

# **EEL7030 - Microprocessadores**

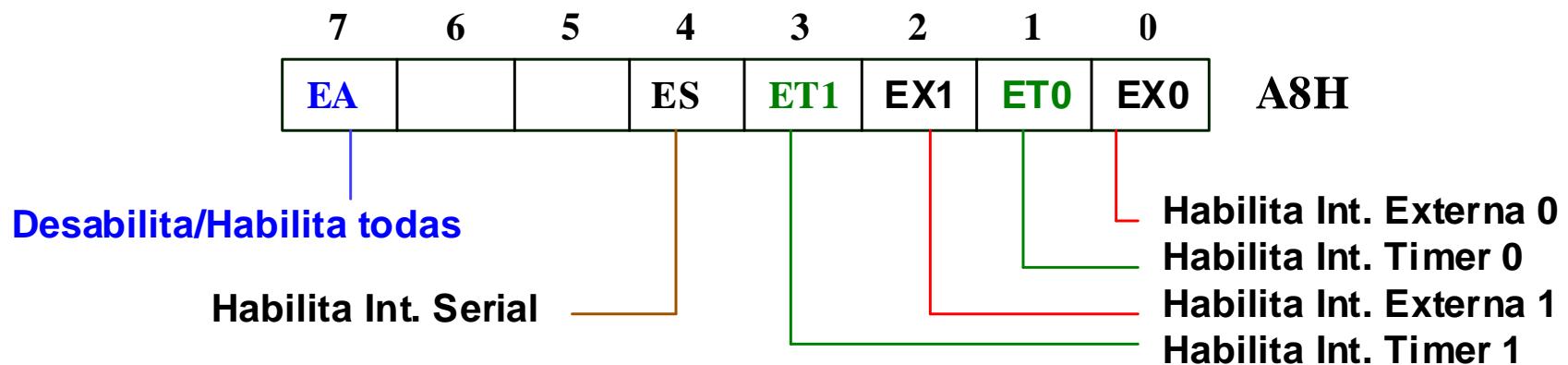


**Prof. Raimes Moraes**  
**GpqCom – EEL**  
**UFSC**

# Fontes de interrupção e endereços dos tratadores de interrupção do 8051

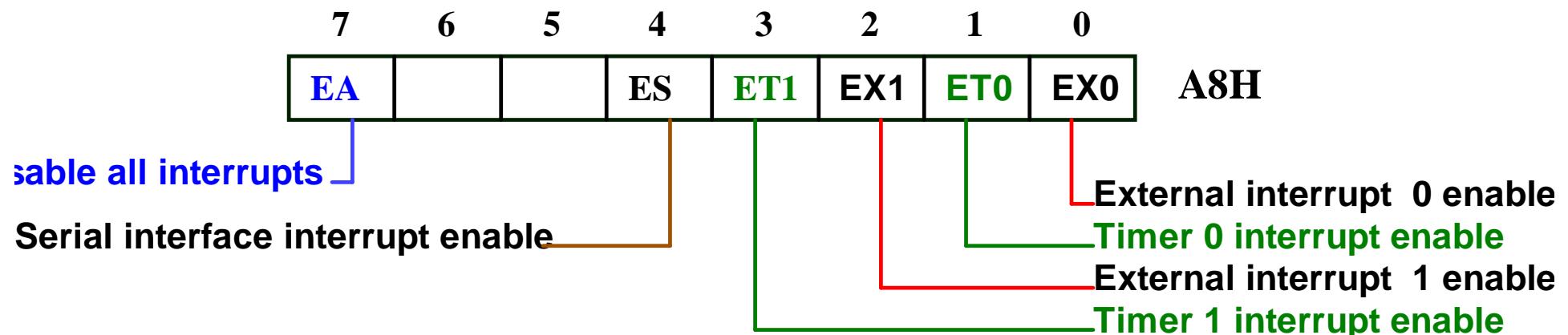
Interrupção	Endereço
Interrupção externa 0	0003H
Timer 0	000BH
Interrupção externa 1	0013H
Timer 1	001BH
Canal serial	0023H

## IE - Interrupt Enable Register - Bit Addressable



# Habilitação das Interrupções

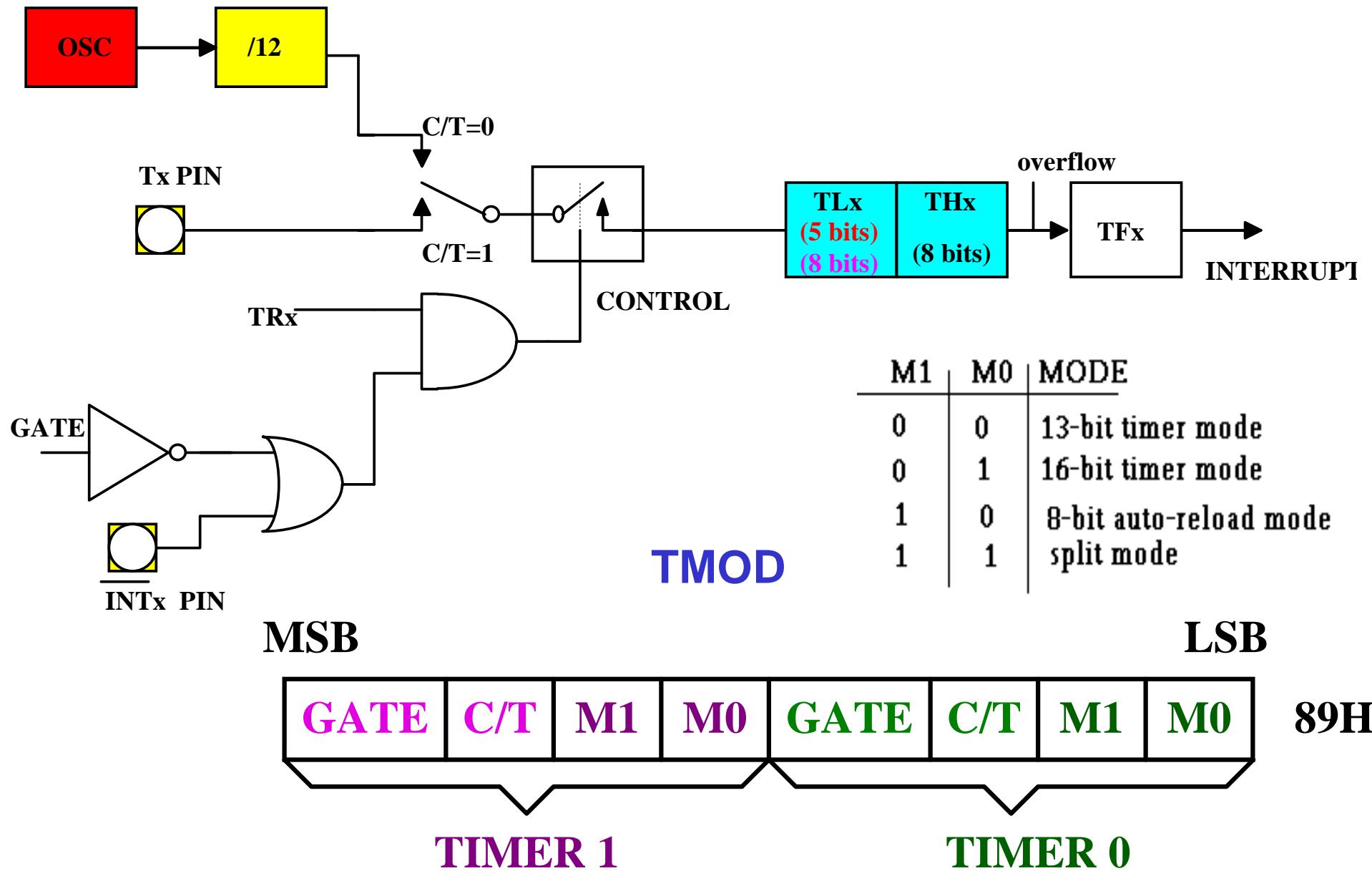
**IE - Interrupt Enable Register - Bit Addressable**



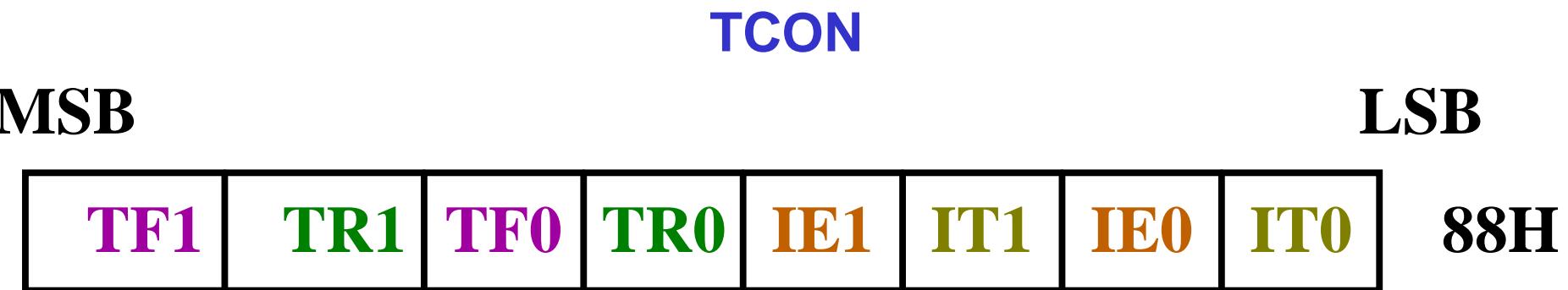
```
MOV IE,#10000101B; habilita INT0 E INT1
```

# Temporizadores / Contadores

## Modos 0 e 1



## Registrador Timer Control



**ITx - Interrupt control bit.** 1 => borda de descida  
0 => nível lógico baixo

### IEx - External Interrupt flag.

**Setado pelo hardware quando interrupção detectada.  
Apagada pelo software qdo salta para o tratador int.**

# Programação dos Timers

- 1. Alocar tratador para o contador/temporizador no seu devido endereço;**
- 2. Habilitar interrupção do contador/temporizador no registrador IE (EA, ETx);**
- 3. Especificar o modo de funcionamento do contador/temporizador (TMOD);**
- 4. Especificar intervalo de contagem (THx e TLx);**
- 5. Iniciar contagem (setb TRx);**
- 6. Ao término da contagem, realizar recarga nos modos: 0, 1 e 3.**

**OBS1:** Caso se deseje medir largura de pulso de sinal aplicado em /INT1, fazer GATE= '1'. A contagem ocorrerá durante intervalo de tempo no qual o pino /INT1 estiver em nível lógico alto.

**OBS2:** É possível testar, frequentemente, se houve encerramento da contagem sem necessidade de habilitar interrupções (*polling* dos flags TFx) em programas mais dedicados.

