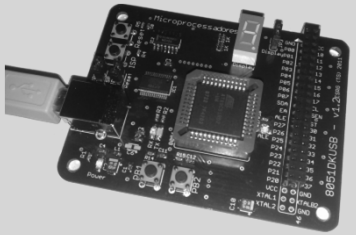
Nota:

- *Antes de usar a directiva 'using' deve-se comutar de banco através da programação do registo PSW;*
- *Em vez de se utilizar MOV PSW,#18H podíamos ter utilizado ORL PSW,#18H? (pós e contras?)*



Ciclo Máquina





Rotinas de atraso (delay)

```
DELAY1:  MOV     R2,#X
         DJNZ    R2,$
         RET

DELAY2:  MOV     R3,#Y
D2LOOP:  MOV     R2,#X
         DJNZ    R2,$
         DJNZ    R3,D2LOOP
         RET

DELAY3:  MOV     R4,#Z
D3L1:    MOV     R3,#Y
D3L2:    MOV     R2,#X
         DJNZ    R2,$
         DJNZ    R3,D3L2
         DJNZ    R4,D3L1
         RET
```

MOV Rn, #immediate

C	AC	F0	RS1	RS0	OV		P

Bytes 2

Cycles 1

Encoding 01111nnn | immediate

Operation
```
MOV
Rn = immediate
```

DJNZ Rn, offset

C	AC	F0	RS1	RS0	OV		P

Bytes 2

Cycles 2

Encoding 11011nnn | offset

Operation
```
DJNZ
PC = PC + 2
Rn = Rn - 1
IF Rn <> 0
  PC = PC + offset
```

RET

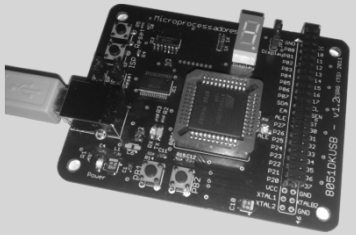
C	AC	F0	RS1	RS0	OV		P

Bytes 1

Cycles 2

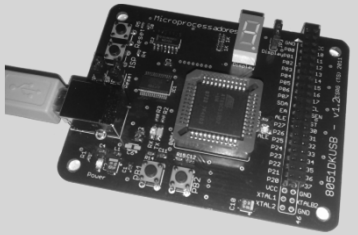
Encoding 00100010

Operation
```
RET
PC15-8 = (SP)
SP = SP - 1
PC7-0 = (SP)
SP = SP - 1
```



Rotinas de atraso (delay)

- ***Delay* por software**:
  - O tempo de espera é estabelecido pelo número de ciclos máquina necessários para executar a rotina de *delay*. O microprocessador fica bloqueado, ou seja, durante a execução da rotina *delay* não pode executar outro código;
- **Difíceis de controlar**:
  - Para além de ocuparem registos, o valor a colocar em cada registo não é “simples” de obter. Muitas vezes opta-se por implementar uma rotina de *delay* fixo (ex:1000µs) e invocá-la várias vezes;
- **Dependem**:
  - Do número de registos e dos seus valores, do nº de ciclos máquina necessários à execução das instruções e do cristal utilizado utilizado.



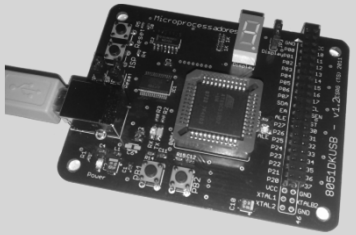
Rotinas de atraso (delay)

- Suponham que pretendemos gerar uma onda quadrada no pino P1.0. Qual a maior frequência possível e qual o *duty-cycle* dessa onda?

```
ONDA:   SETB    P1.1    ;NC=1⇒1µs
        CLR     P1.1    ;NC=1⇒1µs       T=4µs⇒f=250KHz
        SJMP    ONDA    ;NC=2⇒2µs       D=t_on/T*100=1µs/4µs=25%
```

- Altere o código de modo a garantir um *duty-cycle* de 50%. Qual a frequência da onda quadrada?

- Faça um rotina que permita gerar uma onda quadrada de 20KHz com um *duty-cycle* de 50%.



