



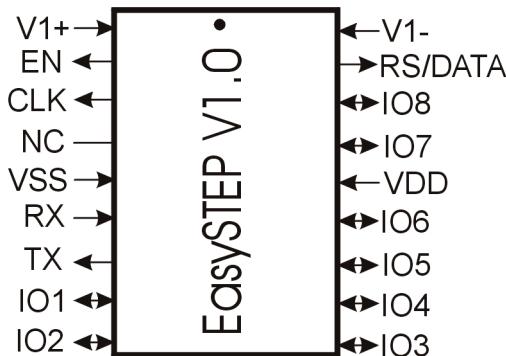
**www.cerne-tec.com.br  
(21) 4063-9798  
(11) 4063-1877  
Skype: cerne-tec  
MSN: cerne-tec@hotmail.com**



## **Manual de Comandos e Funções da EasyStep Versão 1.0**

## Introdução

A EasyStep é um microcontrolador programado através da linguagem BASIC utilizando o compilador AutoEASY. Com uma alta curva de aprendizagem, este microcontrolador pode ser utilizado em várias aplicações na área de robótica, domótica, automação e etc. A pinagem deste microcontrolador é a seguinte:



Vejamos a nomenclatura utilizada:

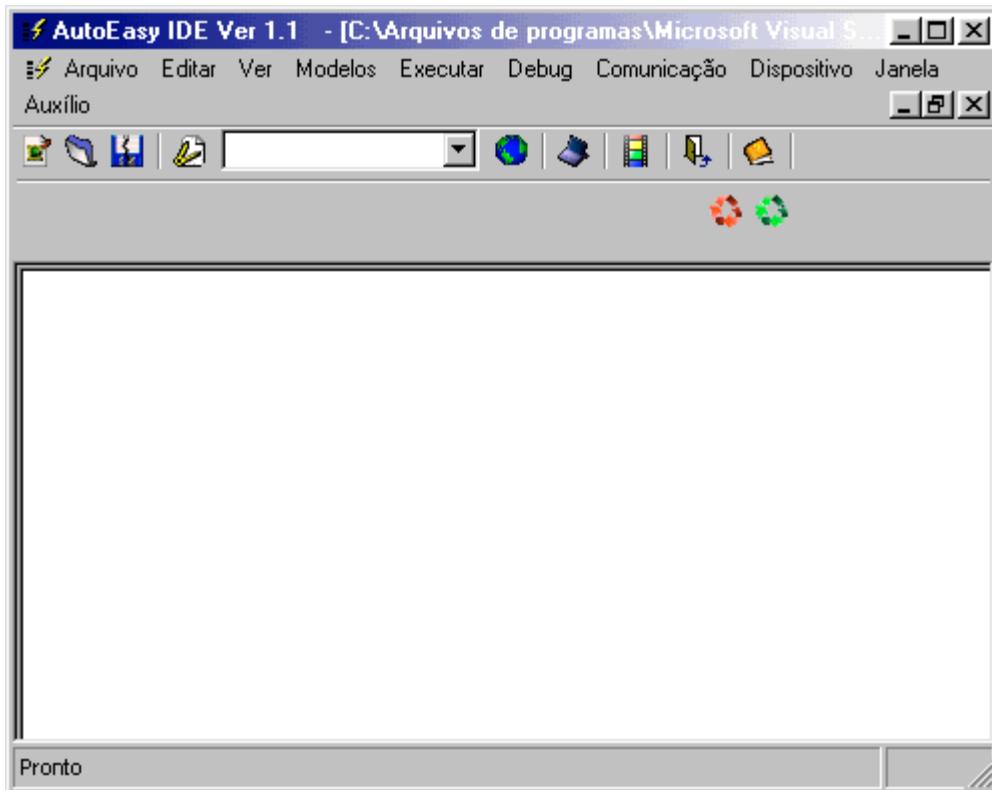
Pino	Descrição	Função
18	V1-	Entrada negativa comparador
1	V1+	Entrada positiva do comparador
2	EN	Pino de geração de clock para o display LCD
3	CLK	Pino de geração de clock para o shift register
17	RS/DATA	Pino de seleção para o display ou dados para o shift register
4	NC	Não conectado
5	VSS	Ligaçāo negativa da fonte
14	VDD	Ligaçāo positiva da fonte
8	IO1	Pino de entrada ou saída 1
9	IO2	Pino de entrada ou saída 2
10	IO3	Pino de entrada ou saída 3
11	IO4	Pino de entrada ou saída 4
12	IO5	Pino de entrada ou saída 5
13	IO6	Pino de entrada ou saída 6
15	IO7	Pino de entrada ou saída 7
16	IO8	Pino de entrada ou saída 8
6	RX	Pino de Recepção serial

