

Aprendendo a programar com o BASIC Step M8

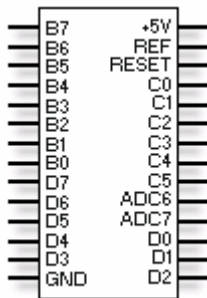
O BASIC Step M8 é o mais novo processador desenvolvido pela Tato Equipamentos Eletrônicos. O seu baixo custo e grande capacidade de processamento o torna perfeito para todos os tipos de controle. O compilador para ele pode ser baixado gratuitamente no site www.tato.ind.br. Com ele você pode escrever programas em BASIC e C, estes programas são compilados para um formato muito eficiente e rápido, fazendo com que o BASIC Step M8 seja tão ou mais poderoso que os outros processadores do mercado.

A gravação do processador é muito simples e rápida, não dependendo de gravadores caros e difíceis de usar. Um simples e barato cabo de gravação é tudo que você precisa para trabalhar. O gravador já está incluído no próprio compilador fazendo com que o trabalho de desenvolvimento seja extremamente simples.

Esta série de artigos serve como introdução ao BASIC Step M8, já que ele tem muito mais recursos que o BASIC Step 1.

O BASIC Step M8 possui 3 portas que podem ser configuradas como entrada ou saída, com os nomes porta B, C e D além de 2 pinos de conversor analógico / digital. Todos os pinos da porta C podem ser configurados para serem entradas analógicas também.

Veja abaixo a pinagem do BASIC Step M8



A primeira coisa que temos que fazer é baixar o compilador no site da Tato e instalar. Após instalado, configure-o para o BASIC Step M8 e linguagem BASIC no menu ferramentas. Após isto você poderá digitar todos os exemplos apresentados aqui.

Agora que o programa está instalado vamos começar a escrever os programas.

Nosso primeiro programa vai demonstrar como controlar a direção de um pino em uma porta e como ler e escrever dados em um bit de cada vez.

Para este exercício nós vamos querer ler o bit 2 da porta D e escrever o resultado no bit 3 da porta B. Se nós colocarmos um botão no bit 2 da porta D e um LED no bit 3 da porta B poderemos controlar o acendimento o LED com o botão.

Nós definimos a direção dos pinos do processador com dois comandos simples.

Conhecendo o BASIC Step M8 – parte 1

```
MAKEIN D, 2  
MAKEOUT B, 3
```

Nós queremos que o programa leia o bit de entrada e escreva o resultado no bit de saída e repita esta operação para sempre. Isto é feito com a estrutura DO ... LOOP.

```
DO  
...  
LOOP
```

Os pontos entre o DO e o LOOP no exemplo acima representam o código que queremos que sejam executados em cada repetição, os quais nós ainda não escrevemos.

Dentro deste loop nós queremos ler o bit de entrada e escrever o bit de saída. Para ler um bit nós usamos o comando INBIT. Neste exemplo nós iremos ler o bit D,2 em uma variável que chamaremos "temp". Existem diversos de variáveis disponíveis para uso. A mais simples é a variável sem nenhum sufixo, como temp, que declara que a variável será armazenada em um registrador. (Nós vamos trabalhar com outros tipos de variáveis em outra ocasião).

Após ler o bit de entrada, teremos que tomar uma decisão para ver se ligamos a saída (colocamos em 1) ou a desligamos (colocamos em 0). Para isto usaremos a estrutura IF ... THEN ... ELSE ... ENDIF. Esta estrutura é muito parecida com a sua equivalente em diversas variações da linguagem BASIC, com algumas diferenças.

Primeiro, nós podemos usar somente um teste em cada estrutura. As possibilidades são "<" (menor que), ">" (maior que), "=" (igual) e "!=" (diferente). Iremos usar a condição "diferente" porque estamos interessados em quando a entrada é 0 ou diferente de 0.


Depois nós iremos ligar o bit de saída (colocando em 1) ou desligá-lo (colocando em 0) dependendo do estado do bit de entrada. Isto é feito com os comandos SETBIT e CLRBIT.