Exemplo: Faça um programa que escreva na porta P1, a cadeia de 16 caracteres: 'Microcontrolador' à taxa de 10 kiB/s ( (1/10000) = 100 us)

```
reset    equ 0h
Itmr0    equ 0bh ; local tratador
state    equ 20h
```

```
org reset      ;PC=0 depois de reset
jmp inicio
```

```
org Itmr0
jmp handler
```

inicio:

```
mov ie,#10000010b ; habilita
mov tmr0,#01h       ; modo 1
```

```
mov th0,#0ffh
mov tl0,#09ch
```

```
mov state,#0h ;inicialização
mov r0,# state
mov dptr,#tabela
mov r1,#0
```

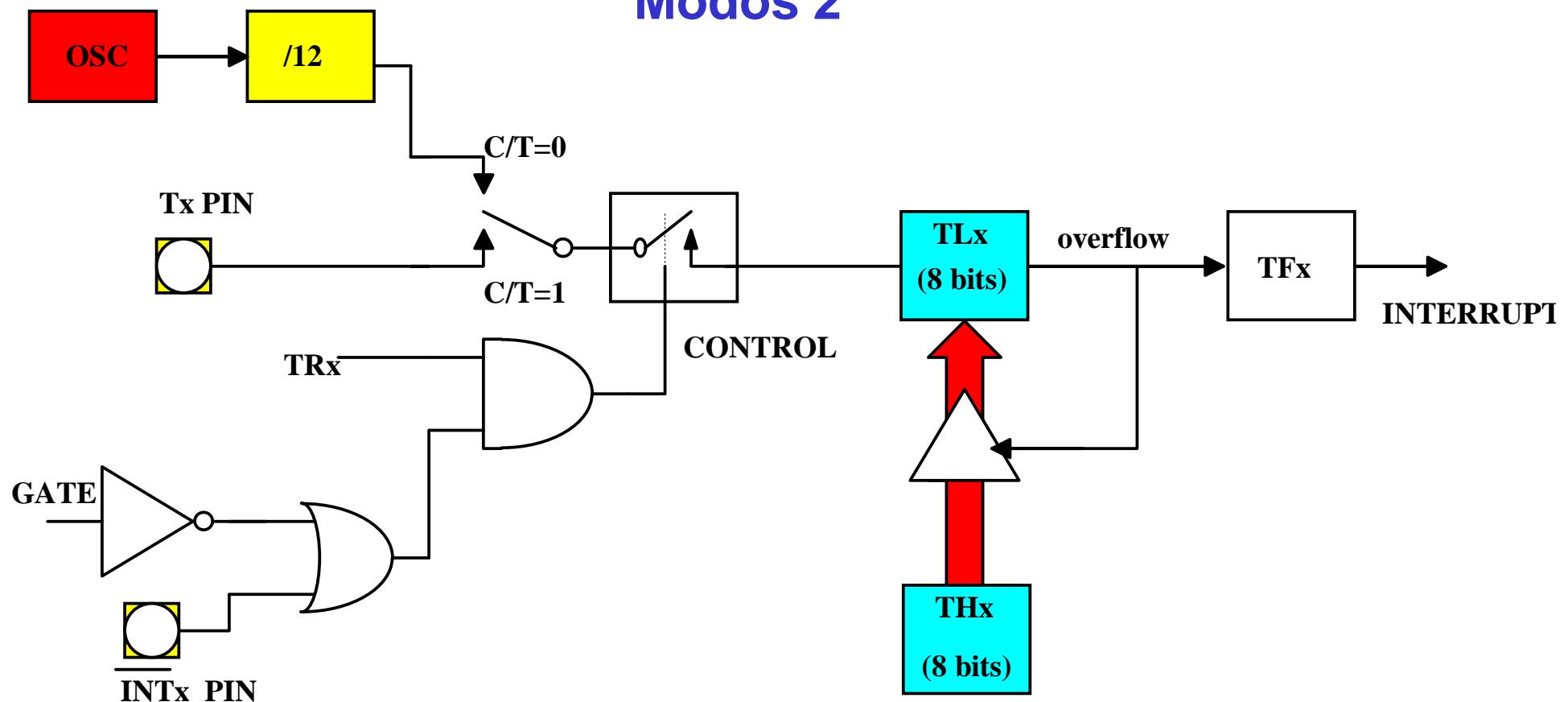
```
setb tr0
```

```
volta:   cjne    @r0,#1,volta
          mov     th0,#0ffh
          mov     tl0,#09ch
          mov     state,#0h
          mov     a,r1
          movc   a,@a+dptr
          mov     p1,a
          inc    r1
          cjne   r1,#16,volta
          clr    tr0
          jmp    $
```

```
handler:  mov     state,#1h
           reti
```

```
tabela: db 'Microcontrolador'
         end
```