7	TX	Pino de Transmissão serial
---	----	----------------------------

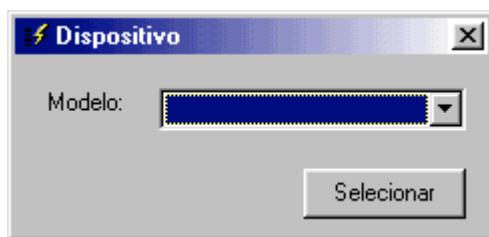
Este microcontrolador funciona com uma tensão de 3V até 5,5 V e a uma temperatura de 0 a 70ºC.

## Ambiente de Programação

O software utilizado para desenvolver as aplicações no microcontrolador é a AutoEasy. Este software pode ser baixado gratuitamente no site da Cerne Tecnologia ([www.cerne-tec.com.br](http://www.cerne-tec.com.br)) porém o mesmo está incluído neste CD. Após a instalação e inicialização, veremos a seguinte tela:



A primeira ação que devemos ter é selecionar o dispositivo utilizado para o desenvolvimento de projetos, sendo neste caso a EasySTEP. Para isso vá no menu *Dispositivo*. A seguinte tela surgirá:

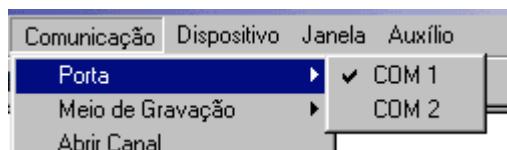


Clique em modelo e escolha o microcontrolador EasySTEP. De volta a área principal de programação, para escrevermos nossos códigos basta inserir estes

nesta área e após este procedimento salvar o mesmo através de *Arquivo -> Salvar*. Com o código escrito e salvo, basta iniciar a compilação para gerar o arquivo no qual a EasySTEP é capaz de entender. Pressione F10 ou vá em *Executar -> Compilar* para que se inicie o processo de compilação. Caso haja algum erro neste processo, o compilador irá reportar o mesmo a você para que o mesmo seja sanado.

De posse do arquivo compilado, para transferir o este é necessário utilizar uma porta de comunicação RS-232. Estas portas comumente são encontradas através dos conectores DB9 fêmea e ficam atrás de um PC. Neste momento, o cabo de gravação deve estar conectado entre a porta e a EasySTEP de forma a permitir a comunicação além de a EasySTEP estar ligada em uma fonte de alimentação.

Para abrir a porta de comunicação serial, vá no menu *Comunicação -> Porta*. Teremos a seguinte situação:



Selecione a porta que estiver disponível no seu PC para comunicação. Após esta tarefa, clique em *Abrir Canal* que está neste mesmo menu. Agora clique em *Transmite Programa* para que o programa seja transferido para a EasySTEP. Outra opção bastante interessante para que fique marcada é a opção *Gravar após compilação*. Neste caso, sempre que uma compilação for bem sucedida, a AutoEasy iniciará automaticamente a transferência do arquivo compilado para a EasySTEP. Caso haja algum erro de comunicação, o programa reportará a você.

## Configuração do Ambiente

Para que seu projeto funcione, você precisará da seguinte estrutura:



No PC ficará instalada a AutoEasy



Através do cabo serial você irá transmitir o programa do PC para a EasySTEP



O cabo serial fica conectado nos pinos de TX, RX e VSS fazendo desta forma a comunicação entre o PC e a AutoEasy. Note que este cabo somente é necessário no momento da gravação do microcontrolador. No pino de RX deve se ter conectado um resistor de 2K2. Verifique isto no esquema em anexo.

## Comentários

Um comentário na AutoEasy inicia com ; (ponto e vírgula). Um comentário não é interpretado pelo compilador e pode ser usado sem problemas para melhorar o entendimento pelo software. Vejamos um exemplo:

high 4

;liga a saída 4

;veja que todo a frase iniciada com ; não é interpretado  
;pelo compilador

## Representações numéricas

Existem três formas de representarmos um número, sendo estas a binária, hexadecimal e decimal. Vejamos como proceder para esta operação:

%valor em binário

0X valor em hexadecimal

valor em decimal

Veja que quando você querer trabalhar com constantes binárias, basta iniciar com o sinal %. Veja o exemplo abaixo em que usado o comando shift out:

shift out(%00110011)

Já a representação em hexa inicia com 0X. Abaixo uma atribuição a uma variável em hexadecimal:

b1=0X10

E finalmente as constantes em decimal:

b2=100

# Comandos e Funções da Linguagem

---

## DIRPIN

### **Ação:**

Ajusta a direção dos pinos de I/O, ou seja, se os mesmos serão de entrada ou saída.

