80C51

Parte 2

Endereços dos Vetores de Interrupção

Fonte	Endereço
IE0	03H
TF0	0BH
IE1	13H
TF1	1BH
RI&TI	23H

O 8051 inicia sua execução no endereço 0000H após Reset.

IP: Registrador de Prioridade de Interrupção

-----	-----	-----	PS	PT1	PX1	PT0	PX0

- **PS : Interface Serial.**
- **PT1 : Timer 1.**
- **PX1 : Interrupção Externa 1.**
- **PT0 : Timer 0.**
- **PX0 : Interrupção Externa 0.**

0 = Baixa prioridade.

1 = Alta prioridade.

80C51(CMOS) vs. 8051(NMOS)

- Redução da Potência Controlada
  - Estado “Idle”
  - Estado “Power down”
- Economia de potência nas portas CMOS
- Flags de Software de Propósito Geral
- Versões de alta velocidade nos 80C51 (até 30MHz)
- Versões estáticas em desenvolvimento

PCON : Registrador de Controle de Potência

SMOD	----	----	----	GF1	GF0	PD	IDL

☞ **Operação em "POWER DOWN"**

- PD=1, para o oscilador.

- Conteúdo da RAM é preservado.

- Saída via Reset.

- Alguns (novos) derivados do 80C51 permitem sair do "Power-Down" via Interrupção.

PCON : Registrador de Controle de Potência

- **OPERAÇÃO EM MODO "IDLE"**
  - **- IDL = 1, desconecta clock , porém o oscilador continua ativo.**
  - **- Todos os registradores e a RAM continuam intactos.**
  - **- Fontes de interrupção continuam ativas:**
    - **Interface Serial.**
    - **Interrupções externas.**
    - **Timers.**
  - **- Saída com qualquer interrupção habilitada ou Reset.**

- **GF0, GF1 são flags de software de propósito geral.**

- **SMOD bit de controle da interface serial.**
  - **- Dobra "baud rate" nos modos 1,2, e 3.**

- **SMOD disponível somente na tecnologia NMOS.**

Consumo de Potência

Exemplo : para 80C51 com Vcc = 5V.

Modo / Freq.	0.5 MHz	16 MHz
Operação	2.2 mA	20.5 mA
“Idle”	0.9 mA	5.0 mA
“Power Down”	50 uA	50 uA

Modos de Endereçamento do 8051

Há basicamente 5 maneiras de endereçamento fonte/destino de operando:

1. Recursos "On-chip" :

Isto inclui o Acumulador (A), Stack Pointer (SP), Data Pointer (DP), Program Counter (PC), e Carry (C). Outros registradores são mapeados em memória e possuem Op-codes especiais.

2. Operando Imediato:

Determinado pelo sinal # . São 8-bits exceto para o conteúdo do DPTR (16-bits).

3. Operando Registrador:

Indicado como Rn, onde n varia de 0..7. É usado um dos quatro bancos de Registradores (seleção em dois bits de PSW).

4. Operando Direto:

De 00 to FF Hexadecimal, especifica o endereço das posições de dados internos.

5. Endereçamento indireto:

Designado como @Ri, onde *i* é 0 ou 1, usa o conteúdo de R0 ou R1, do Banco de Registradores selecionado, para especificar o endereço. Outra forma é @A, usando o conteúdo do Acumulador.

Conjunto de Instruções : Aritmética

Mnemônicos	Operandos	Bytes/Ciclos
ADD	A, Rn	1/1
ADDC	A, direct	2/1
SUBB	A, @Ri	1/1
	A, #data	2/1
INC	A	1/1
DEC	Rn	1/1
	direct	2/1
	@Ri	1/1
INC	DPTR	1/2
MUL	AB	1/4
DIV	AB	1/4
DA	A	1/1

Conjunto de Instruções : Lógica

Mnemônicos	Operandos	Bytes/Ciclos
ANL	A, Rn	1/1
ORL	A, direct	2/1
XRL	A, @Ri	1/1
	A, #data	2/1
	direct, A	2/1
	direct, #data	3/2
	C, bit	2/2
	C, /bit	2/2
CLR	A	1/1
CPL	C	1/1
	bit	2/1

Conjunto de Instruções : Lógica (cont.)