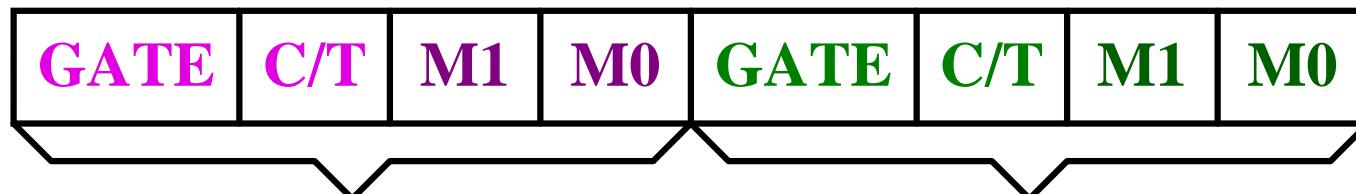
## Temporizadores / Contadores Modos 2



MSB

**TMOD**

LSB



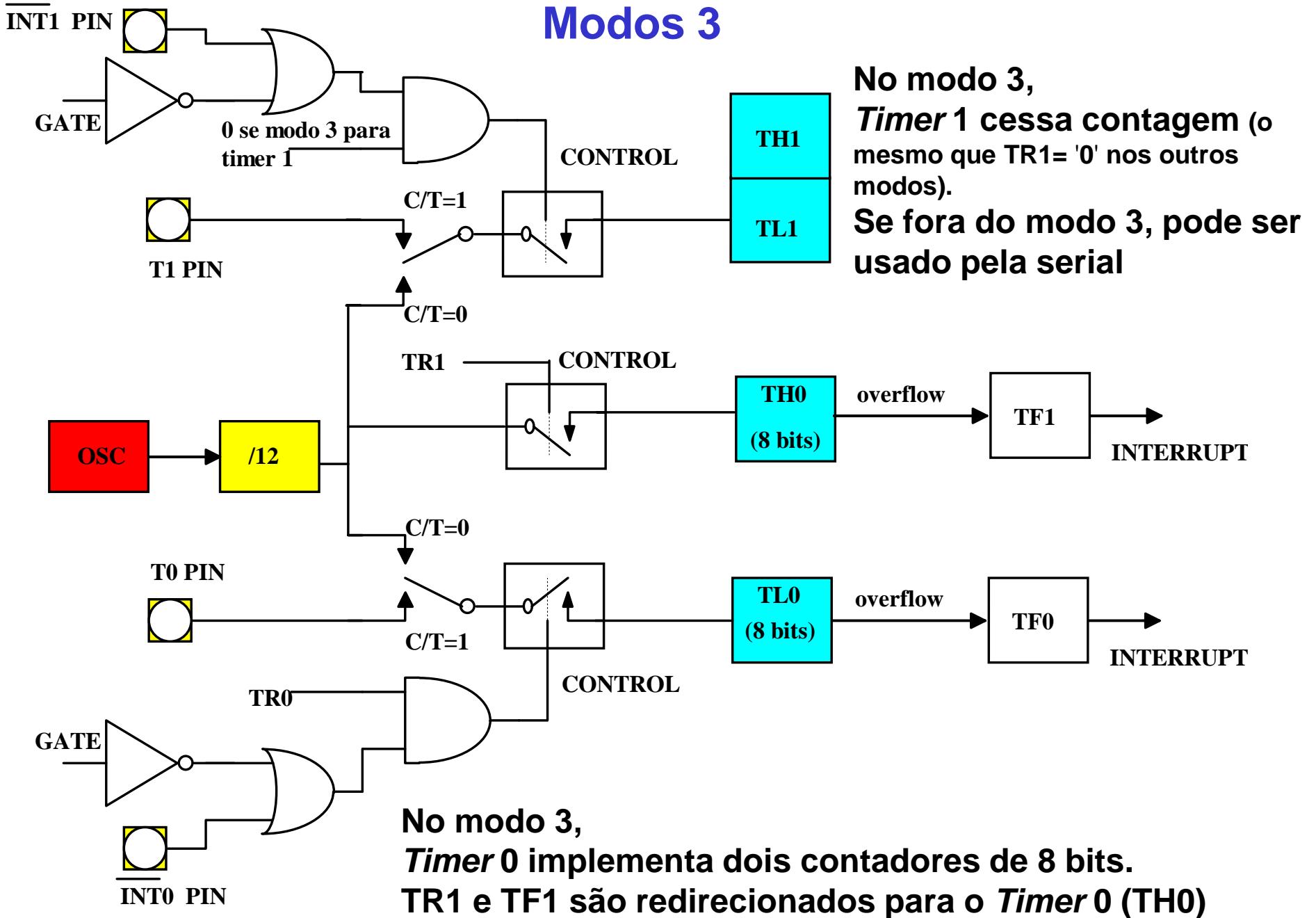
**TIMER 1**

**TIMER 0**

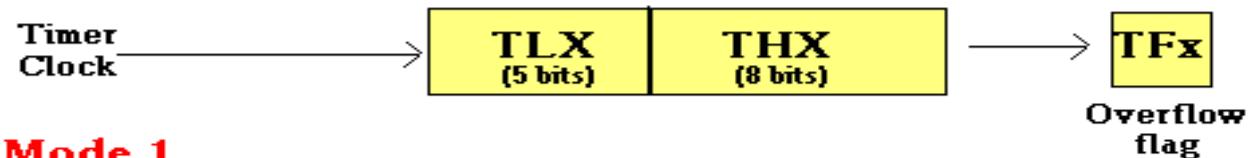
89H

# Temporizadores / Contadores

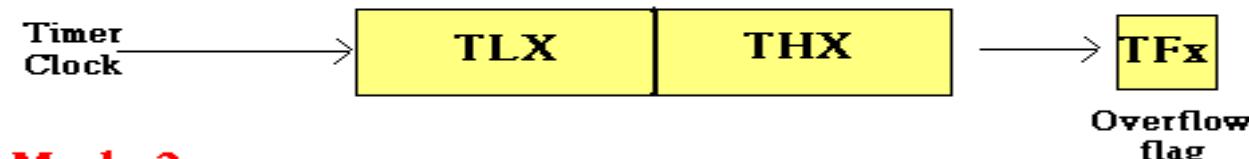
## Modos 3



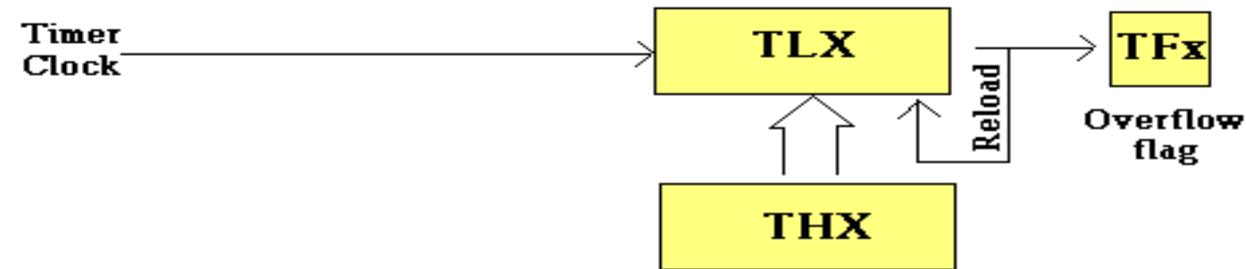
### Mode 0



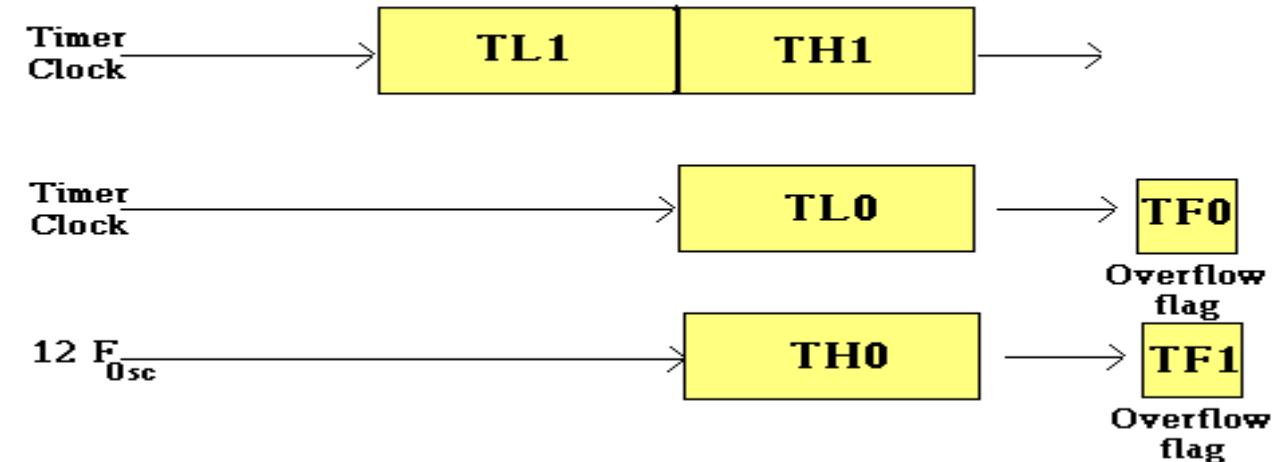
### Mode 1



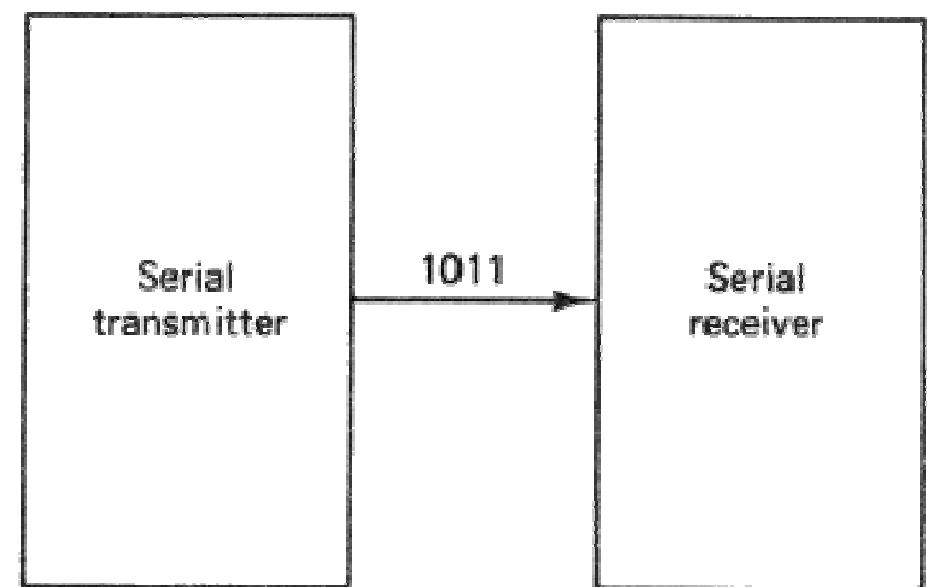
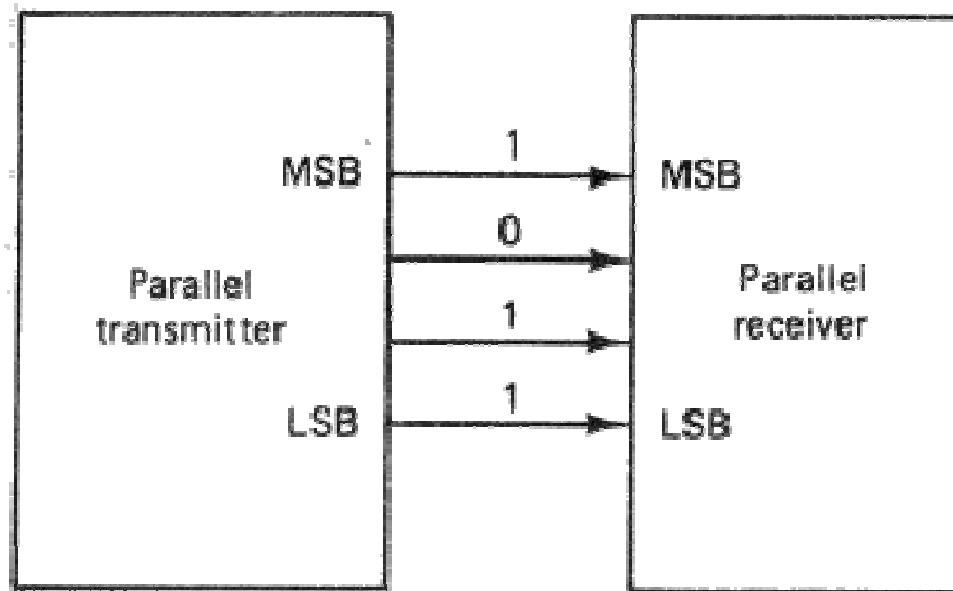
### Mode 2



### Mode 3



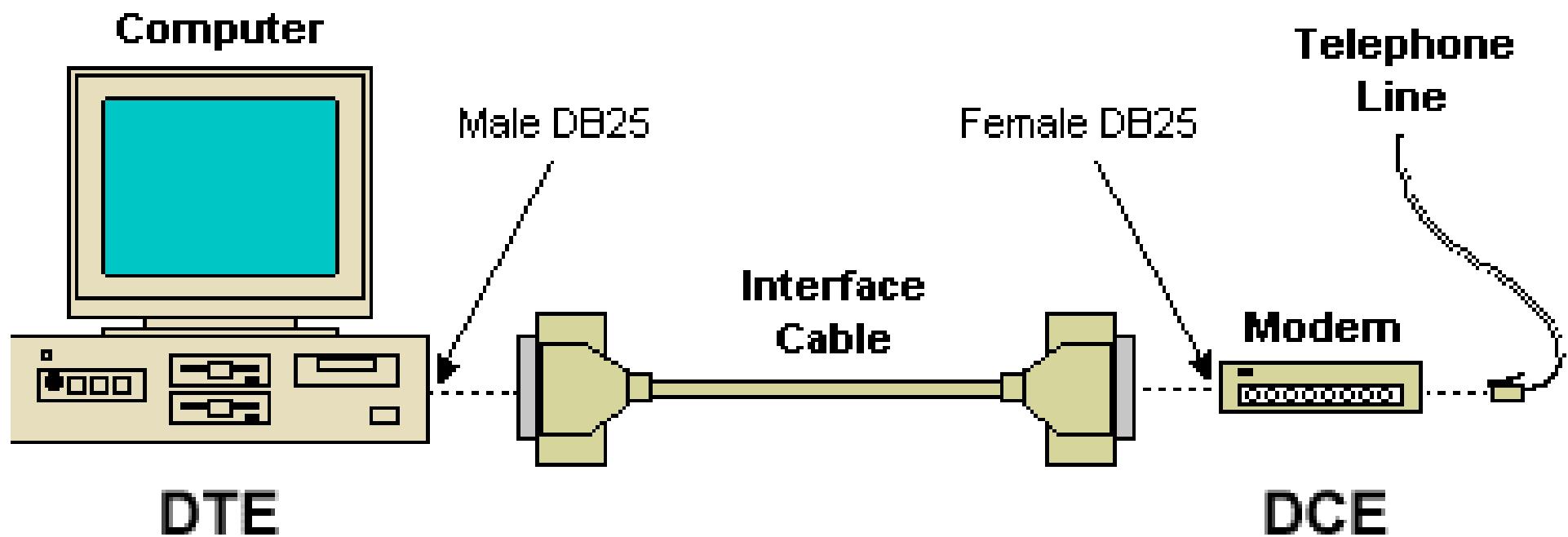
# Transmissão Serial



# Transmissão Serial

**USART:** *Universal Synchronous Asynchronous Receiver Transmitter*

**EIA RS-232C:** padrão industrial para a comunicação serial de dados binários entre um **DTE** (terminal de dados) e um **DCE** (comunicador de dados).

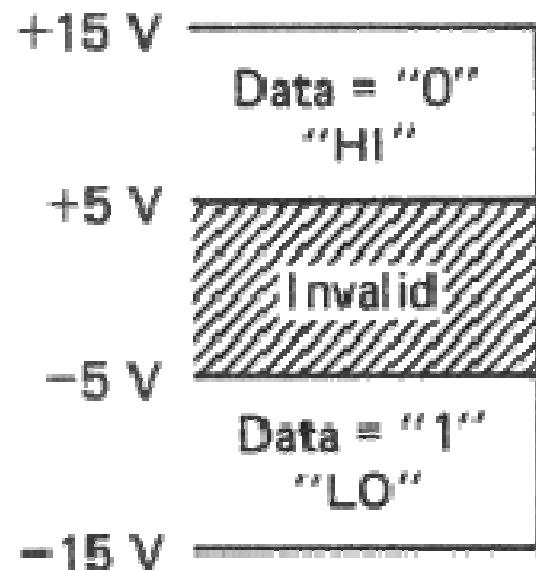


Foi largamente utilizado em PCs, estando ainda presente em muitos equipamentos.

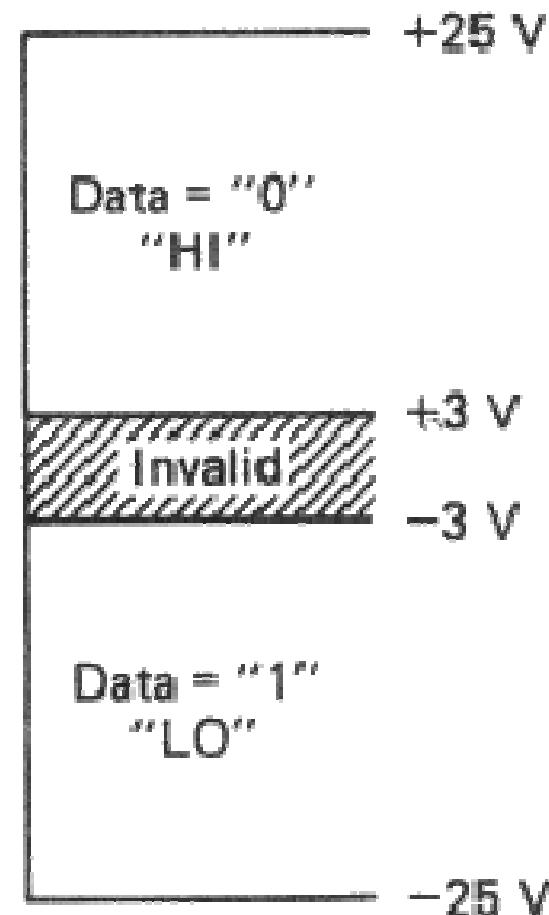
# Níveis Lógicos EIA RS-232C

(Electronics Industries Association)