### **Descrição:**

Ao todo, são disponibilizadas oito portas de comunicação com o mundo externo ao EasySTEP. Dependendo do dispositivo que você for ligar em um dos pinos, o mesmo deve ser configurado como entrada ou saída. A configuração obedece a seguinte sintaxe:

DIRPIN=% IO8 IO7 IO6 IO5 IO4 IO3 IO2 IO1 IO0

Onde caso o bit referente ao teste estiver em 1, significa que o mesmo está configurado como entrada e caso esteja em 0, como saída.

### **Exemplo:**

Digamos que desejemos configurar os pinos IO8, IO7, IO6 e IO5 como entradas e o restante como saída, teríamos o seguinte resultado:

dirpin=%11110000 ;Configura os IOS 8, 7, 6 e 5 como entrada  
;e o restante como saída.

---

## IOS

### **Ação:**

Impõe um nível lógico na saída dos I/Os.

### **Descrição:**

Para impormos um nível lógico em na saída do port do microcontrolador, devemos utilizar este registrador para esta tarefa. A organização dele é a seguinte:

IOS=% IO8 IO7 IO6 IO5 IO4 IO3 IO2 IO1 IO0

Onde caso o bit referente ao teste estiver em 1, significa que o pino ficará em nível alto e caso esteja em 0, ficará em nível baixo.

### ***Exemplo:***

Vamos imaginar que existem oito leds conectados em todo o port de I/O e que o programa queira deixar todos estes acesos, sabendo que eles são acionados em nível alto.

```
dirpin=%00000000 ;Configura todos os pinos como saída  
ios=%11111111 ;Impõe nível alto em todas as saídas
```

---

## **DELAY\_MS**

### ***Ação:***

Gera um retardo no programa em ms.

### ***Descrição:***

Imagine que você vá desenvolver um pisca-pisca. Neste caso, uma rotina de retardo será de suma importância. A linguagem disponibiliza uma rotina de tempo, onde o usuário passa o tempo em ms para ela no intervalo de 1 a 255 para que a mesma possa aguardar um tempo.

### ***Exemplo:***

```
dirpin=%00000000 ;Configura todos os pinos como saída  
novamente:  
    ios=%11111111 ;Liga todas as saídas  
    delay_ms(200) ;Aguarda 200 ms  
    ios=%00000000 ;Desliga todas as saídas  
    delay_ms(200) ;Aguarda 200 ms  
    goto novamente ;Volta para novamente
```

---

## **DELAY\_SEG**

### ***Ação:***

Gera um retardo no programa em segundos.

### ***Descrição:***

Imagine que você vá desenvolver um pisca-pisca. Neste caso, uma rotina de retardo será de suma importância. A linguagem disponibiliza uma rotina de tempo, onde o usuário passa o tempo em seg para ela no intervalo de 1 a 255 para que a mesma possa aguardar um tempo.

### ***Exemplo:***

dirpin=%0000000 ;Configura todos os pinos como saída  
novamente:  
ios=%11111111 ;Liga todas as saídas  
delay\_seg(1) ;Aguarda 1 s  
ios=%00000000 ;Desliga todas as saídas  
delay\_seg(1) ;Aguarda 1 s  
goto novamente ;Volta para novamente

---

## TOGGLE

**Ação:**

Inverte o estado das saídas do microcontrolador EasySTEP.

**Descrição:**

Inverte o estado das saídas. Quem está em nível alto fica em baixo e vice-versa.

**Exemplo:**

dirpin=%0000000 ;Configura todos os pinos como saída  
novamente:  
toggle ;Inverte o estado de todos os pinos  
delay\_ms(200) ;Aguarda 200 ms  
toggle ;Inverte o estado de todos os pinos  
delay\_ms(200) ;Aguarda 200 ms  
goto novamente ;Volta para novamente

---

## GOTO

**Ação:**

Salta para um ponto específico do programa.

**Descrição:**

O goto (vá para) é o salto incondicional do programa e serve para alterar o fluxo do programa.

**Exemplo:**

dirpin=%0000000 ;Configura todos os pinos como saída

novamente:

goto novamente ;Entra em loop

---

## GOSUB

### Ação:

Executa uma rotina.

### Exemplo:

```
dirpin=0 ;configura os pinos
gosub aguarda_1Seg ;chama a rotina para aguardar 1 segundo
end ;finaliza o programa

aguarda_1Seg:
    delay_Seg(1) ;aguarda 1 segundo
    return ;retorna
```

---

## RETURN

### Ação:

Retorna de uma sub-rotina.