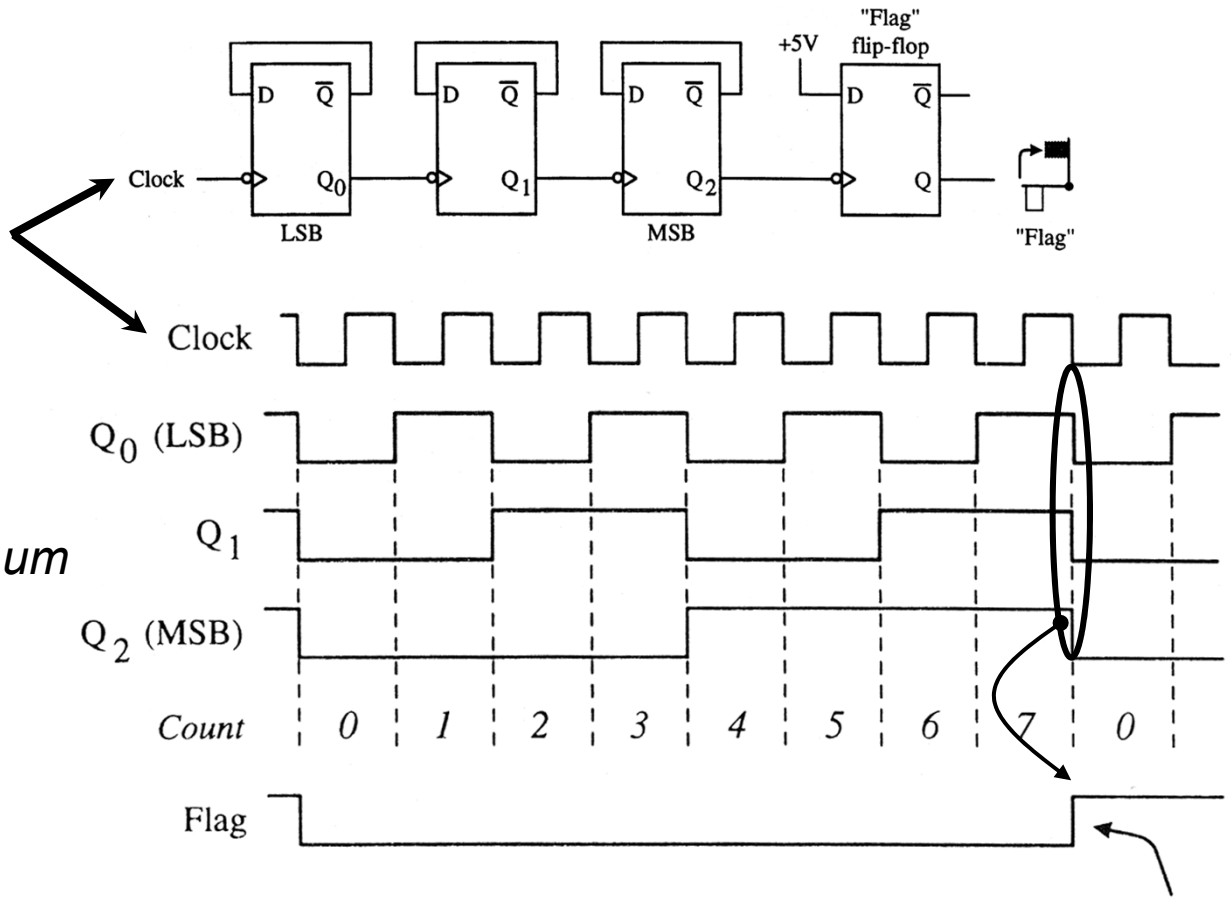
Timers/Counters

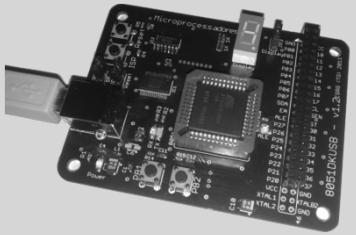
Como se divide um sinal de relógio?

O sinal de clock seria obtido através do relógio do microcontrolador – ***temporizador***

ou

através de um sinal de relógio externo ligado a um pino de E/S do microcontrolador - ***contador de eventos***

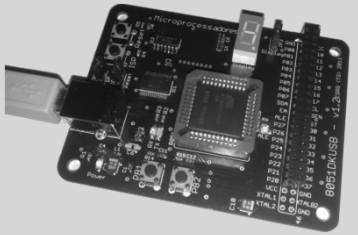




Timers/Counters

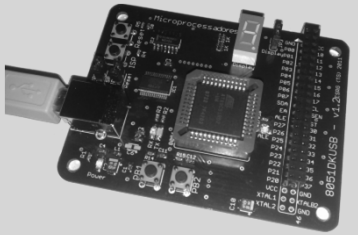
- *O 8051 tem duas unidades de temporização e contagem:*
  - *Timer* 0 e *Timer* 1;
  - Os modelos da família 8052 têm mais uma unidade: *Timer* 2.

- *Os timers podem operar como temporizadores ou como contadores de eventos externos ao microcontrolador:*
  - Quando opera como temporizador, os registos do *timer*, são incrementados a cada ciclo máquina (utiliza o relógio do CPU), ou seja a taxa de contagem é 1/12 da frequência do relógio;
  - No modo de operação de contagem de eventos externos, os registos são incrementados a cada transição de 1 para 0 na entrada externa do *timer*.



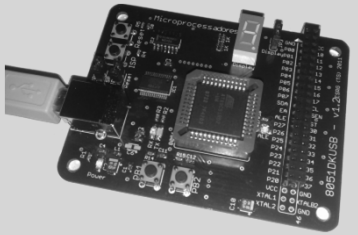
Timers/Counters

- *No modo de contagem, os registos do timer são incrementados sempre que há uma transição de 1-para-0 no respectivo pino de entrada (T0, T1 ou T2).*
  - O pino de entrada é amostrado durante o estado **S5P2** do ciclo de instrução.
  - O incremento é feito quando num ciclo de instrução a entrada estiver a ‘1’ e no ciclo seguinte estiver a ‘0’.
    - *O registo é actualizado durante o estado **S3P1** do estado a seguir a detecção.*
    - *Uma vez que são necessários dois ciclos de instrução, a maior taxa de contagem permitida é de 1/24 da frequência do relógio.*



Timers/Counters

- *Os timers 0 e 1 permitem quatro modos de funcionamento:*
  - *Modo 0: temporizador/contador de 13-bit.*
  - *Modo 1: temporizador/contador de 16-bit.*
  - *Modo 2: temporizador/contador de 8-bit com auto-reload.*
  - *Modo 3: duplo temporizador/contador de 8-bit.*