- Tamanho máximo do cabo: 15 metros
- Velocidade de transmissão: 1Mbits/s

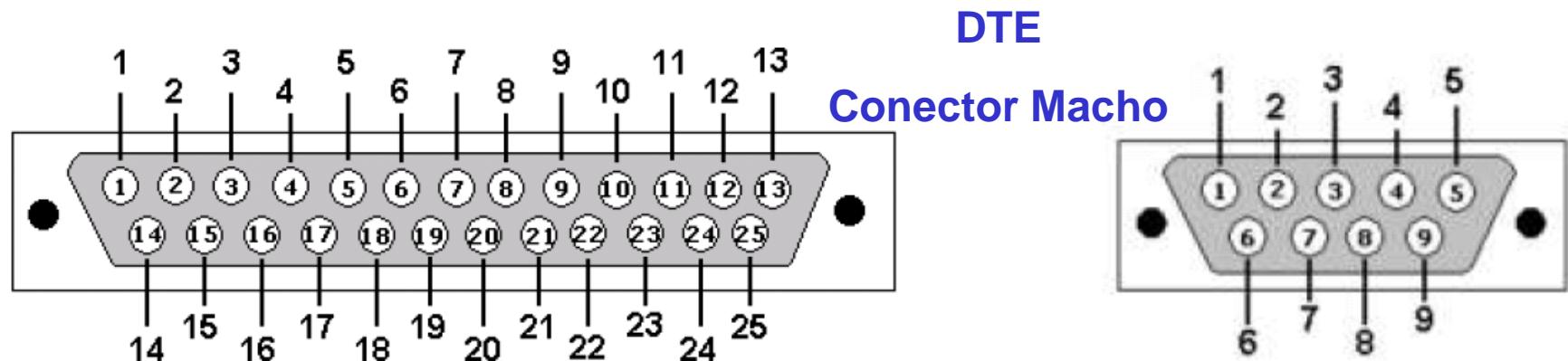


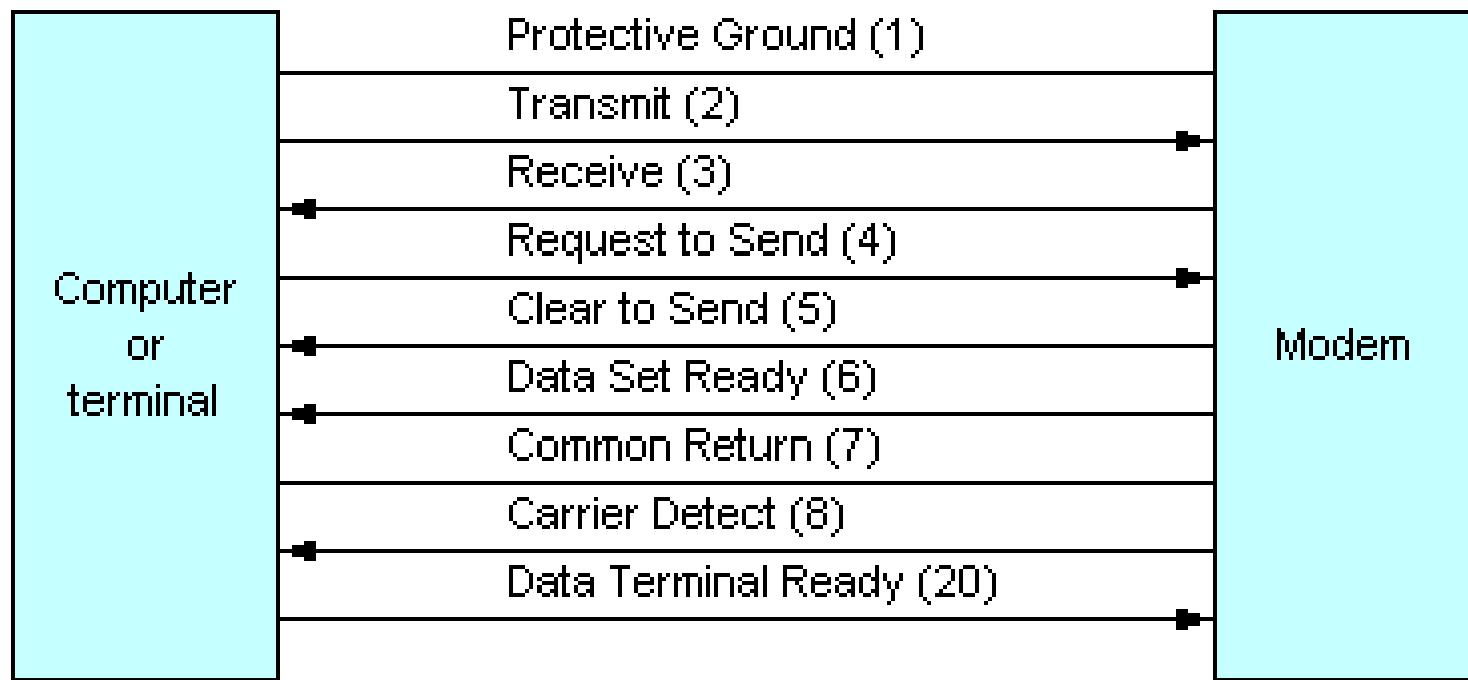
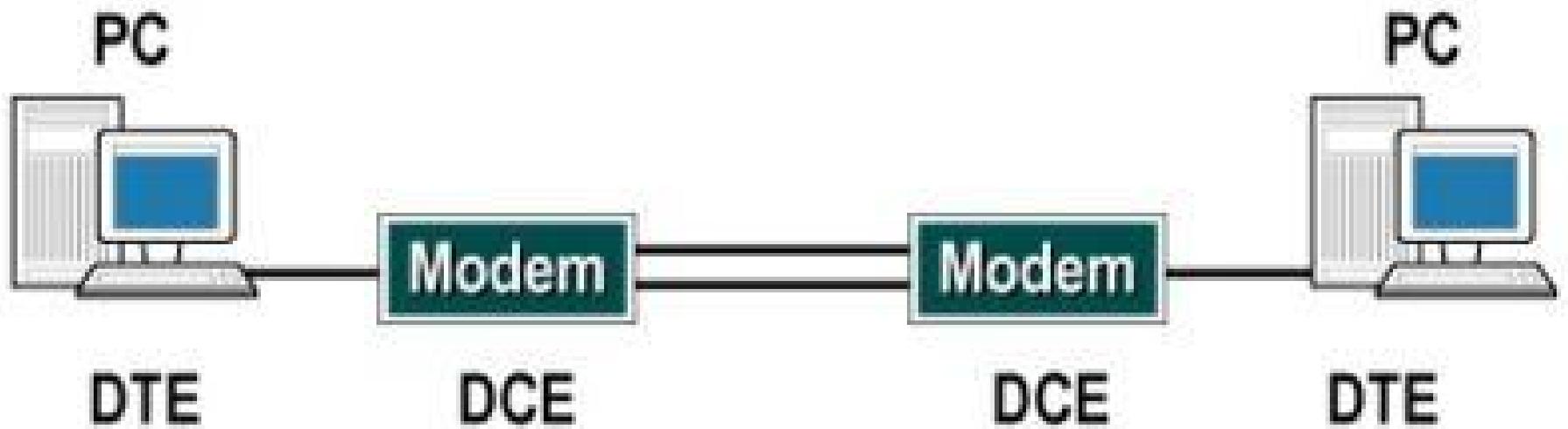
Driver



Receiver

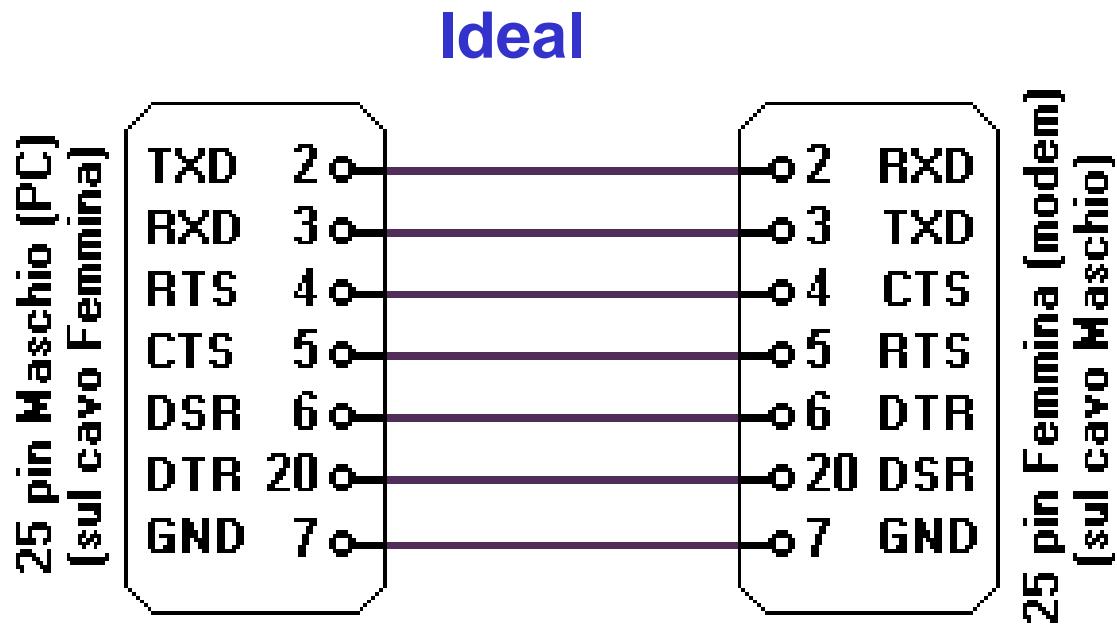
Pin				
DB25	DB9	Signal Name		Direction
1		CD	Chassis Ground	-
2	2	TD	Transmit Data	DTE → DCE
3	3	RD	Receive Data	DTE ← DCE
4	7	RTS	Request To Send	DTE → DCE
5	8	CTS	Clear To Send	DTE ← DCE
6	6	DSR	Data Set Ready	DTE ← DCE
7	5	SG	Signal Ground	-
8	1	DCD	Data Carrier Detect	DTE ← DCE
20	4	DTR	Data Terminal Ready	DTE → DCE
22	9	RI	Ring Indicator	DTE ← DCE



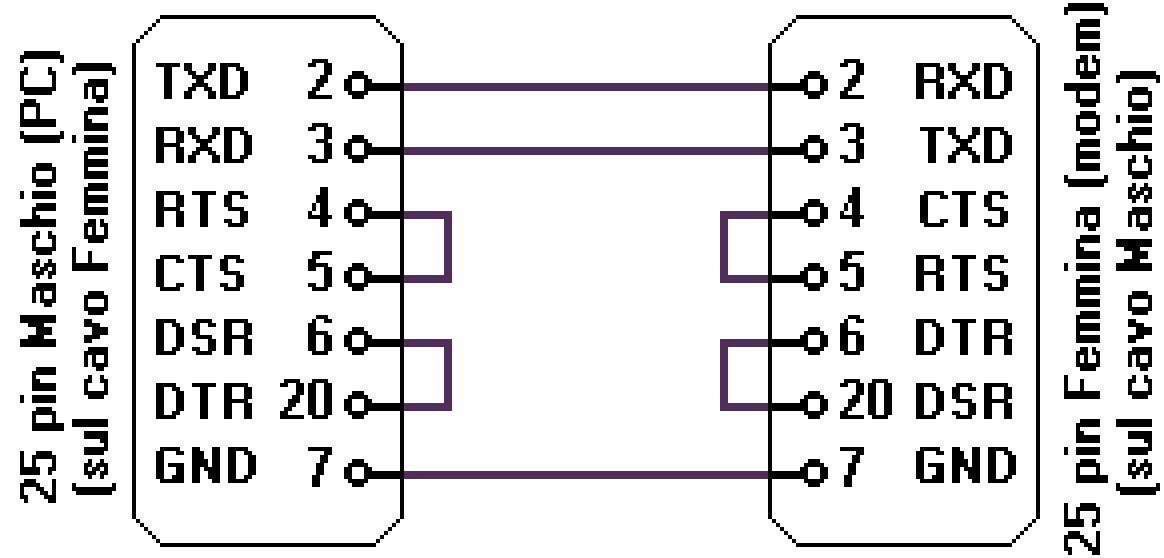


DCD indica quando um modem está conectado a outro modem remoto via linha telefônica

# Conexões



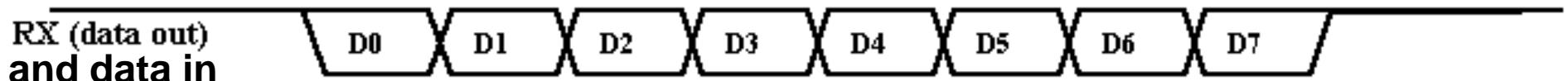
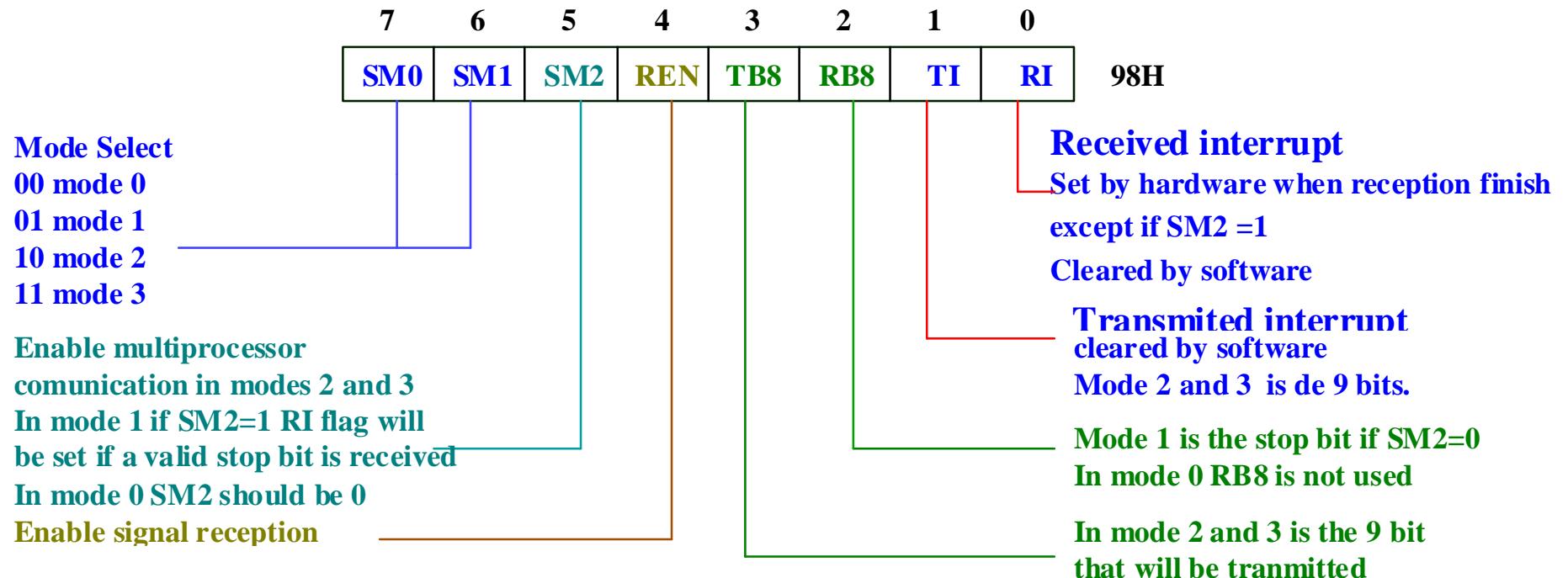
**Mínimo**