Mnemônicos	Operandos	Bytes/Ciclos
RL	A	1/1
RLC	A	1/1
RR	A	1/1
RRC	A	1/1
SWAP	A	1/1
SETB	C	1/1
CLR	bit	
CPL		
		2/1

Conjunto de Instruções : Transferência de Dados

Mnemônicos	Operandos	Bytes/Ciclos
MOV	A, Rn	1/1
	A, direct	2/1
	A, @Ri	1/1
	A, #data	2/1
	Rn, A	1/1
	Rn , direct	2/2
	Rn, #data	2/1
	direct, A	2/1
	direct, Rn	2/2
	direct, direct	3/2
	direct, @Ri	2/2
	direct, #data	3/2

Conjunto de Instruções : Transferência de Dados (cont.)

Mnemônicos	Operandos	Bytes/Ciclos
MOV	@Ri, A	1/1
	@Ri, direct	2/2
	@Ri, #data	2/1
	DPTR, #data16	3/2
	C, bit	2/1
	bit, C	2/2
MOVX	A,@DPTR	1/2
	@DPTR,A	1/2
	A,@Ri	1/2
	@Ri,A	1/2

Conjunto de Instruções : Transferência de Dados (cont.)

Mnemônicos	Operandos	Bytes/Ciclos
MOVC	A, @A+DPTR	1/2
	A, @A+PC	1/2
PUSH	direct	2/2
POP	direct	2/2
XCH	A, Rn	1/1
	A, direct	2/1
	A, @Ri	1/1
XCHD	A, @Ri	1/1

Conjunto de Instruções : Desvios

Mnemônicos	Operandos	Bytes/Ciclos
LCALL	addr16	3/2
ACALL	addr11	2/2
RET	-	1/2
RETI	-	1/2
LJMP	addr16	3/2
AJMP	addr11	2/2
SJMP	rel	2/2
JMP	@A+DPTR	1/2
JZ	rel	2/2
JNZ	rel	2/2

Conjunto de Instruções : Desvios (cont.)

Mnemônicos	Operandos	Bytes/Ciclos
CJNE	A, direct, rel	3/2
	A, #data, rel	3/2
	Rn, #data, rel	3/2
	@Ri,#data,rel	3/2
DJNZ	Rn, rel	2/2
	direct, rel	3/2
NOP	-	1/1
JC	rel	2/2
JNC	rel	2/2
JB	bit, rel	3/2
JNB	bit, rel	3/2
JBC	bit, rel	3/2

Codificação do 80C51 :

Idéias e Exemplos

Lendo um Timer "em vôo"

```
ReadTimer:
        MOV     ValH,TH0            ;Lê os valores iniciais do timer 0.
        MOV     ValL,TL0
        MOV     A,TH0               ;Lê o byte mais significativo do timer 0.
        CJNE    A,ValH,Readagain    ;Mudou ?
        SJMP    RTEX                ;Se não, retorna.

Readagain:
        MOV     ValH,TH0            ;Lê novamente valores  do timer 0.
        MOV     ValL,TL0
RTEX:
        RET
```

Comparação

O 80C51 não possui instrução básica para comparação. Contudo, a instrução CJNE (compare and jump if not equal) deixa a flag de carry = 1 depois da execução, se o dado em comparação for maior que o conteúdo do registrador em questão (A, Rn ou @Ri), permitindo assim comparações de magnitude adicionais.

Este método vale para todas as variações de CJNE:

```
CJNE    A,direct,rel

CJNE    A,#data,rel

CJNE    Rn,#data,rel

CJNE    @Ri,#data,rel
```

Comparação

Exemplos de quatro variações comparação de magnitude:

```
        CJNE   A,Valor,Test      ;Desvie se A < Valor.
Test:   JC     LT

        CJNE   A,Valor,Test      ;Desvie se A >= Valor.
Test:   JNC    GTE

        CJNE   A,Valor,Test      ;Desvie se A > Valor.
        SJMP   Else
Test:   JNC    GT
Else:   -----

        CJNE   A,Valor,Test      ;Desvie se A <= Valor.
        SJMP   LTE
Test:   JC     LTE
```

Leitura e Escrita - Modificação nas Portas

Muitas instruções que fazem referência à dados nas portas do 80C51, lêem o valor nos pinos das portas ao invés de fazê-lo no "latch" das mesmas. Outras instruções fazem o inverso.