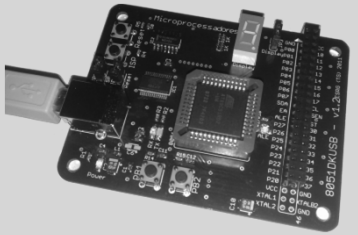


Timers/Counters

- *Registos usados na programação dos timers 0 e 1:*
  - *TMOD: Permite programar os modos de funcionamento dos timers.*
  - *TCON: Permite controlar a activação e verificar o estado dos timers.*
  - *THx/TLx: Registo de 16 bits (THx: MSB, TLx: LSB).*

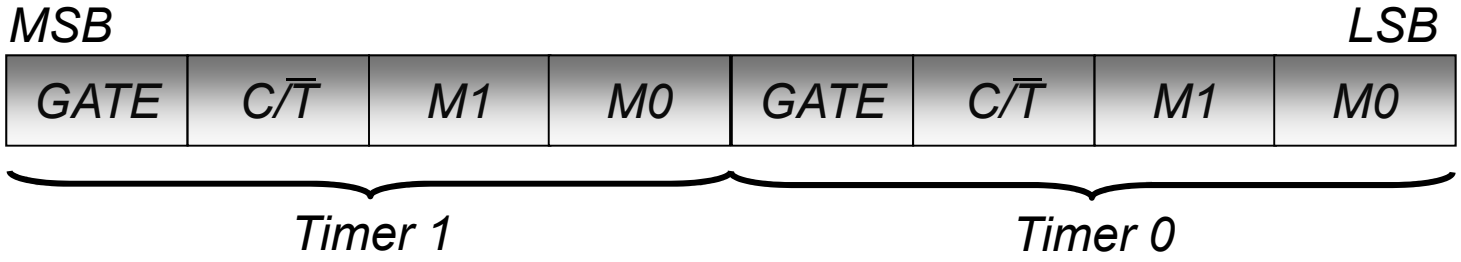
    Contêm os valores de contagem de eventos ou para a temporização de um intervalo de tempo
  - *[IE: Permite controlar as interrupções associadas aos timers]*

Endereços: (TH1:8Dh ; TH0:8Ch ; TL1:8Bh ; TL0:8Ah) ⇒ ***SFR***



Timers/Counters

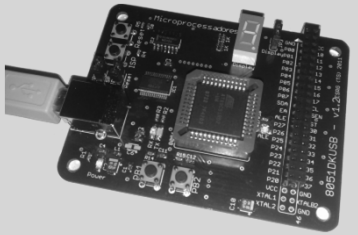
- *Registo:* ***TMOD*** (89h)



GATE: *Quando activado (‘1’), o timer x só é habilitado quando /INTx está a ‘1’ e o pino de controlo TRx está activado (‘1’). Quando desactivado, o timer x é habilitado quando TRx está activado (‘1’).*

C/$\overline{T}$: *Selecciona o modos de funcionamento que podem ser como temporizador ou como contador (contagem feita através do sinal no pino Tx).*

M0, M1: Selecciona os modos de operação.



Timers/Counters

- *Registo:* ***TMOD*** (89h)

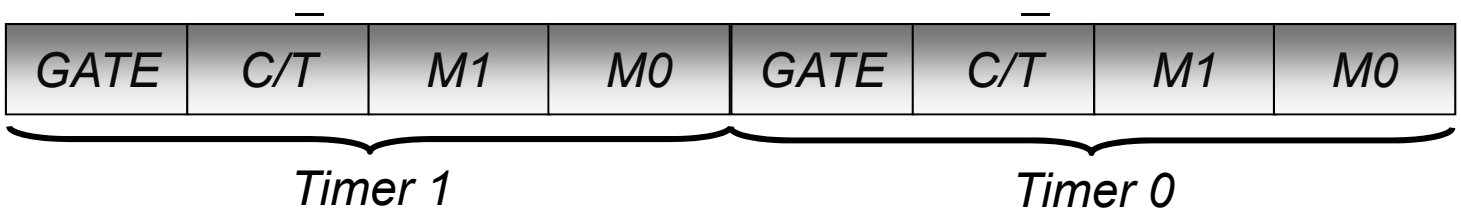
M1, M0: *0 0* *Modo 0: Temporizador/Contador X de 13-bit*

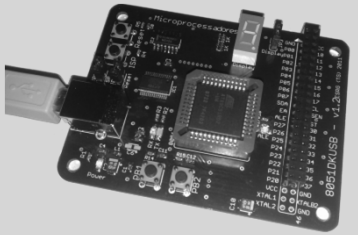
0 1 *Modo 1: Temporizador/Contador X de 16-bit*

1 0 *Modo 2: Temporizador/Contador X de 8-bit com auto-carregamento*

1 1 *Modo 3: Timer 0 – TL0 é um temporizador/contador de 8 bits controlado pelos bits de controlo do timer 0. TH0 é um temporizador/contador de 8-bit controlado pelos bits de controlo do timer 1.*

1 1 *Modo 3: Timer 1 – Está desabilitado (pode estar a fornecer o relógio à UART).*





Timers/Counters

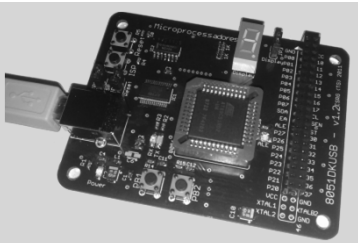
- *Registo:* ***TCON**** (88h)

MSB							*LSB*
TF1	*TR1*	*TF0*	*TR0*	*IE1*	*IT1*	*IE0*	*IT0*

TFx: *Timer overflow flag. É colocada a '1' pelo hardware da unidade, quando há overflow na contagem do timer. É limpa automaticamente pelo hardware, quando a rotina de serviço à interrupção é chamada. Caso não se use a interrupção, tem de ser limpa por software.*

