MOTOR DE PASSO, GIRO CONTÍNUO

– ROTINA DE ATRASO COM **TIMER0**

Este programa é similar ao anterior, no que se refere ao programa principal. O que muda é a rotina de atraso, que neste programa faz uso do timer0 (ver apresentação 80c51 parte 1).

Passo 1 -Como a frequência do motor de passo para este programa é de 100Hz, e portanto seu período muito maior que o de um ciclo de máquina de uma instrução do microcontrolador, foi considerado somente o tempo de execução da subrotina MEIOT na geração do período do sinal de clock, desprezando-se dessa forma, o tempo ti das instruções dentro do loop do programa principal. O timer0 é constituído por dois registradores de 8 bits, TH0 (8MSB) e TL0 (8LSB), e seu uso nesta aplicação, requer a programação do registrador TMOD como segue:

1- Os 4 MSB pertencem ao timer1(não usado) e são programados como "0".
2- GATE do timer0 é programado como "0" indicando disparo por software por meio do bit TR0 (do registrador TCON).
3- C/T=0 indica que o timer0 é programado como **timer,** o que significa que a frequência de entrada do timer0 é = (frequência do cristal)/12.
4- O modo de operação M1M0 = 01, indicando que o timer0 será utilizado como timer de 16 bits

O valor do TMOD resultante é = 01H

Passo 2- A etapa seguinte consiste em determinar qual valor deve ser carregado em TH0 e TL0 para que a frequência do clock do motor de passo gerada **pelo loop do programa principal** seja de 100Hz. Observe que a subrotina MEIOT é responsável por gerar **meio** período ou seja (1/2)*(1/100)s = (1/200)s. A seguir é mostrado o cálculo dos valores a serem carregados em TH0 e TL0

TIMER 0: 16 bits

- função TIMER (incrementado a cada ciclo de máquina: $f_{xtal}/12$)

TH0: 8 MSB e **TL0:** 8 LSB

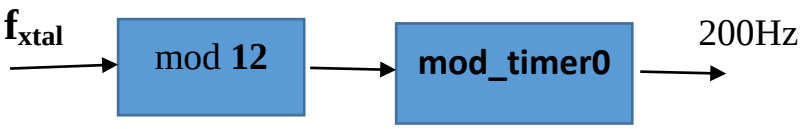
TH0	TL0

- Disparo por software (bit **TR0** do registrador TCON);

- Controle de fim de contagem com flag **TF0** (registrador TCON); quando **TH0 TL0**=FFFFH, TF0 = 1

Se f_M = 100Hz então $\mathbf{T_{meiot}}$ = 1/(200)s ou $\mathbf{f_{meiot}}$ = 200Hz

Internamente, de acordo com a programação de TMOD, o timer0 é alimentado por $f_{xtal}/12$, o que pode ser representado por dois contadores em sequência:



mod 12 é abreviatura de módulo 12.

Pode-se escrever que a $f_{xtal}/200$ é igual à multiplicação dos módulos:

(12) x mod_timer0 = (11059200)/200

mod_timer0= 4608 ; que é o número de estados que o timer0 deve gerar.

Como o timer0 conta no modo crescente, e é de 16 bits, o valor X e Y a ser carregado em TH0 e TL0 será :

$XY = 2^{16} - 4608 = 65536 - 4608 = 60928 = EE00H$

Dessa forma: TH0 recebe EEH (ou X= 0EEH)

e TL0 recebe 00H (ou Y = 00H)

O timer 0 contando de EE00H até FFFFH gera 4608 estados.

Qual a menor frequência que o timer0 consegue gerar?

Resposta: a menor frequência é obtida quando o timer0 for programado para gerar o número máximo de estados, que nesse caso é 2^{16} ou 65536 estados:

$XY = 2^{16} - 65536 = 65536 - 65536 = 0000H$ (X = 00H e Y= 00H)

$f_{meiot} = f_{xtal} / (12 \times 65536) = \sim 14Hz$;

Passo 3 - O próximo passo é fazer a **simulação** que é idêntica à realizada para o projeto anterior do motor de passo (loop aninhado). Após determinar a frequência para um passo via simulador, altere o programa para realizar uma volta completa e determine, via simulador, o tempo que o motor leva para realizar 200 passos.

```
CLOCK     BIT   P0.0
DIR       BIT   P0.1
HAB       BIT   P0.2
; programa principal
          ORG  0000H
          MOV TMOD,#01H
          SETB  DIR
          SETB  HAB
          SETB  CLOCK
LOOP:     CLR  CLOCK
          ACALL  MEIOT
          SETB  CLOCK
          ACALL  MEIOT
          SJMP  LOOP
; Rotina de atraso com  o  TIMER 0
MEIOT:    MOV TL0,#XH
          MOV TH0,#YH
          SETB TR0
          JNB  TF0,$
          CLR  TR0
          CLR  TF0
          RET
          END
```

Microcontrolador AT89s52:

f_{xtal} = 11.059.200Hz ou

11,0592MHz

T = (1/ f_{xtal})

Motor de Passo:

f_M = 100 Hz e T_M = (1/ f_M)

1 passo = 1,8 graus

No Simulador:

1 – mudar valor do clock para **11059**

2 – na opção SIMULATOR

selecionar: **allow breakpoint**

3 – Selecionar **VIEW** para inserir **número** nas instruções, para usar breakpoint