# **Handshaking**

- Troca de sinais para estabelecer comunicação condicional
- Sinais DTR e DSR permite que os dispositivos identifiquem que estão conectados entre si.
- Processo
  - Transmissor ativa RTS
  - Receptor detecta CTS por interrupção ou *polling*
  - Receptor ativa RTS
  - Transmissor aguarda CTS
  - Transmissor envia dados

## SCON - Serial Port Control Register - Bit Addressable

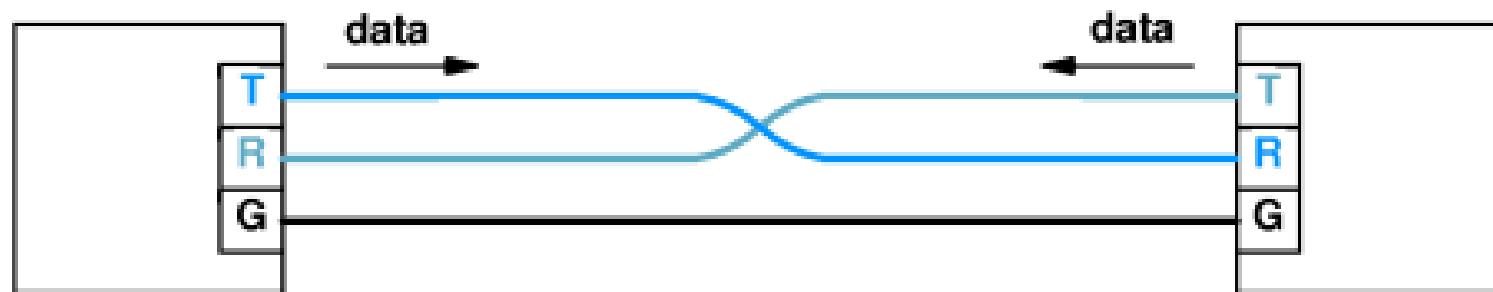


# Modos de Programação

SM0	SM1	MODO	PROTOCOLO	BITS	TAXA
0	0	0	SÍNCRONO - HD	8	Fclock/12
0	1	1	ASSÍNCRONO- FD	10	Variável
1	0	2	ASSÍNCRONO - FD	11	Fclock/32 ou /64
1	1	3	ASSÍNCRONO - FD	11	Variável

# *Full Duplex*

**Full duplex (FD):** dados são (ou podem ser) transmitidos e recebidos simultaneamente por ambos os dispositivos conectados, requerendo para tal, conexões adequadas



## Programação *Baud Rate*

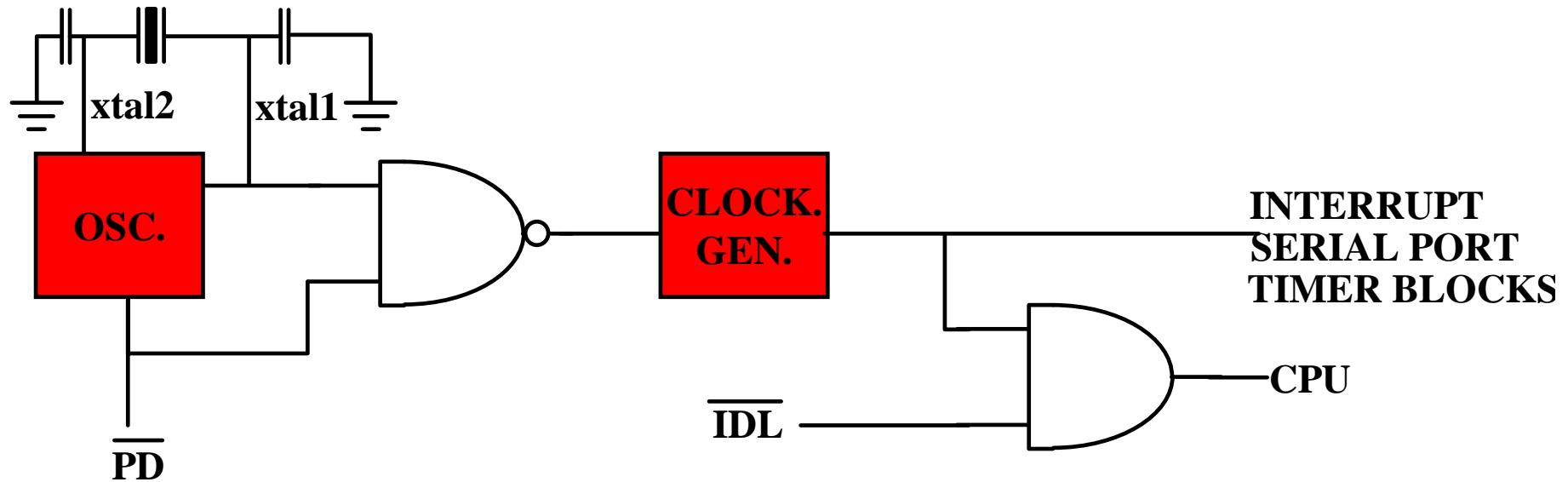
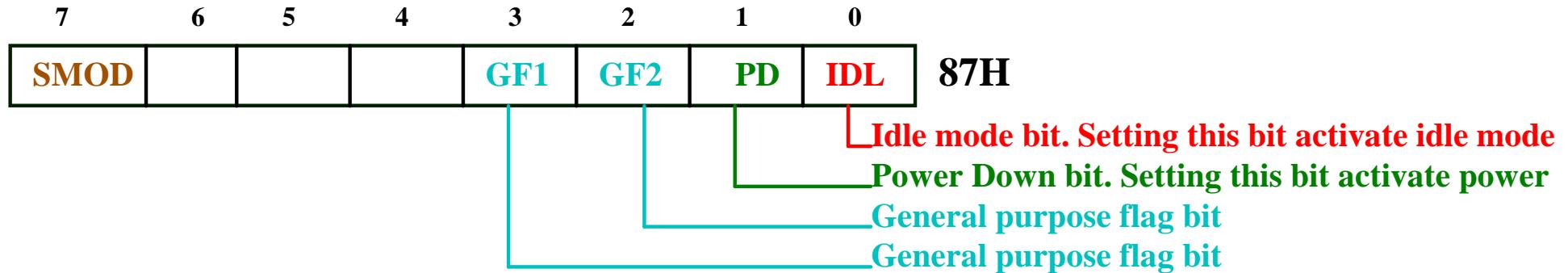
Modo 2 =>  $BR = [(2^S\text{MOD}) / 64] \times f_{osc}$

Modo 1 e 3 => Timer 1 é utilizado para estabelecer a *baud rate*, de acordo com:

$$BR = [(2^S\text{MOD}) / 32] \times \{f_{osc} / [12 \times (256 - TH1)]\}$$

<b>Baud Rate</b>	<b>Fosc</b>	<b>SMOD</b>	<b>----- Timer 1 -----</b>		
			<b>C/T</b>	<b>Mode</b>	<b>Reload Value</b>
<b>Mode 0 Max: 1 MHz</b>	<b>12 MHz</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>
<b>Mode 2 Max: 375K</b>	<b>12 MHz</b>	<b>1</b>	<b>X</b>	<b>X</b>	<b>X</b>
<b>Modes 1, 3: 62.5K</b>	<b>12 MHz</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>FFH</b>
<b>19.2K</b>	<b>11.059 MHz</b>	<b>1</b>	<b>0</b>	<b>2</b>	<b>FDH</b>
<b>9.6K</b>	<b>11.059 MHz</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>FDH</b>
<b>4.8K</b>	<b>11.059 MHz</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>FAH</b>
<b>2.4K</b>	<b>11.059 MHz</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>F4H</b>
<b>1.2K</b>	<b>11.059 MHz</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>E8H</b>
<b>137.5K</b>	<b>11.986 MHz</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>1DH</b>
<b>110K</b>	<b>6 MHz</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>72H</b>
<b>110K</b>	<b>12 MHz</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>FEEBH</b>

## PCON - Power Control Register



Para sair de Idle: por uma interrupção que esteja habilitada ou por Reset.

Por interrupção: vai para o tratador e após RETI , continua de onde parou. Por Reset: continua de onde parou.

Para sair de Power Down - Somente por Reset (não altera RAM).

# Programação da Interface Serial: Usando Interrupção

- 1) Alocar tratador de interrupção da serial no endereço 23H;
- 1) Habilitar interrupção serial (ES, EA);
- 2) Especificar o modo de funcionamento da serial (SCON). Habilitar recepção quando for o caso;
- 3) Especificar taxa de transmissão/recepção de dados no Timer 1 (TMOD, TH1 e SMOD)
- 4) Disparar contagem (setb TR1);
- 5) Carregar dado em SBUFF para iniciar transmissão. Carregar ou ler dado de SBUFF quando da ocorrência da interrupção.
- 6) Ao saltar para o tratador de interrupção serial, limpar flag que a solicitou.

OBS: É possível testar, freqüentemente, se os flags RI e TI estão setados (indicando buffer de recepção cheio ou buffer de transmissão vazio, respectivamente) sem a necessidade de habilitar a interrupções (polling dos flags) em programas mais dedicados.

Faça um programa que transmita, serialmente, a cadeia de 16 caracteres: 'Microcontrolador'. Realize a transmissão de forma assíncrona sem envio de bit de paridade à taxa de 9600 bauds (kibi/s).

```
RESET    EQU    00H
LTSERIAL EQU    23H ; local tratador
STATE     EQU    20H

ORG RESET  ;PC=0 depois de reset
JMP INICIO

ORG LT SERIAL
CLR TI
MOV STATE,#1H
RETI

INICIO: MOV IE,#10010000B
        MOV SCON,#01000000B
        MOV TMOD,#00100000B
        MOV TH1,#0FDH
        MOV TL1,#0FDH
        MOV PCON,#0H
        SETB TR1
```

```
MOV STATE,#0H
MOV R0,# STATE
MOV DPTR,#TABELA
MOV R1,#1
MOV SBUF,#'M'

VOLTA: CJNE @R0,#1,VOLTA
       MOV STATE,#0H
       MOV A,R1
       MOVC A,@A+DPTR
       MOV SBUFA
       INC R1
       CJNE R1,#16,VOLTA
       CLR TR1
       JMP $

TABELA: DB 'Microcontrolador'
        END
```

# Programação da Interface Serial: Usando *Polling*

Faça um programa que transmita, serialmente, a cadeia de 16 caracteres: 'Microcontrolador'. Realize a transmissão de forma assíncrona sem envio de bit de paridade à taxa de 9600 bauds (kibi/s).

```
RESET EQU 00H
```

```
ORG RESET ;PC=0 depois de reset
JMP INICIO
```

```
INICIO: MOV SCON,#01000000B
        MOV TMOD,#00100000B
        MOV TH1,#0FDH
        MOV TL1,#0FDH
        MOV PCON,#0H
        SETB TR1
        CLR TI
```