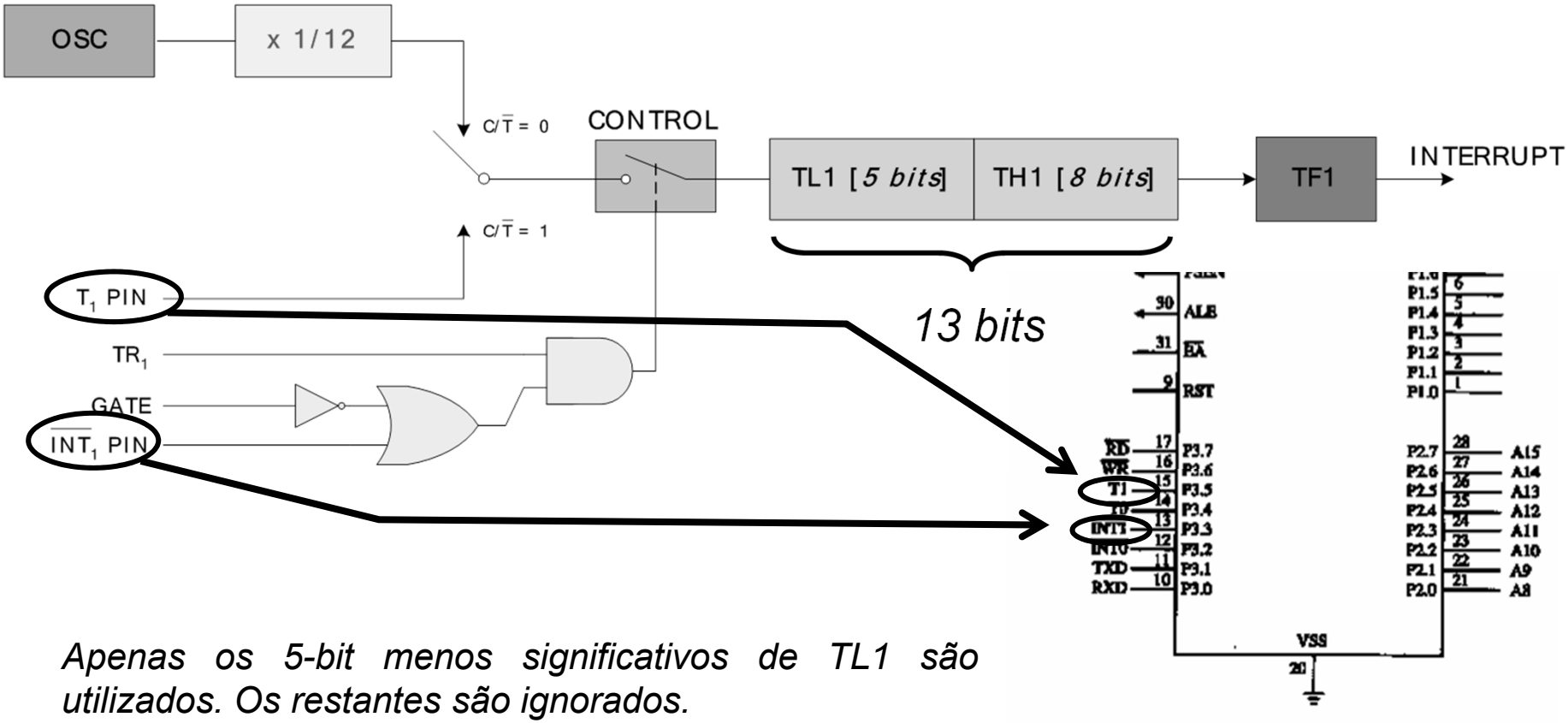
TRx: *Bit de controlo do timer x. Activado/limpo por software para habilitar/desabilitar o timer x.*