Agora o nosso programa se parece com este.

```
MAKEIN D, 2  
MAKEOUT B, 3  
DO  
    INBIT temp, D, 2  
    IF temp != 0 THEN  
        SETBIT B, 3  
    ELSE  
        CLRBIT B, 3  
    END IF  
LOOP
```

É isto. Este é o programa inteiro. Digite-o no Compilador BASIC Step e salve-o com o nome UMBIT.BAS

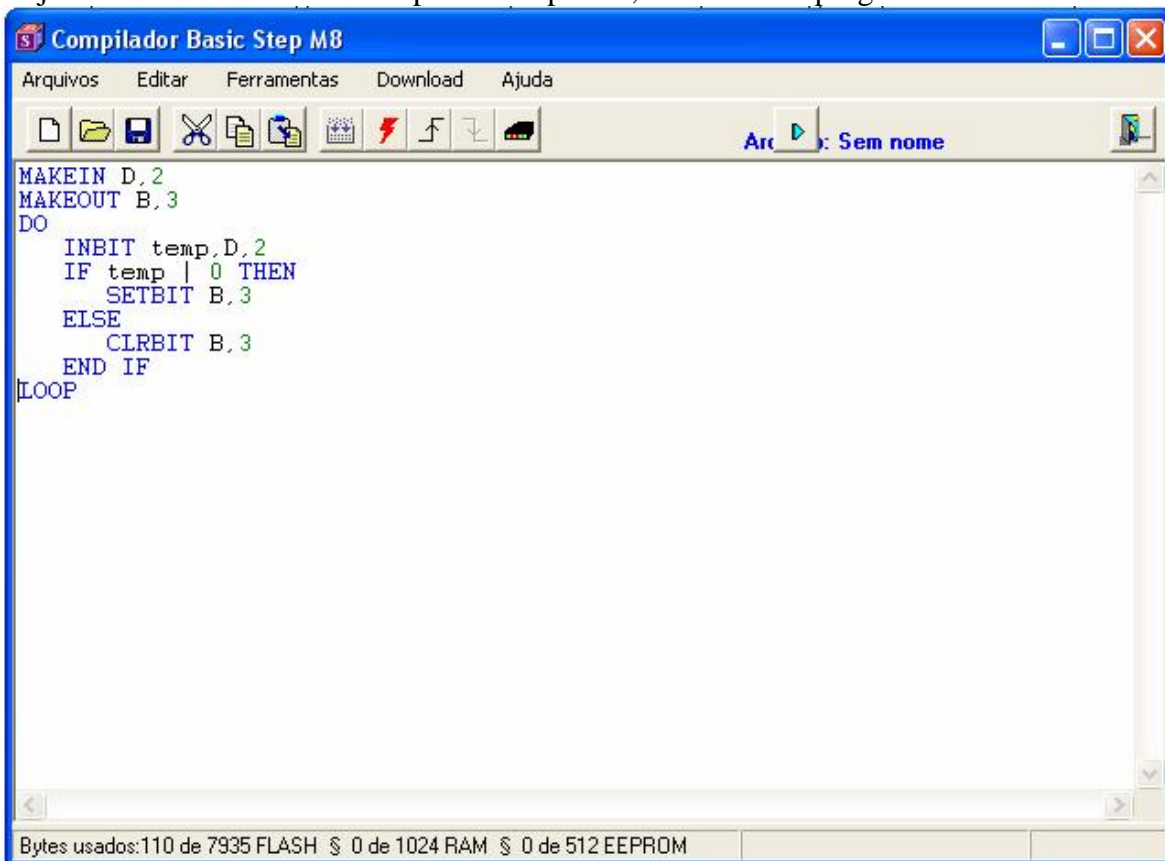
Conhecendo o BASIC Step M8 – parte 1



Agora vamos compilar o programa e ver as informações. Para compilar aperte o botão  na barra de botões na parte superior do compilador. Se você tiver digitado o programa corretamente, a barra de status na parte inferior do compilador vai mostrar as seguintes informações:

Bytes usados:110 de 7935 Flash § 0 de 1024 RAM § 0 de 512 EEPROM

Estas informações nos dizem que usamos 110 byte de um total de 7935 bytes da memória de programa, nenhum byte da memória RAM (memória para as variáveis) e nenhum byte da memória não volátil.