### Exemplo:

```
dirpin=0 ;configura os pinos
gosub aguarda_1Seg ;chama a rotina para aguardar 1 segundo
end ;finaliza o programa

aguarda_1Seg:
    delay_Seg(1) ;aguarda 1 segundo
    return ;retorna
```

---

## HIGH

### Ação:

Impõe nível lógico alto em uma das saídas do microcontrolador.

### Descrição:

Eleva o nível em alguma das saídas do microcontrolador. É importante observar que o pino já deve estar configurado como saída para que este comando funcione.

A sintaxe deste comando é a seguinte:

high pino

Onde pino pode variar de 1 a 8 dependendo do pino a ser acionado.

**Exemplo:**

```
dirpin=%0000000          ;Configura todos os pinos como saída  
high 1                  ;liga a saída 1
```

---

## LOW

**Ação:**

Impõe nível lógico baixo em uma das saídas do microcontrolador.

**Descrição:**

Abaixa o nível em alguma das saídas do microcontrolador. É importante observar que o pino já deve estar configurado como saída para que este comando funcione.

A sintaxe deste comando é a seguinte:

low pino

Onde pino pode variar de 1 a 8 dependendo do pino a ser acionado.

**Exemplo:**

```
dirpin=%0000000          ;Configura todos os pinos como saída  
low 8                   ;desliga a saída 1
```

---

## SHIFT OUT

**Ação:**

Transferi para o shift register um dado.

**Descrição:**

Faz a comunicação com o shift register passando para este um dado de 8 bits.

A sintaxe deste comando é a seguinte:

shift out (dado)

Onde dado pode ser representado de forma binária, hexadecimal ou decimal.

Os pinos utilizados para comunicação com o shift register são os pinos CLK e RS/DATA .

**Exemplo:**

shift register (%01010000)	;envia sequência binária para o shift ;register
----------------------------	--

---

## PWM

**Ação:**

Permite alterar o ciclo ativo do PWM.

**Descrição:**

O PWM da AutoEasy funciona na freqüência de 1kHz. Cada período é de 1ms dividido em 100 partes. Cada parte equivale a aproximadamente a 10 us e quando a função PWM é chamada, deve se passar para esta um valor entre 0 e 100 de forma a controlar a potência da carga externa.

A sintaxe é a seguinte:

pwm (ciclo ativo)

Onde ciclo ativo varia de 0 a 100 %.

O pino do microcontrolador que é utilizado para PWM é o pino I/O2.

**Exemplo:**

pwm (100)	;PWM com ciclo ativo máximo
delay_ms(200)	;Aguarda 200 ms
pwm(50)	;PWM com ciclo ativo pela metade

---

## INC PWM

**Ação:**

Incrementa o ciclo ativo do PWM.

**Exemplo:**

pwm (0)	;PWM com ciclo ativo mínimo
delay_ms(200)	;Aguarda 200 ms
inc(pwm)	;Incrementa o PWM

---

## DEC PWM

**Ação:**

Decrementa o ciclo ativo do PWM.

**Exemplo:**

pwm (100)	;PWM com ciclo ativo máximo
delay_ms(200)	;Aguarda 200 ms
dec(pwm)	;Decrementa o PWM

---

## TXDATA

**Ação:**

Transferi uma seqüência de caracteres a 9600 bps via canal serial de comunicação.

**Descrição:**

Uma das aplicações mais poderosas que um microcontrolador pode disponibilizar é a parte de comunicação serial entre ele e um dispositivo externo como por exemplo um PC. A EasySTEP permite esta comunicação a uma taxa de 9600 bps através do comando TXDATA que tem a seguinte sintaxe:

TXDATA(sequência de caracteres)

**Exemplo:**

TXDATA(EasySTEP, o microcontrolador do futuro!)  
;transfere uma seqüência de caracteres para o PC.

---

## IF RXDATA

**Ação:**

Testa o buffer de recepção serial.

**Descrição:**

Verifica se há algum caracter no buffer de recepção serial e se o conteúdo do mesmo.

```
if rxdata=caracter then
```

Onde caracter é algum caracter printável.

**Exemplo:**

```
dirpin=%00000001 ;configura a direção dos pinos
```

repete:

```
if rxdata="A" then ;se recebeu o caracter "A" ...
    high 4 ;liga a saída 4
end if ;fim do se
```

```
if rxdata="a" then ;se recebeu o caracter "a" ...
    low 4 ;desliga a saída 4
end if ;fim do se
```

```
goto repete ;volta para repetir o teste
```

---

## ROTATE LEFT

**Ação:**

Rotaciona o conteúdo de uma variável byte ou word para esquerda.