*[**IEx**, **ITx**: Gestão das interrupções externas 0 e 1]*

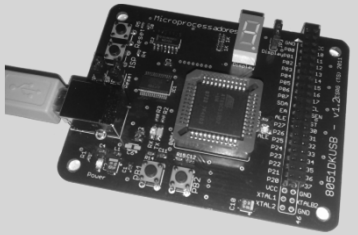


Timers/Counters

- *Timer: funcionamento no modo 0*

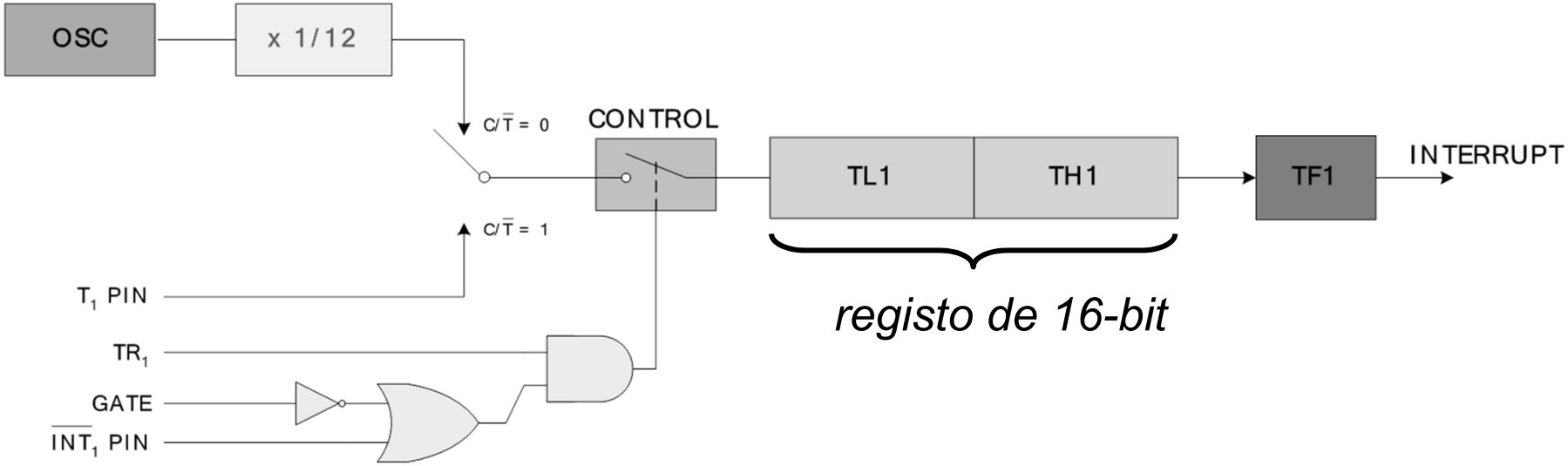


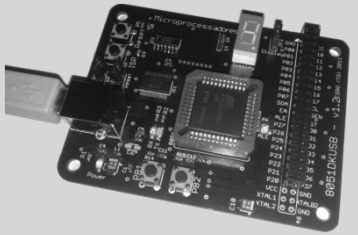
Apenas os 5-bit menos significativos de TL1 são utilizados. Os restantes são ignorados.
TL1 funciona como um pré-divisor por 32.