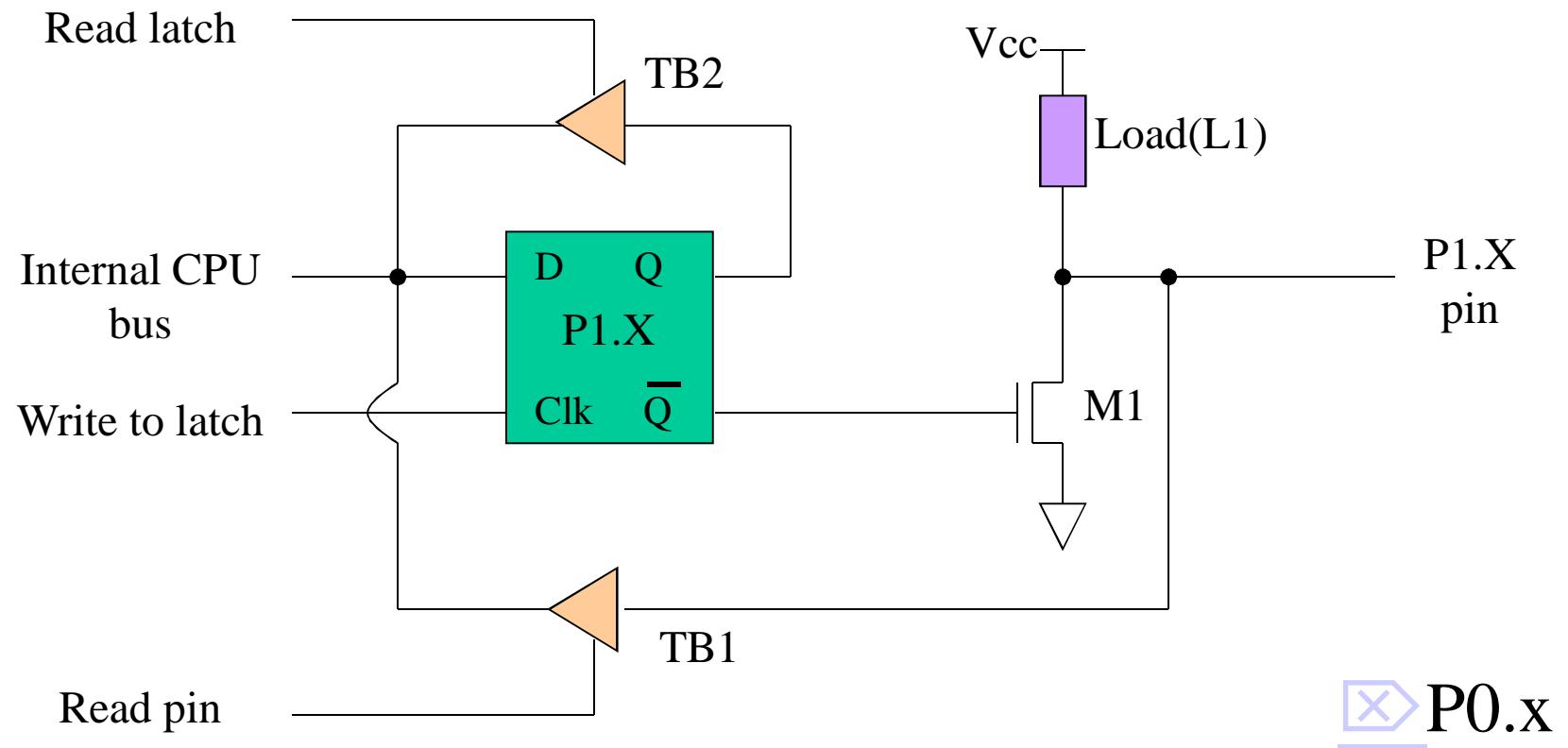
```
MOV DPTR,#TABELA
MOV R1,#1
MOV SBUF,#'M'
```

```
VOLTA: JNB TI,VOLTA
        CLR TI
        MOV A,R1
        MOVC A,@A+DPTR
        MOV SBUFA,A
        INC R1
        CJNE R1,#16,VOLTA
        CLR TR1
        JMP $
```

```
TABELA: DB 'Microcontrolador'
END
```

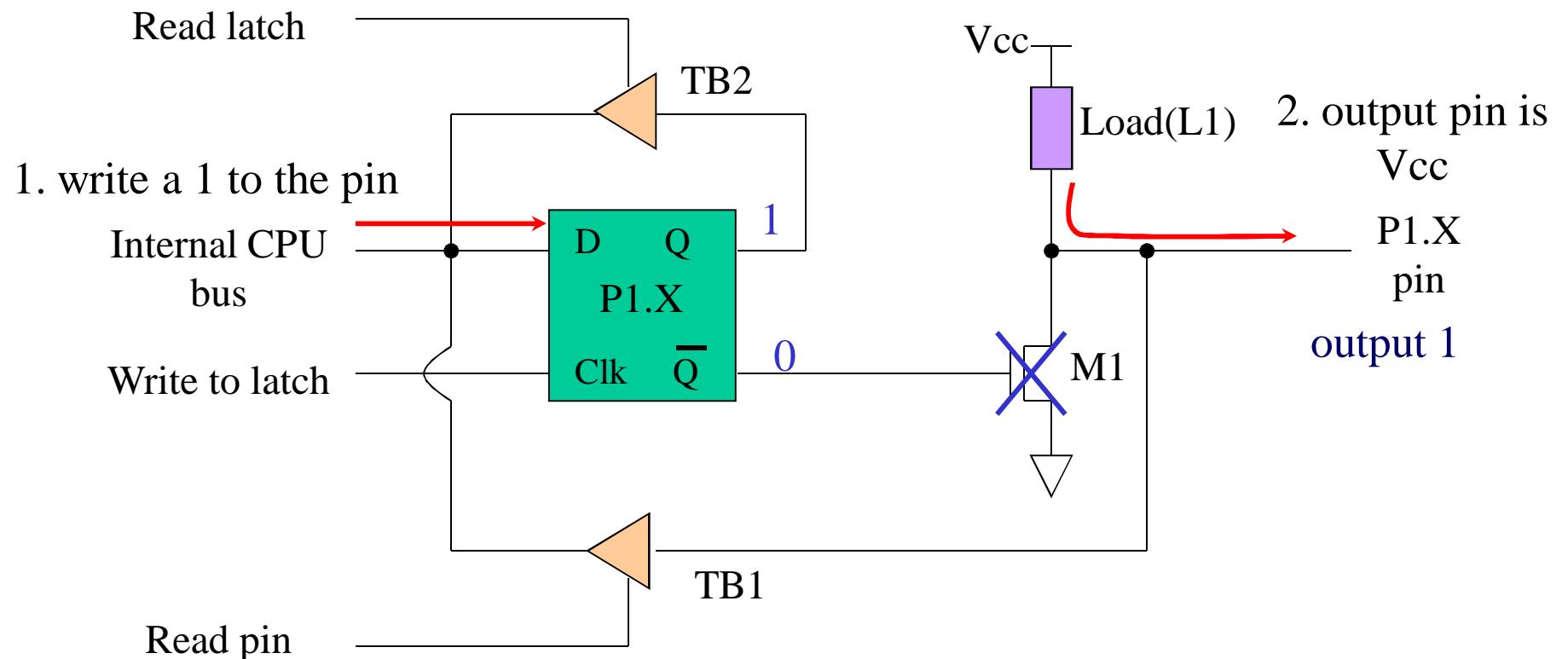
- **SM2:** Habilita comunicação serial com múltiplos processadores nos modos 2 e 3 (aceitam 9 bits)
- **Nestes modos, se SM2 for colocado em '1', RI não será setado a menos que o nono bit recebido (RB8) seja '1'.**
- Desta forma, é possível que um dispositivo acesse vários 8051 em um mesmo barramento e peça para um deles modificar SM2 para que estabeleça comunicação apenas com o mesmo para TB8= '0'.