1) Operações aritméticas ou lógicas que podem alterar os valores das portas:

```
ANL     port,src
ORL     port,src
XRL     port,src
INC     port
DEC     port
DJNZ    port,label
```

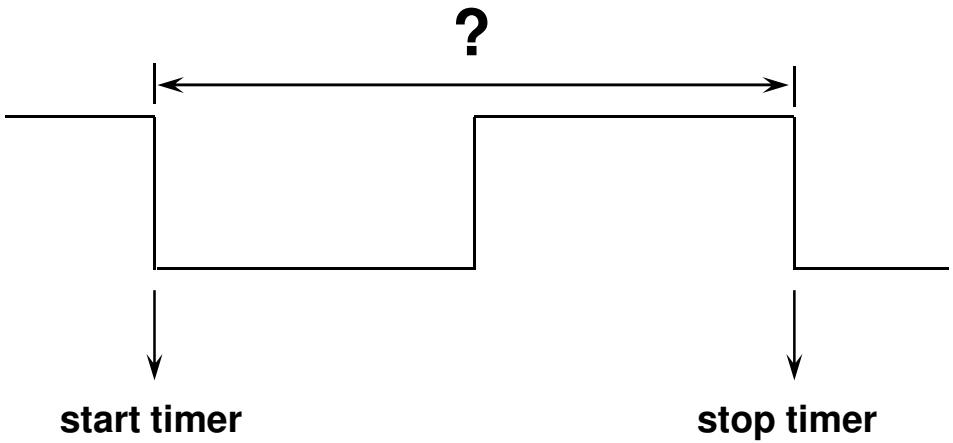
3) Instruções que podem alterar os bits das portas:

```
MOV     bit,C
JBC     bit,label
CPL     bit
CLR     bit
SETB    bit
```

Medida de Largura de Período

Problema:

- Medir o período de um pulso de entrada



Assumindo:

- Usar interrupção externa 0 para entrada do pulso.

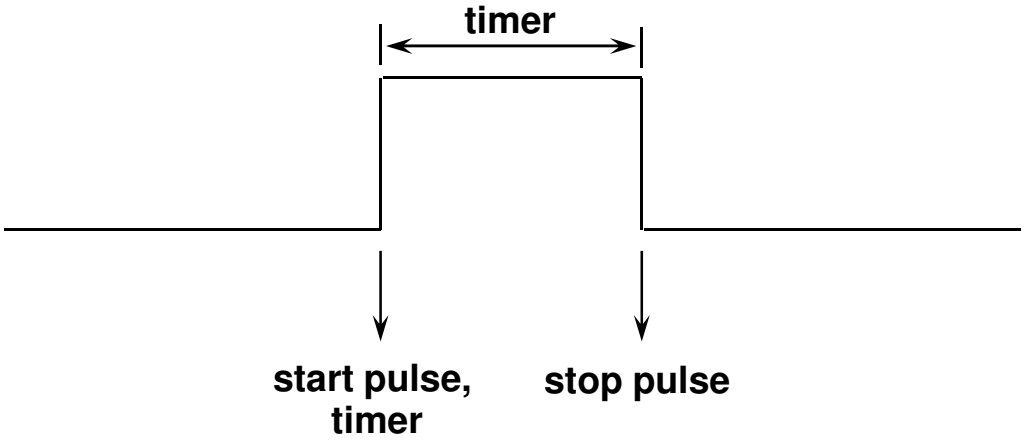
Medida de Largura de Período

```
Setup:      (como no exemplo anterior, mas deixando a função "gate" do timer "off")

            MOV     TMOD,#01h           ;Timer 0 no modo 1.

;Rotina de serviço de Interrupção Externa 0.
EXINT0:     CPL     TR0                 ;Complementa a flag "timer run". Isso
parte e                                 ;para o timer em interrupções alternadas.
            JB      TR0,INT0EX          ;Sai se o timer estiver rodando.
            MOV     ValH,TH0            ;Se não, lê o valor do timer
            MOV     ValL,TL0
            MOV     TH0,#0              ;Inicializa o timer de modo que outra
            MOV     TL0,#0              ;leitura possa ser feita
INT0EX:     RETI
```

Criando um Pulso de Saída

Problema:

- Criar um pulso de duração desconhecida em um pino de uma porta.



Assumindo:

- Usar qualquer bit livre de qualquer porta para saída.