Timers/Counters

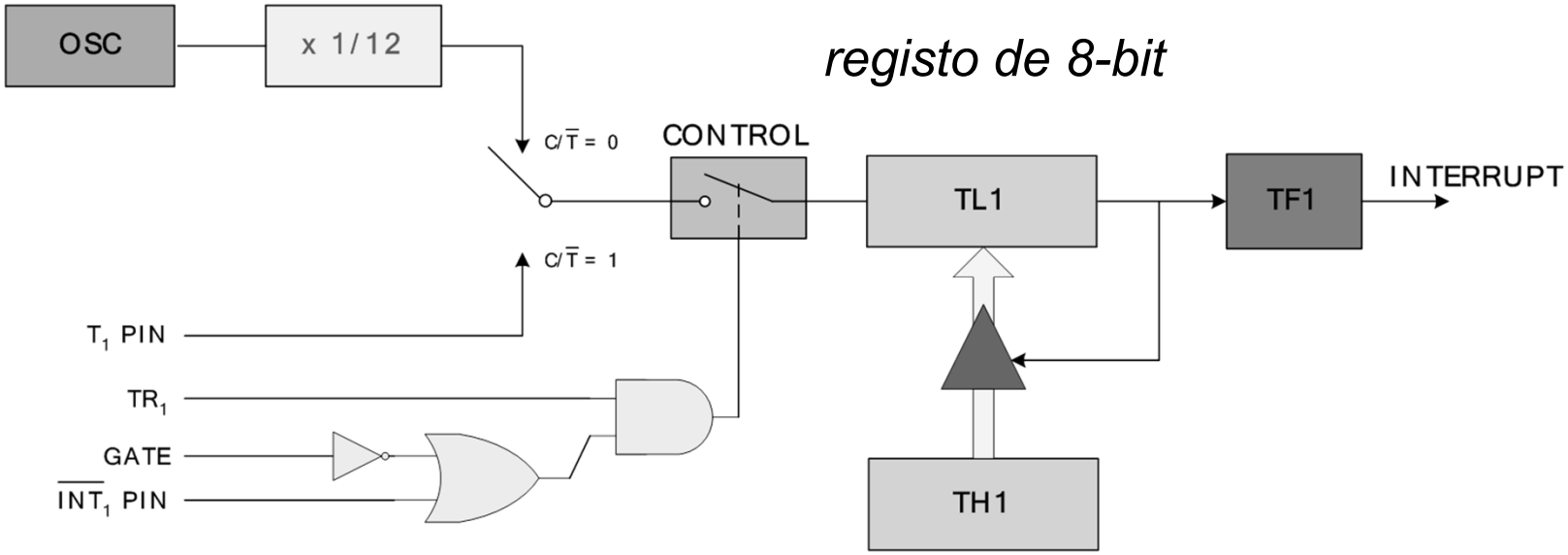
- *Timer: funcionamento no modo 1*



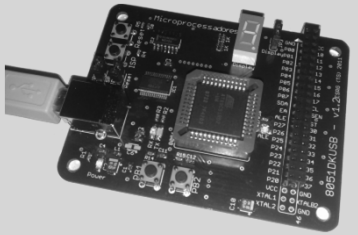


Timers/Counters

- *Timer: funcionamento no modo 2*

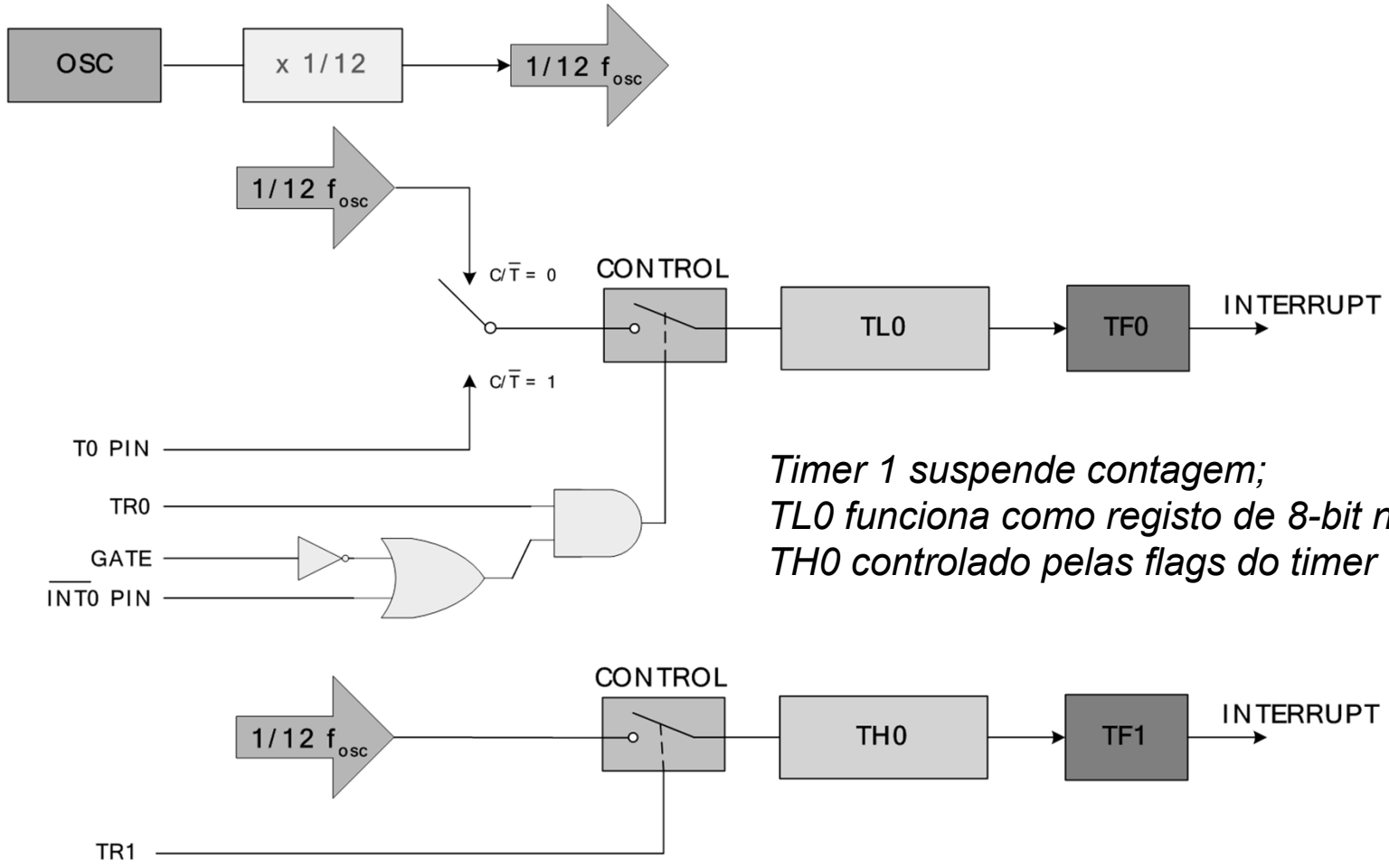


Sempre que ocorre o overflow do registo TLx, este é recarregado com o valor de THx.

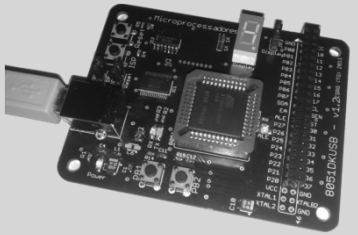


Timers/Counters

- *Timer: funcionamento no modo 3*

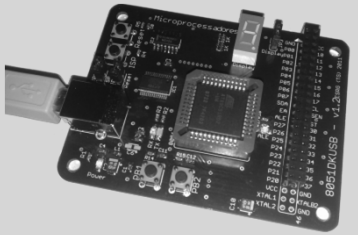


Timer 1 suspende contagem;
TL0 funciona como registo de 8-bit normal;
TH0 controlado pelas flags do timer 1.



Timers/Counters

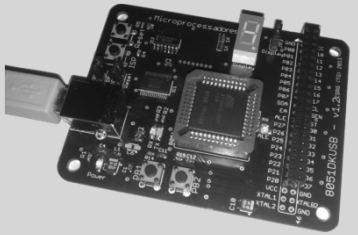
- *O timer 2 é um timer/counter de 16-bit (apenas na família 8052);*
- *Este permite três modos de funcionamento:*
  - ***Modo 0:*** *temporizador/contador de 16-bit com auto-carregamento;*
  - ***Modo 1:*** *modo de captura de 16-bit;*
  - ***Modo 2:*** *gerador de baud-rate para as comunicações série.*



Timers/Counters

- *Registos usados na programação do timer 2:*
  - *T2CON: Permite controlar a activação e verificar o estado dos timers – P89C51 ainda tem o T2MOD;*
  - *TH2/TL2: Registo de 16-bit (TH2: MSB, TL2: LSB);*
  - *RCAP2H/RCAP2L: Registo de 16-bit usado na captura e recarregamento;*
  - *[IE: Permite controlar a geração da interrupção associada aos timers]*

*(**T2CON***:0C8h;**T2MOD**:0C9h;**TH2**:0CDh;**TL2**:0CCh;**RCAP2H**:0CBh;**RCAP2L**:0CAh)* $\Rightarrow$ ***SFR***



Timers/Counters

- *Registo:* ***T2CON*** *(0C8h)

