

Aprendendo a programar com o BASIC Step M8

Neste artigo nós vamos aprender a medir e controlar tempo usando o BASIC Step M8. Você deve ter compreendido o material das partes 1 e 2, se ainda não leu, faça isto antes de prosseguir.

A medição de tempo e o controle de temporização são provavelmente as funções mais usadas nos microcontroladores. Por este motivo o BASIC Step M8 possui 3 contadores / timers que são muito úteis.

Controle de Temporização

Primeiro, vamos escrever um programa de controle de temporização para a placa Super StepLab. Nós vamos escrever este programa para que a porta B seja conectada aos LEDs. Você pode fazer estas ligações com os fios incluídos na placa.

O primeiro comando que iremos usar neste programa é o PAUSE. Este comando permite parar a execução do programa por um tempo determinado. Para especificar o tempo, coloque o tempo desejado, em milisegundos, após a palavra PAUSE. Por exemplo se quisermos fazer o LED piscar uma vez por segundo, ele deverá estar ½ segundo ligado e ½ segundo desligado. Para fazer isto basta ligá-lo ou desligá-lo e então parar o programa por 500 milisegundos. Isto é feito com o comando:

```
PAUSE 500
```

O comando PAUSE permite uma grande variação de tempo. Você pode especificar um tempo tão pequeno quanto 10 microsegundos (PAUSE 0.01) e tão grande quanto 8 segundos (PAUSE 8000)

Um outro comando muito útil é o TOGGLE. Este comando simplesmente muda o estado de um único pino de saída.

Então aqui está o programa para piscar um LED colocado na porta B,4 uma vez por segundo.

```
MAKEOUT B, 4  
  
DO  
    TOGGLE B, 4  
    PAUSE 500  
LOOP
```

Você pode digitar, compilar e testar este programa. Depois mude-o para fazer alguma coisa diferente.

Medição de Tempo

O próximo problema é diferente, ao invés de controlar uma temporização, nós queremos medir um tempo.

Quando nós usamos o comando PAUSE no programa anterior, o compilador utilizou um contador interno ao chip do processador chamado TIMER0. O compilador configurou o contador para uma frequência de incremento correta para o tempo que digitamos, disparou o contador e esperou até que ele chegasse ao seu limite, antes de prosseguir com a execução do programa. Se você achou isto complicado, é porque realmente é, mas o compilador fez todo o trabalho por nós.

Em nosso próximo programa iremos medir o período da frequência entrando pelo pino D,0. Para fazer isto nós iremos esperar até que o sinal vá para nível alto. Assim que detectarmos esta mudança, iremos ler o valor do contador, retorná-lo a zero e esperar pela próxima mudança no pino de entrada.

Vamos assumir que desejamos medir uma frequência a partir de 40 Hz. O contador pode ser incrementado a frequência de operação do processador dividida por 1, 8, 64, 256 ou 1024.

O TIMER0 é um contador de 8 bits, então ele conta de 0 a 255. Então a frequência que ele chega ao seu limite é 256 vezes menor que o seu clock. Vamos calcular a frequência de estouro para cada valor de divisão.

$$\text{Frequência de estouro} = \text{clock do processador} / \text{divisor} / 256$$

Divisor	Frequência de estouro (Hz)
1	31250
8	3906,25
64	488,28
256	122,08
1024	30,52

Como nós queremos medir uma frequência a partir de 40 Hz, precisamos usar um valor de divisor que não de estouro para esta frequência, então vamos usar um divisor de 1024.

Apenas pro diversão, após medir a frequência nós vamos escrever o número (o período do sinal) na porta B para poder ser visualizado nos LEDs. Você pode ligar um gerador de sinais na entrada e ver que quando a frequência muda, os LEDs também mudam.

Aqui está o programa.

```
DIRPORT D, IN
DIRPORT B, OUT

INBIT last, D, 0
```

```
perodo = &HFF

DO
  OUTPORT B,perodo
  INBIT temp,D,0
  IF last = 0 THEN
    IF temp | 0 THEN
      `... aqui é onde lemos o valor do contador
      TIMER0 READ perodo
      `... e aqui o zeramos para a próxima leitura
      TIMER0 ON 1024
    END IF
  END IF
  last = temp
LOOP
```

Nós precisamos comentar sobre como podemos escrever os números no Compilador BASIC Step. Você pode escrever números hexadecimais (base 16) colocando “&H” antes do número. Você também pode escrever números em binário precedendo o número por “&B”. Então a linha do programa:

```
Periodo = &HFF
```

Coloca o valor decimal 255 na variável período.

Ok, nós medimos o período de uma entrada mas como poderíamos medir mais de uma entrada ao mesmo tempo? Existe uma maneira e é apenas um pouco mais complicada que o exemplo anterior.

O que nós vamos fazer é deixar o contador incrementar livremente. A cada vez que detectarmos a mudança desejada no sinal de entrada nós simplesmente lemos o valor do contador. O período do sinal é a diferença entre a leitura atual e a anterior. Este tipo de operação matemática precisa de uma explicação.

Vamos assumir que a frequência de entrada corresponde a 5 incrementos no contador. Se nós detectarmos uma mudança quando o contador estiver em 254, a próxima mudança ocorrerá quando o contador estiver em 3. Usando variáveis byte se nós subtrairmos o valor antigo do atual veremos que

$$03 - 254 = 5$$

Esta equação é verdadeira para a aritmética com bytes pois haverá um estouro de capacidade e um bit de carry será setado, mas o resultado será 5. Vendo em representação binária teremos

$$\begin{array}{r} 00000011 \\ - 11111110 \\ \hline 00000101 \end{array}$$

Conhecendo o BASIC Step M8 – parte 3

Então vemos que subtraindo leituras sucessivas sempre teremos uma resposta correta, desde que o período de entrada seja menor que o período de estouro do contador.

Vamos assumir novamente que os nossos sinais de entrada não serão menores que 40 Hz. Nós configuraremos o contador para incrementar livremente e apenas iremos esperar que os sinais de entrada mudem de estado. É o que faremos no programa a seguir. Os valores dos períodos dos dois sinais serão armazenados em per0 e per1 mas não serão mostrados nas portas.

```
DIRPORT D,IN
TIMER0 ON 1024
per0 = &HFF      `...contador do período para D,0
per1 = &HFF      `...contador do período para D,1

DO
    `...faz a medição em D,0
    INBIT tempd0,D,0
    IF last0 = 0 THEN
        IF tempd0 | 0 THEN
            TIMER0 READ temp
            per0 = temp - temp0
            temp0 = temp
        END IF
    END IF
    last0 = tempd0

    `...faz a medição em D,1
    INBIT tempd1,D,1
    IF last1 = 0 THEN
        IF tempd1 | 0 THEN
            TIMER0 READ temp
            per1 = temp - temp1
            temp1 = temp
        END IF
    END IF
    last1 = tempd1
LOOP
```

Você deve copiar este programa, compilar e testar. Não há como testar do modo que ele está pois não indica os períodos. Tente modificar para indicar o período na porta B. Quando estiver funcionando, tente alterar para mostrar per0 ou per1 dependendo do estado de um botão em D,2. Outra idéia é comparar per0 com per1 e ligar ou desligar um LED dependendo de quem for maior.

No próximo artigo continuaremos a aprender sobre o BASIC Step M8. Nós iremos usar o conversor A/D e a UART.