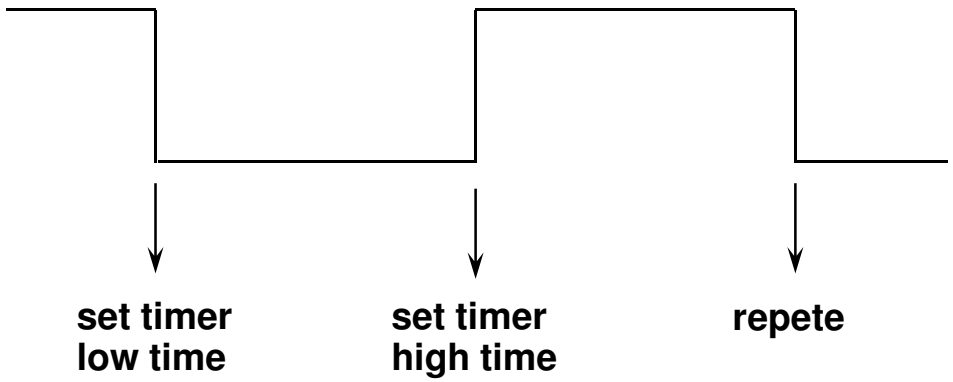
Criando um Pulso de Saída

```
Setup:      MOV   TCON,#0h        ;Certificar que o timer esteja parado.
            MOV   TMOD,#01h       ;Timer para modo 1.
            MOV   TH0,#HiTime     ;Carrega timer com a duração do pulso. O valor
            MOV   TL0,#LoTime     ; deve estar em complemento de dois do número
                                  ; de ciclos de máquina, para ser usado como
                                  ; largura do pulso.
            SETB  TR0             ;Habilita timer.
            SETB  P2.0            ;Habilita pulso

;Rotina de interrupção do Timer 0.
T0INT:      CLR   P2.0            ;Fim do pulso.
            CLR   TR0             ;Para timer.
            RETI
```

Programando uma saída PWM

Problema:

- Criar uma saída PWM em um pino de uma porta.



Programando uma saída PWM

```
T0INT:      CLR     TR0                 ;Para timer.
            CPL     P2.0                ;Complementa bit de saída.
            JB      P2.0,SetPWMHigh     ;Estado corrente é alto ou baixo?
            MOV     TH0,PWMLowH         ;Inicializa PWM p/ tempo menor.
            MOV     TL0,PWMLowL
            SJMP    T0EX

SetPWMHigh: MOV     TH0,PWMHighH        ; Inicializa PWM p/ tempo maior.
            MOV     TL0,PWMHighL

T0EX:       SETB    TR0                 ;Re-inicializa timer.
            RETI
```

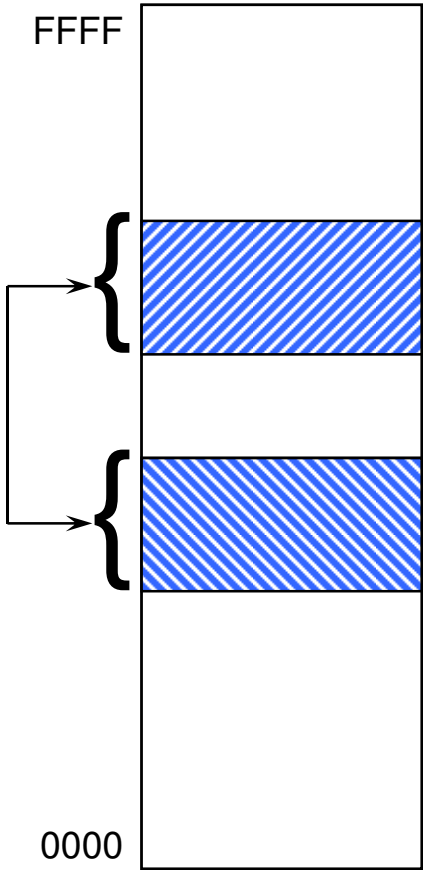
Nota:

Para pulsos de alta freqüência, é possível usar a característica de recarga de timer (modo 2) para obter durações de pulsos mais precisas.

Movimento de Blocos de Memória com Memória de Dados Externa

Problema:

Mover qualquer bloco de memória de dados externa, de qualquer comprimento, para outra localização.



Mover Bloco de Memória

```
BlockMove:  MOV   R0,#LOW(FromAddr)    ;Inicializar ponteiro de memória (de onde).
            MOV   P2,#HIGH(FromAddr)
            MOV   DPTR,#ToAddr         ;Inicializar ponteiro de memória (para onde).
            MOV   R1,#LOW(ByteCount)   ;Inicializar contador de bytes.
            MOV   R2,#HIGH(ByteCount)

Loop:       MOVX  A,@R0                ;Lê um byte do bloco fonte.
            MOVX  @DPTR,A              ;Escreve um byte no bloco destino.
            INC   DPTR                 ;Incrementa ponteiro de memória (para onde).
            INC   R0                   ;Incrementa ponteiro de memória (de onde).
            MOV   A,R0
            JNZ   L1
            INC   R2

L1:         DJNZ  R1,Loop              ;Decrementa contador de byte e
            DJNZ  R2,Loop              ;  testa se é fim de bloco
            RET
```