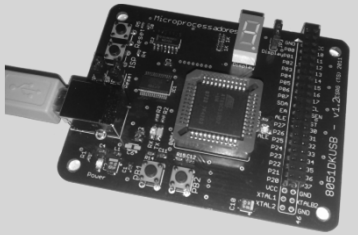
MSB							*LSB*
TF2	*EXF2*	*RCLK*	*TCLK*	*EXEN2*	*TR2*	*C/T2*	*CP/RL2*

TF2: *Quando activa indica o overflow do timer 2. Esta flag deve ser limpa por software;*

EXF2: *Activada quando a captura ou carregamento ocorre devido uma transição negativa de T2EX (pino externo) e EXEN2= 1. Se a interrupção estiver habilitada, ocorrerá uma interrupção e EXF2 terá de ser limpa por software;*

RCLK: *Quando activado indica que o relógio da comunicação série durante a recepção é proveniente do timer 2;*

TCLK: *Idem, mas para o relógio da transmissão.*



Timers/Counters

- *Registo:* ***T2CON**** (0C8h)

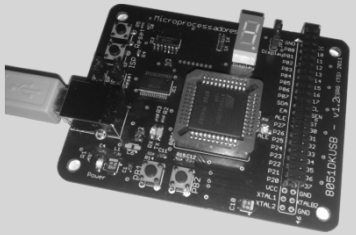
MSB							*LSB*
TF2	*EXF2*	*RCLK*	*TCLK*	*EXEN2*	*TR2*	*C/T2*	*CP/RL2*

EXEN2: Habilita a entrada externa. Quando activado permite que a captura ou o carregamento dependam de uma entrada externa

TR2: Arranque/paragem do timer 2

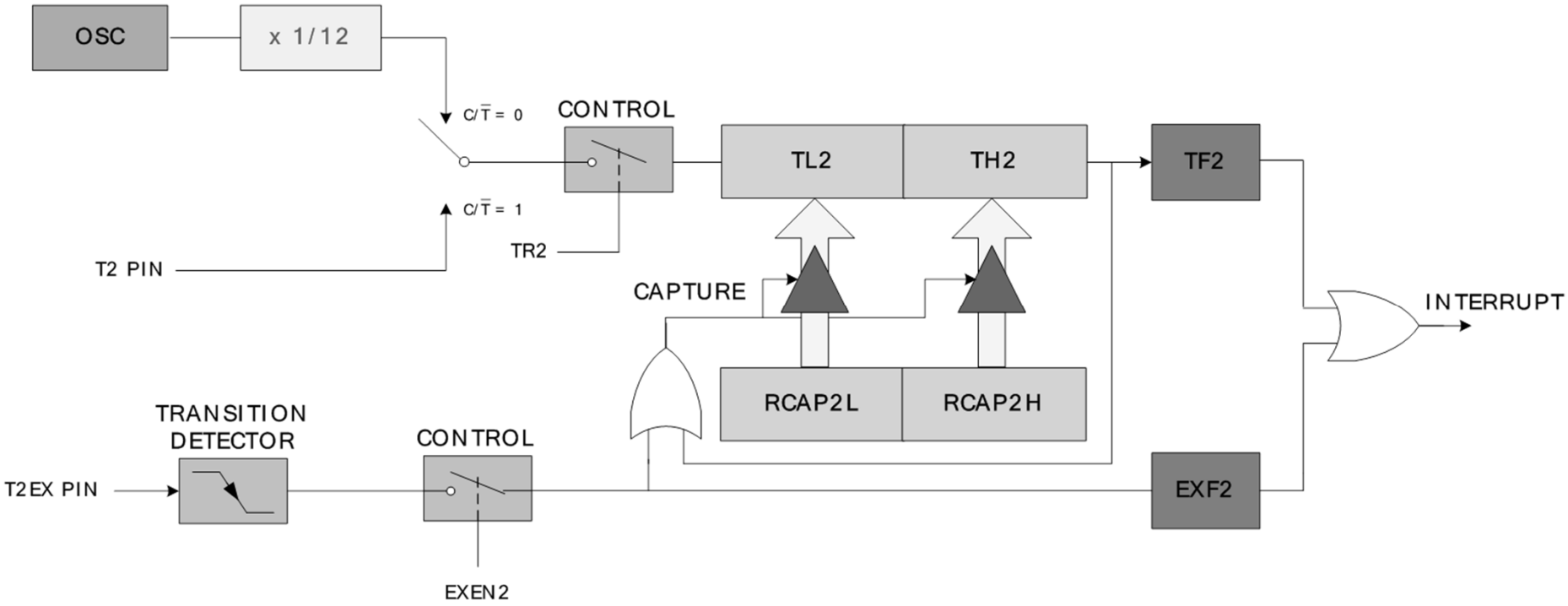
C/$\overline{T2}$: Selecção do modo temporizador (0) ou contador (1)