Veja abaixo como a tela do compilador se parece, com o nosso programa.



Após a compilação ter sido feita sem erros, podemos gravar o programa no processador clicando no botão . O programa será gravado mas ainda não executará pois o processador está em reset. Para liberar o reset e deixar o programa rodar devemos clicar no botão .

Agora vamos melhorar um pouco o nosso programa fazendo com que os bits de entrada e saída tenham nomes mais fáceis de lembrar e de alterar. Esta alteração demonstra o uso do comando EQU.

Conhecendo o BASIC Step M8 – parte 1

```
EQU "D,2","INBIT"
EQU "B,3","OUTBIT"
MAKEIN "INBIT"
MAKEOUT "OUTBIT"
DO
    INBIT temp,"INBIT"
    IF temp | 0 THEN
        SETBIT "OUTBIT"
    ELSE
        CLRBIT "OUTBIT"
    END IF
LOOP
```

Esta versão do programa compila e funciona exatamente do mesmo modo que o anterior. Não há nenhuma perda de eficiência com o uso do comando EQU. A vantagem de usar o comando EQU é que você precisa mudar apenas um lugar do programa quando quiser mudar o bit de entrada ou de saída.

Uma última modificação em nosso programa será acrescentar comentários. Os comentários são reconhecidos no BASIC quando começam com apóstrofe. O seu programa agora deve estar como este:

```
`-----`
` Programa para ler bit -
` e controlar uma saída -
`-----`

` bit de entrada
EQU "D,2","INBIT"

` bit de saída
EQU "B,3","OUTBIT"

`... define a direção dos bits ...
MAKEIN "INBIT"
MAKEOUT "OUTBIT"

`... programa principal ...
DO
    INBIT temp,"INBIT"
    IF temp | 0 THEN
        SETBIT "OUTBIT"
    ELSE
        CLRBIT "OUTBIT"
    END IF
LOOP
```

Conhecendo o BASIC Step M8 – parte 1

Tente executar este programa, se você usar a placa Super StepLab pode usar um LED da placa para indicar a saída e um dos botões do teclado para a entrada. Você pode alterar qual bit será entrada e qual será saída apenas mudando a linha EQU.

Vamos modificar este programa novamente para ele ler todos os 8 bits da porta D e escrevê-los na porta B. Nós iremos usar três novos comandos, DIRPORT, INPORT e OUTPORT. O comando DIRPORT apenas seta a direção de uma porta inteira. O INPORT lê todos os bits de uma porta e armazena em uma variável byte. O comando OUTPORT escreve os bits de uma variável byte em uma porta. Você pode ler no arquivo de ajuda do compilador BASIC Step uma descrição mais detalhada destes comandos. O comando DIRPORT pode ser usado para definir a direção de cada bit de uma porta.

```
`-----`
` Programa para ler porta-
` Copia o byte para uma  -
` porta de saída.       -
`-----`

DIRPORT D,IN
DIRPORT B,OUT

`... programa principal ...
DO
    INPORT temp,D
    OUTPORT B,temp
LOOP
```

Agora, depois de compilar e executar este programa, você vai ver que qualquer botão que for apertado na porta D fará acender o LED correspondente na porta B. Você pode ver também que o tamanho do programa diminuiu.

Você pode ver que nós utilizamos um estilo para escrever nosso programa. Todos os comandos são escritos em maiúsculas e todas as variáveis em minúsculas. Isto faz com que o programa seja mais fácil de se ler.

No próximo artigo nós vamos ver como controlar tempo com o BASIC Step M8.