# Organização Pinos da Porta P1



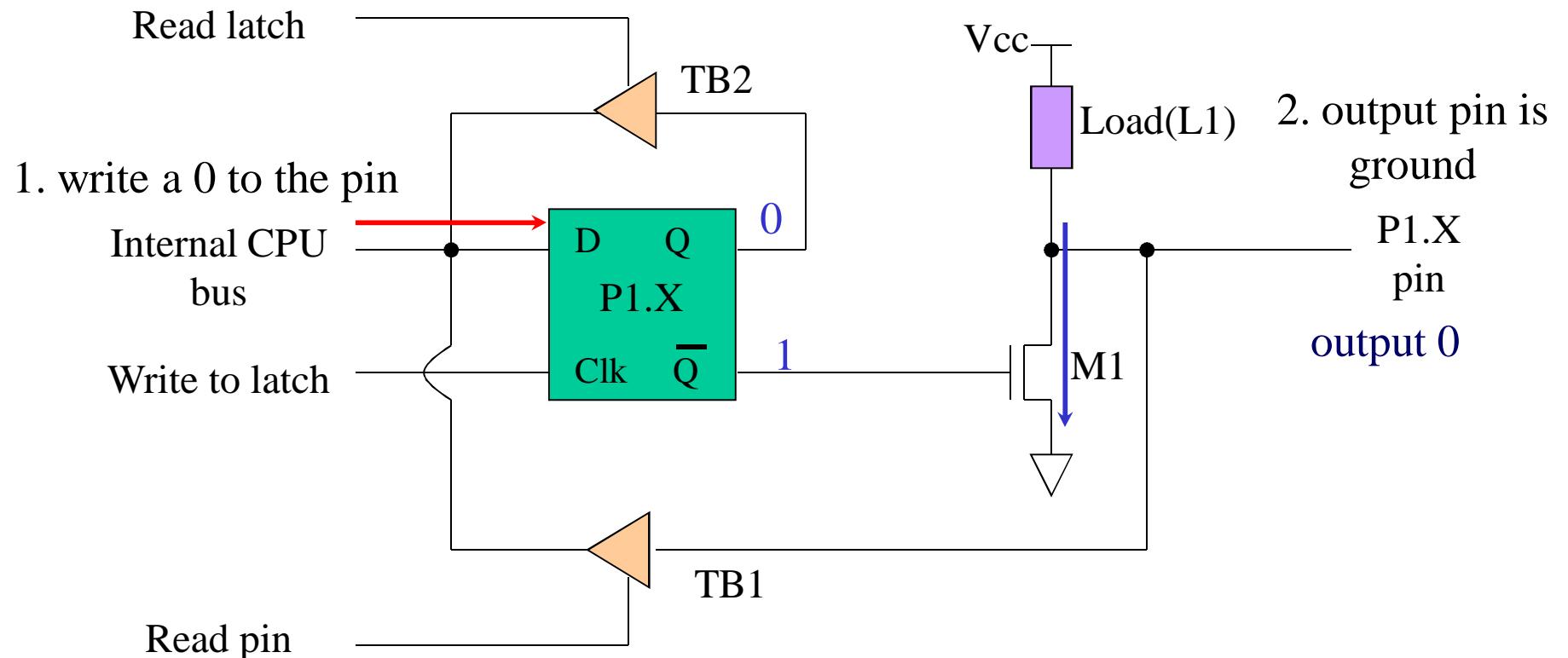
8051

# Escrevendo “1” em Pino P1.X



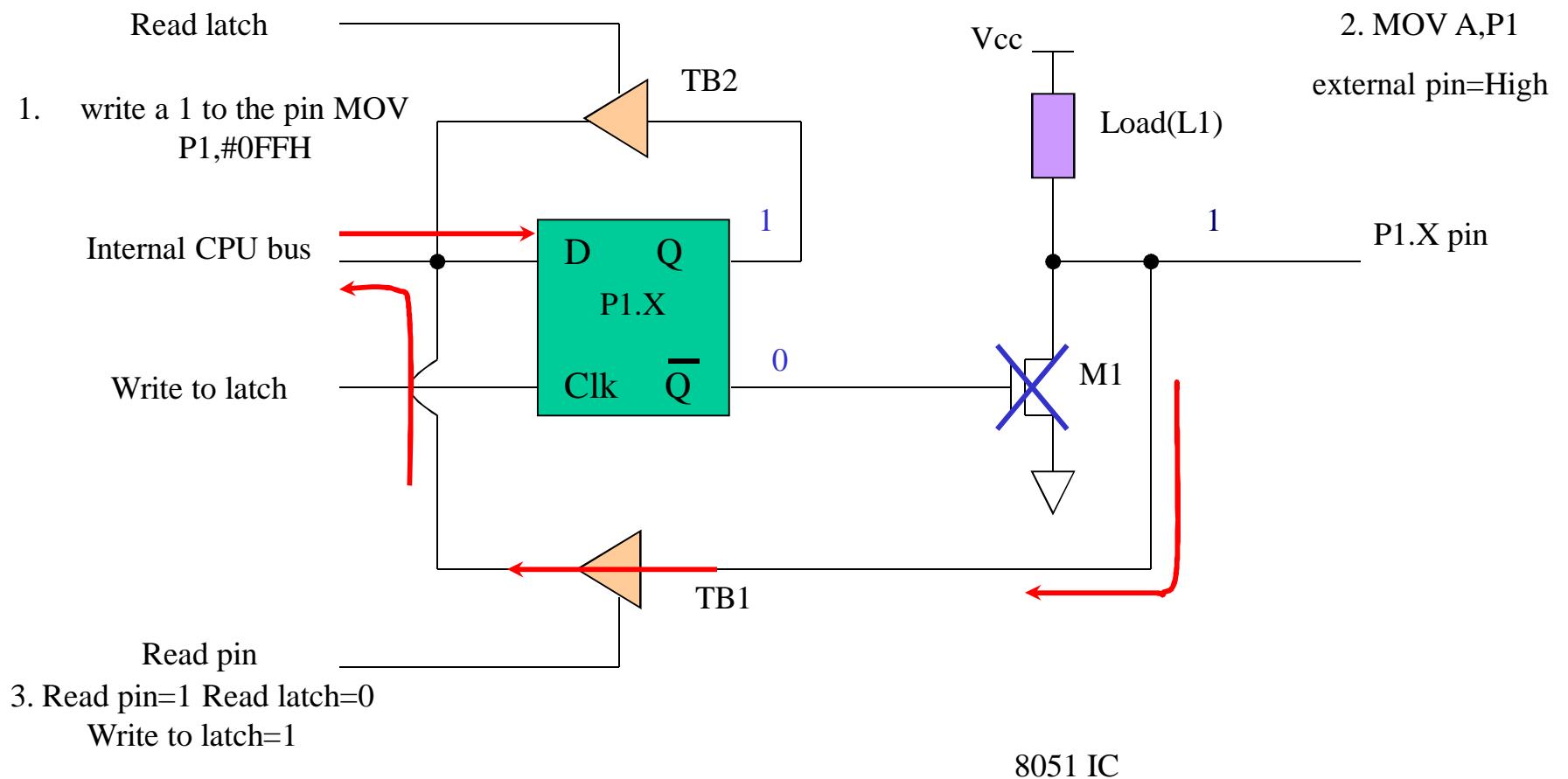
8051

# Escrevendo “0” em Pino P1.X

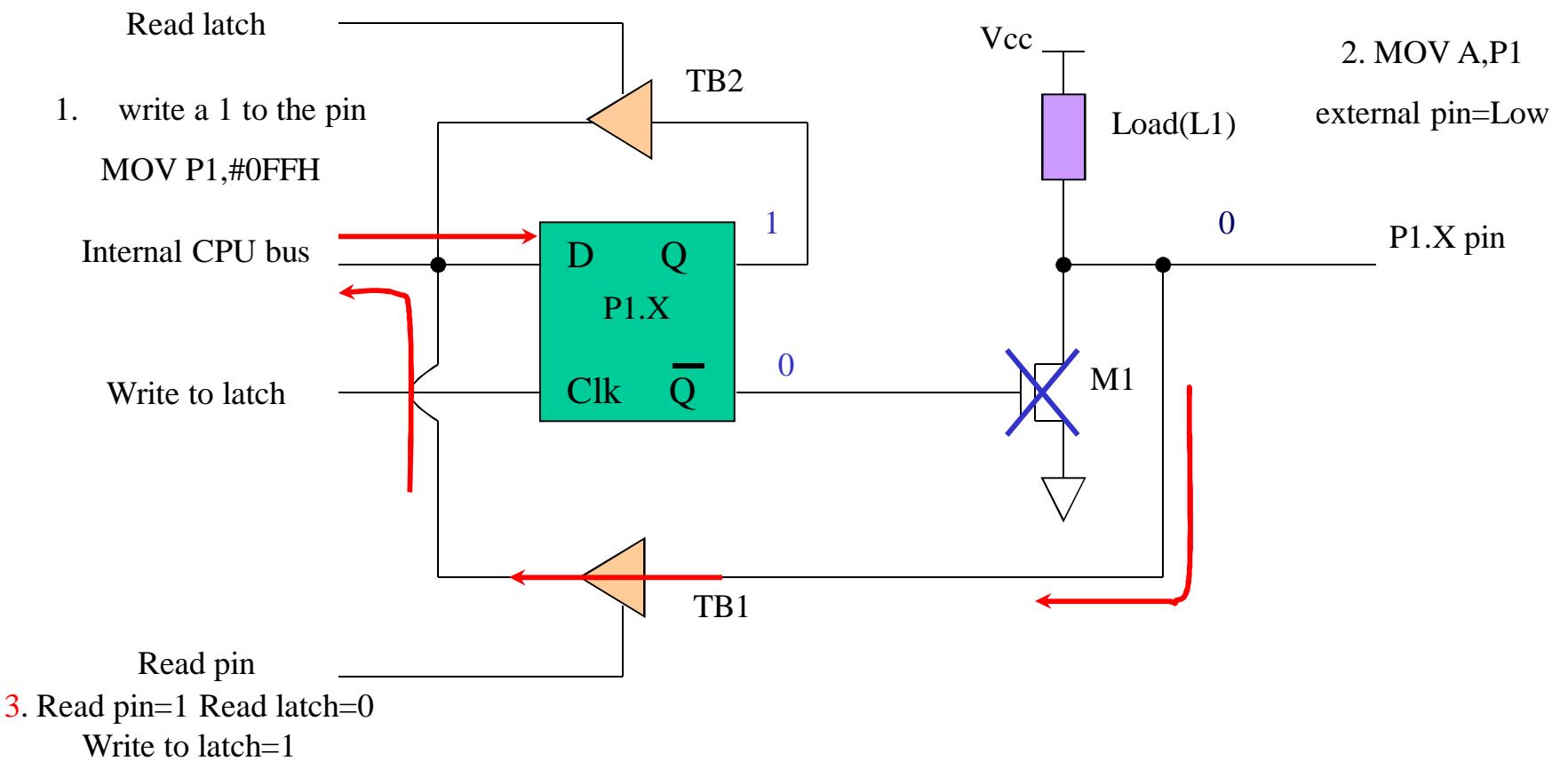


8051 IC

# Lendo “1” em Pino P1.X



# Lendo “0” em Pino P1.X



**8051**

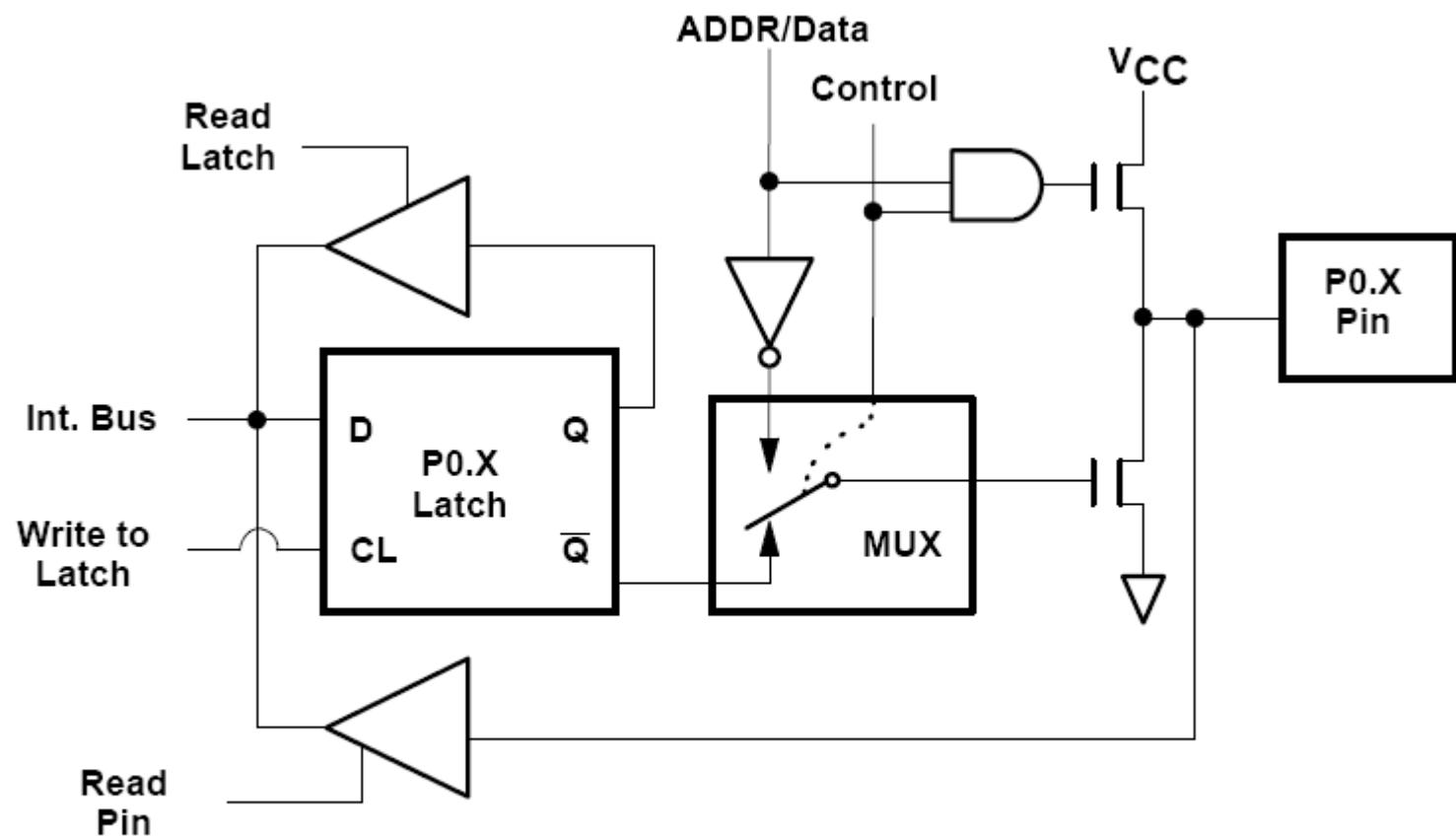
# Lendo do Latch de Porta

Algumas instruções fazem a leitura do Pino (TB1) e outros da saída do Latch (TB2)

Instruções que leêm do latch (“lê-modifica-escreve”). Exemplos:

ANL	AND lógico	ex. ANL P1,A
ORL	OR lógico	ex. ORL P2,A
XRL	XOR lógico	ex. XRL P3,A
CPL	complementa bit	ex. CPL P3.0
INC	incrementa	ex. INC P2
DEC	decrementa	ex. DEC P2
DJNZ	decrementa e salta se não zero	ex. DJNZ P3,LABEL
MOV PX.Y,C	move bit de carry para bit Y da Port X	
CLR PX.Y	limpa bit Y da Port X	
SETB PX.Y	seta bit Y da Port X	