CP/$\overline{RL2}$: Selecção entre o modo de captura ou de carregamento



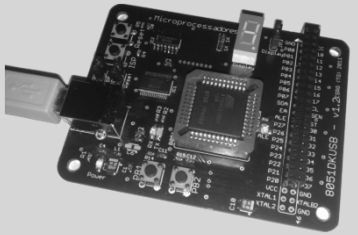
Timers/Counters

- *Timer 2: Modo 0 / Auto-reload*



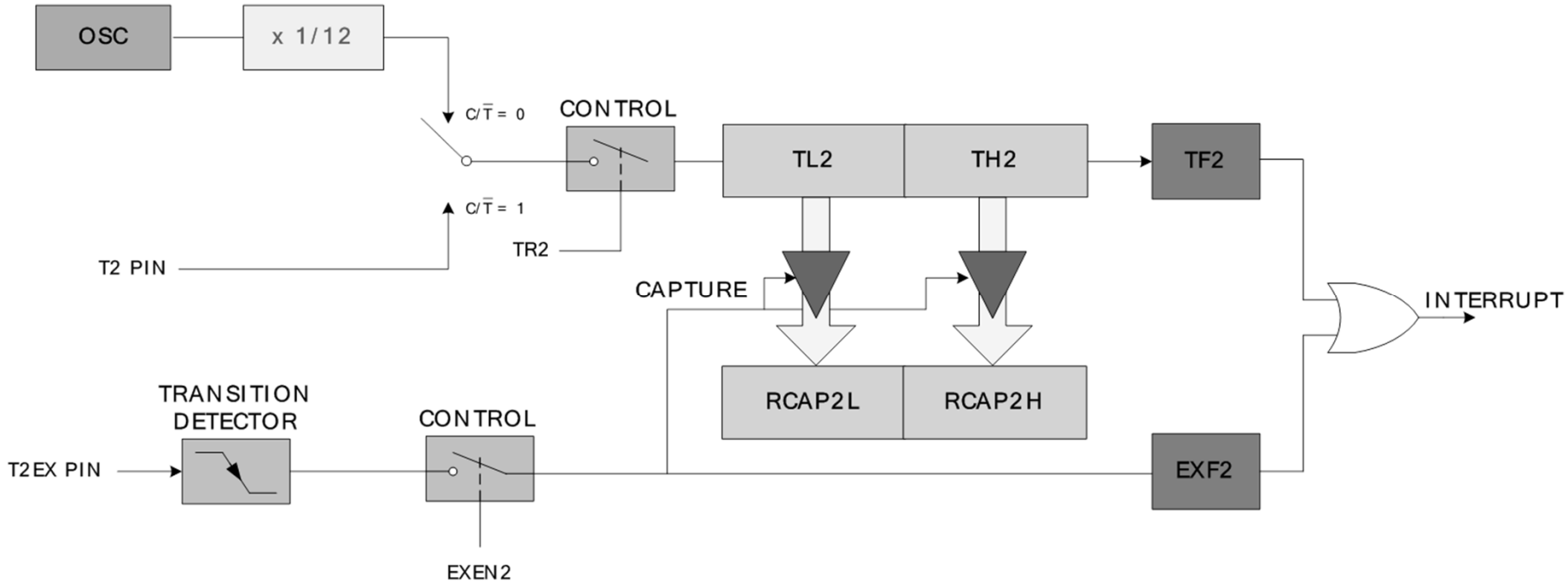
***CP/RL2=0**: timer de 16-bit com auto-reload de RCAP2L e RCAP2H. Colocando EXEN2 a 1 o reload também ocorre numa transição de 1-para-0 no pino T2EX.*

TF2 deve ser limpo por software antes de nova activação



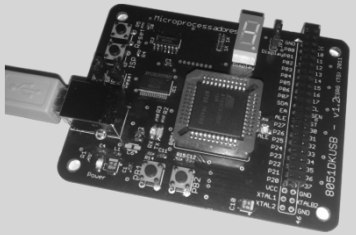
Timers/Counters

- *Timer 2: Modo 1 /Captura*



*Neste modo, **CP/RL2=1** e **EXEN2=1.***
Uma transição em T2EX captura o valor nos registos TH2 e TL2 para os registos RCAP2H e RCAP2L

TF2 sinaliza overflow do timer e deve ser limpo por software antes de nova activação



Timers/Counters

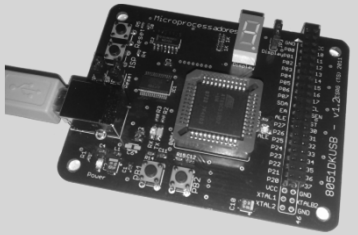
- *Exemplo*

  Escreva um programa que gere uma onda quadrada com 1 KHz de frequência no pino P1.0, usando o timer 0.

Análise:

1. *Repare que o período de uma onda quadrada de 1KHz é de 1000 μs, sendo que o tempo em alto é igual ao do tempo em baixo 500 μs;*
2. *Como este intervalo de 500 μs é superior a 256 μs torna-se impossível usar o timer no modo auto-reload porque neste modo funciona como temporizador de 8-bit;*
3. *Os timers contam no sentido crescente e activam a flag de overflow na transição FFFFh-para-0000h. Assim sendo, para um cristal de 12MHz, o timer tem de ser inicializado com -500=FE0Ch-até-0000h*
   i. *O valor a carregar em TL0/TH0 seria -500=FE0Ch, isto é: (TL0)=0Ch e (TH0)=0FEh*

      Timer0 configurado no modo 1 (temporizador de 16 bits) sendo o auto-reload efectuado após cada overflow pelo software a implementar



Timers/Counters

- *Exemplo*

Qual é o problema com esta solução?

```
        MOV TMOD, #01H      ; 16-bit timer mode.
LOOP:   MOV TH0, #0FEH      ; -500 (MSB).
        MOV TL0, #0C0H      ; -500 (LSB).
        SETB TR0            ; início da contagem
        JNB TF0, $          ; esperar pelo fim da contagem
        CLR TR0             ; parar a contagem --- não é necessário
        CLR TF0             ; limpar a flag de overflow --- obrigatório
        CPL P1.0            ; comutar o bit 0 do porto P1
        SJMP LOOP
        END
```

Haverá sempre um desvio à frequência pretendida devido ás instruções de re-inicialização do timer após cada overflow