# Port 0



a. Port 0 Bit

## **Outros Modos de Transmissão Serial**

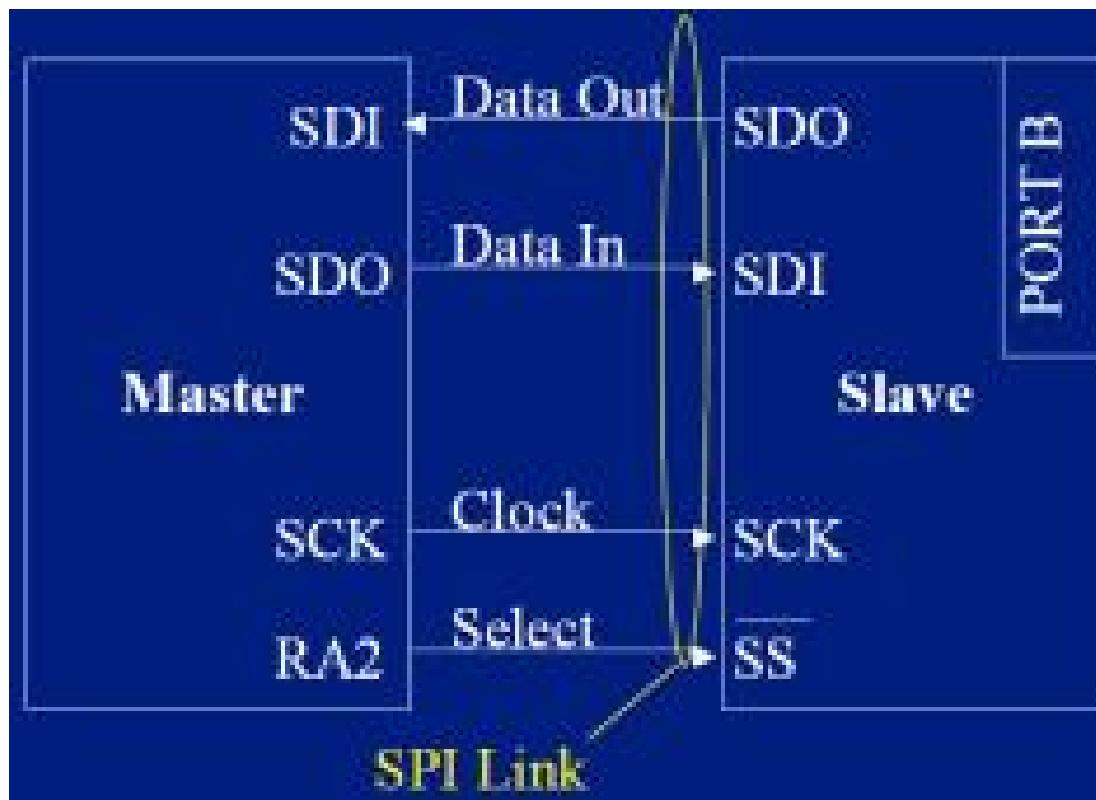
# SPI: Serial Peripheral Interface

**SCK**: Serial Clock (suprido pelo mestre)

**SDI**: Serial Data In

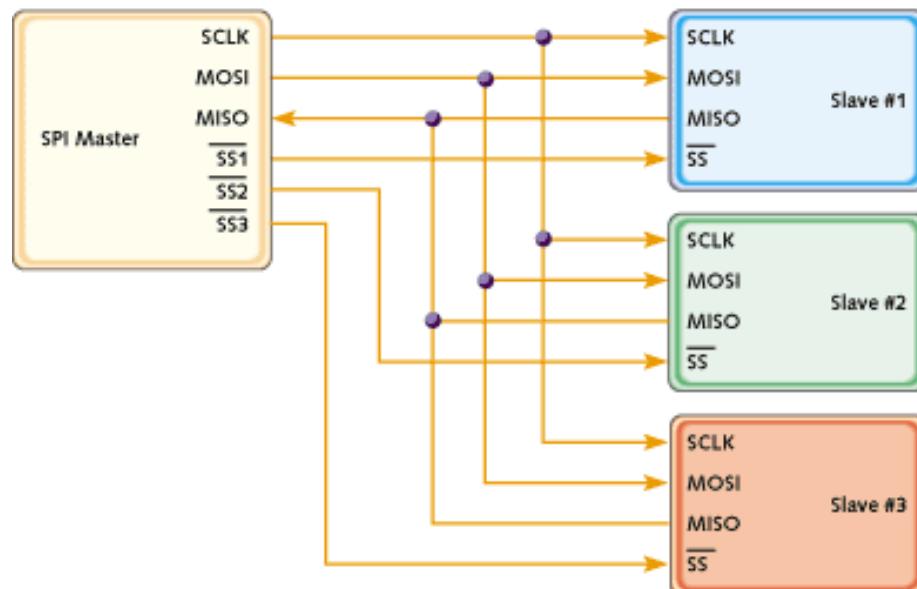
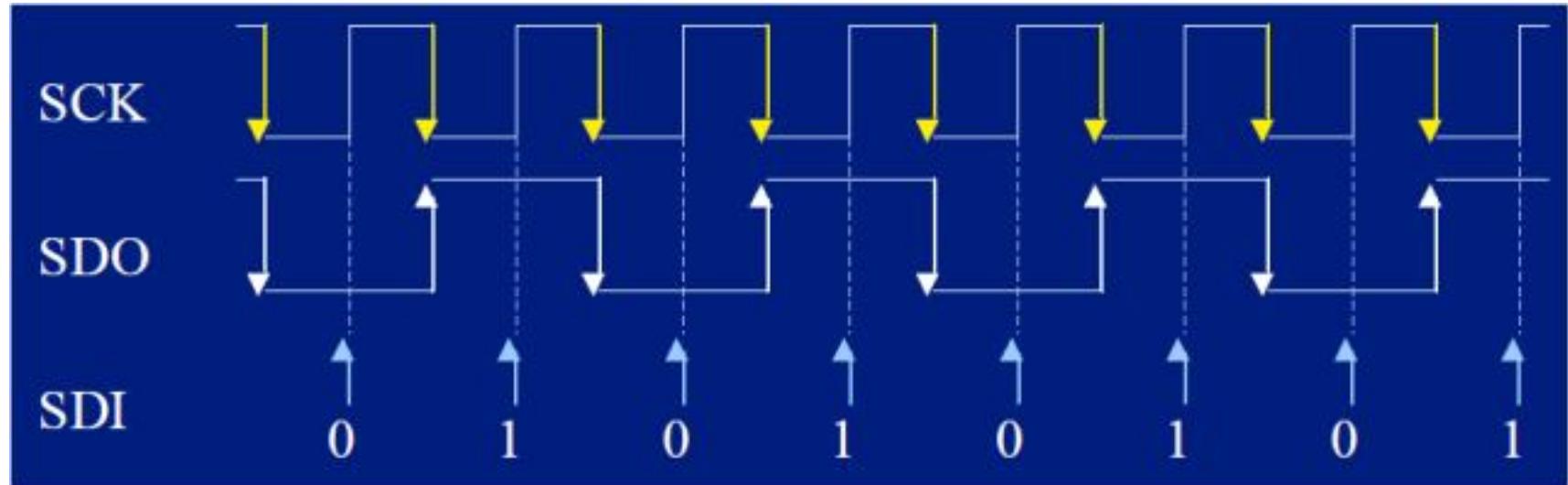
**SDO**: Serial Data Out

**nCS, SS**: Chip Select, Slave Select



- Full Duplex, sem controle do fluxo de dados
- Velocidade de transmissão de dados de dezenas de Mbi/s

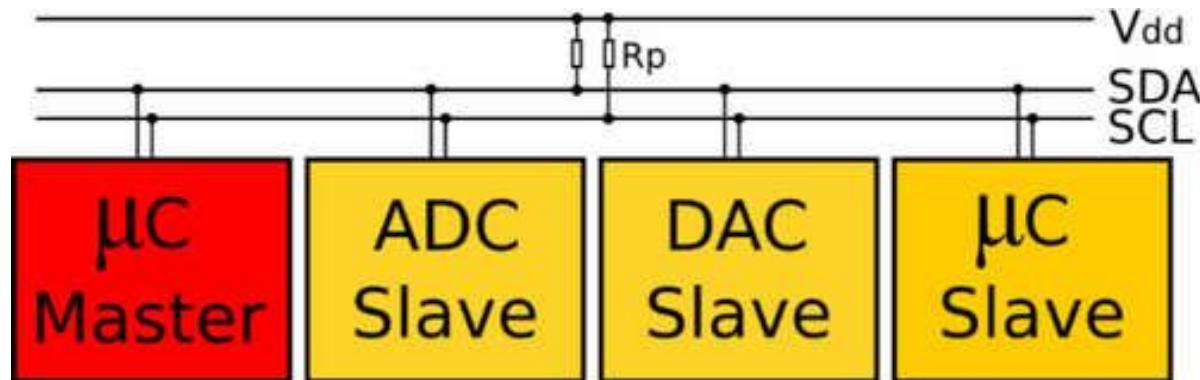
# SPI: Serial Peripheral Interface



# Transmissão Serial

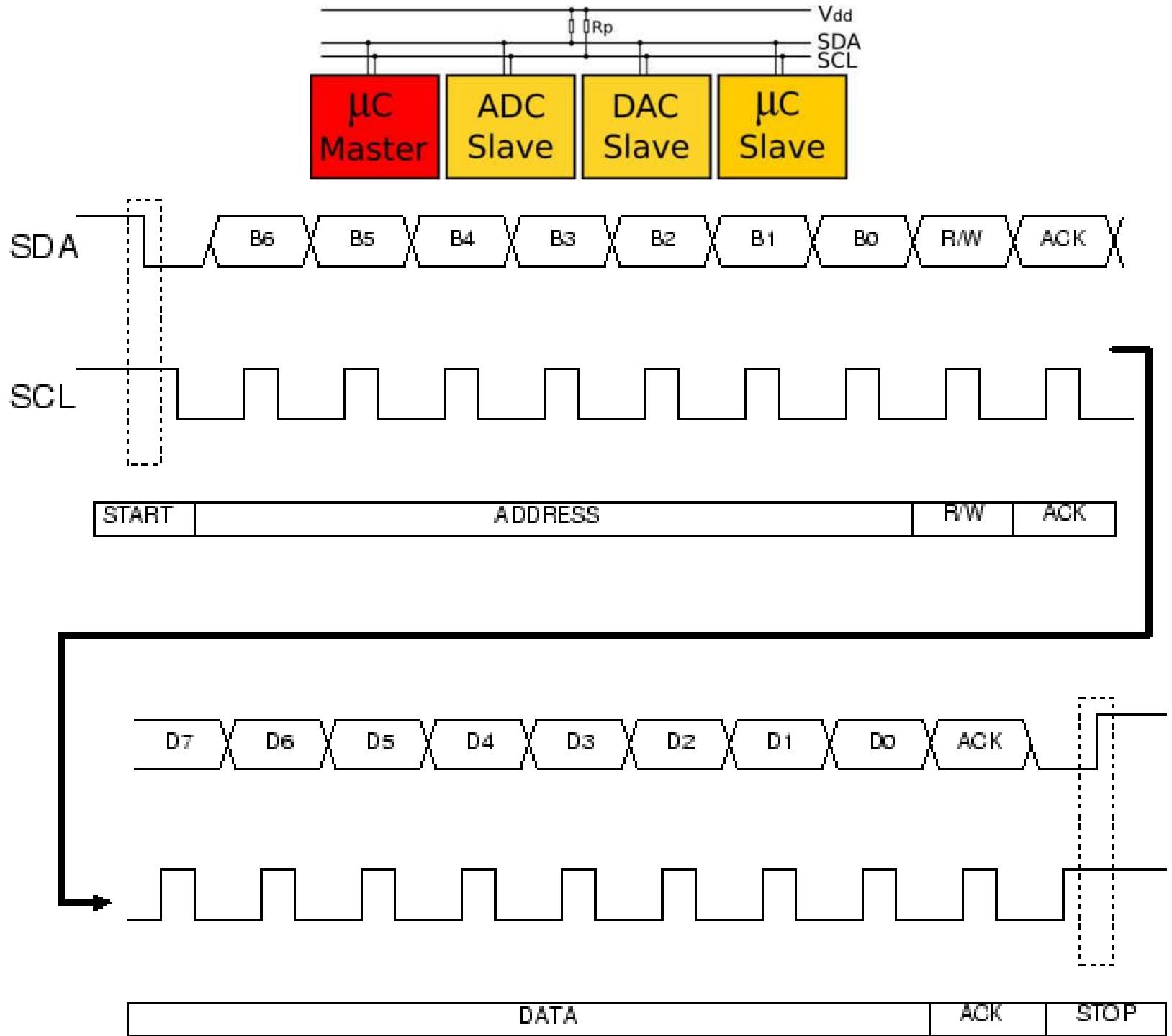
## I<sup>2</sup>C: *Inter-Integrated Circuit*

Mestre Único



**SDA** - Serial DAta line  
**SCL** - Serial CLock line

- Mestre envia clock e endereço do escravo.
- 7 bits são utilizados para endereçamento; há 16 endereços reservados  
=> máximo de 112 escravos
- Cls tem endereço fixo ou os bits menos significativos do endereço do dispositivo são especificados por pinos
- Velocidade de transmissão de dados => Até 3,4 Mbi/s no modo *High Speed*
- Mestre e escravo podem trocar de papel após bit de stop.



## Comunicação de dados entre Mestre-Escravo

Etapa	Escrita	Leitura	Nro Bits	Campo
1	<b>START bit</b>	<b>START bit</b>	1	<b>START</b>
2	<b>Slave address</b>	<b>Slave address</b>	7	<b>ADDRESS</b>
3	<b>bit 0</b>	<b>bit 1</b>	1	<b>W/R</b>
4	<b>Aguarda bit ‘0’ de “recebido”</b>	<b>Aguarda bit ‘0’ de “recebido”</b>	1	<b>ACK</b>
5	<b>Envia dados</b>	<b>Recebe dados</b>	8	<b>DATA</b>
6	<b>Aguarda bit ‘0’ de recebido</b>	<b>Envia bit ‘0’ de recebido ou vai para passo 7</b>	1	<b>ACK</b>
7	<b>STOP bit</b>	<b>STOP bit</b>	1	<b>STOP</b>

Os passos 5 e 6 podem ser repetidos tal que múltiplos bytes sejam transmitidos ou recebidos. Ou seja, após 6, retorna-se ao passo 5 ou prossegue para o passo 7.