**Descrição:**

Rotaciona o conteúdo de uma variável para a esquerda e salva o conteúdo na própria variável. A sintaxe deste comando é o seguinte:

```
rotate left (var, número de rotações)
```

Onde var pode ser de b1 a b8 ou de w1 a w8. O campo número de rotações especifica quantas rotações serão feitas na variável.

**Exemplo:**

```
b1=1 ;Inicializa b1 com 1
rotate left (b1,2) ;Rotaciona duas vezes esta variável, ficando com o
                    ;resultado 4
```

---

## ROTATE RIGHT

**Ação:**

Rotaciona o conteúdo de uma variável byte ou word para direita.

**Descrição:**

Rotaciona o conteúdo de uma variável para a direita e salva o conteúdo na própria variável. A sintaxe deste comando é o seguinte:

rotate right (var, número de rotações)

Onde var pode ser de b1 a b8 ou de w1 a w8. O campo número de rotações especifica quantas rotações serão feitas na variável.

**Exemplo:**

```
b1=128      ;Inicializa b1 com 128
rotate right (b1,8) ;Rotaciona duas vezes esta variável, ficando com o
;resultado 1
```

---

## IF COMPARATOR

**Ação:**

Testa a entrada do comparador para saber o estado do mesmo.

**Descrição:**

Este microcontrolador disponibiliza uma entrada para comparador nos pinos 18 e 1. Quando a tensão aplicada no pino 18 é maior que a do pino 1, este teste fica verdadeiro e caso contrário falso.

**Exemplo:**

```
if comparator then      ;se o comparador for verdadeiro...
    high 4              ;liga a saída 4
else                      ;senão...
    low 4               ;desliga a saída 4
end if                     ;fim do se
```

---

## IF IO

**Ação:**

Testa a entrada dos los para saber se os mesmos estão verdadeiros ou falsos.

**Descrição:**

Testa uma das oito entradas disponíveis no microcontrolador EasySTEP. As entradas ficam verdadeiras em lógica negativa, ou seja, quando a entrada está em nível lógico alto, o teste é falso e caso esteja em nível baixo, o teste é verdadeiro.

A sintaxe deste comando é a seguinte:

```
if io=nível then
```

Onde nível pode ser 0 ou 1 referente ao teste lógico do pino e não físico.

**Exemplo:**

```
dirpin=%00000001 ;configura a direção dos pinos
```

repete:

```
if io1=1 then ;se a entrada 1 for verdadeira...
    high 4 ;liga a saída 4
else ;senão...
    low 4 ;desliga a saída 4
end if ;fim do se
goto repete ;volta para repetir o teste
```

---

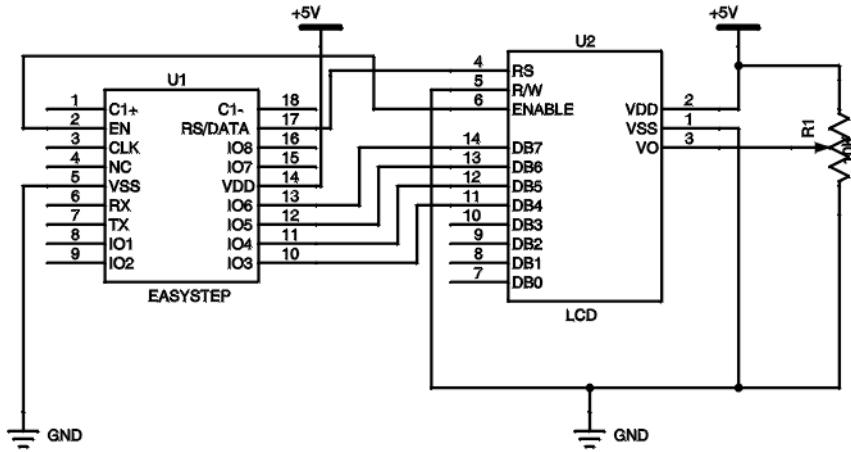
## INIC DISPLAY

**Ação:**

Inicializa o display lcd 16x2 .

**Descrição:**

Para que o display LCD 16x2 funcione, é necessário seguir o seguinte esquema de hardware:



Esta função tem a incumbência de inicializar o display para que o comando display possa escrever os dados nele. Os pinos IO6, IO5, IO4 e IO3 devem estar previamente configurados como saída.

### ***Exemplo:***

```
dirpin=%00000000 ;configura os pinos como saída  
inic display ;inicializa o display
```

## DISPLAY

## **Ação:**

Escreve dados no display LCD.

### **Descrição:**

Permite escrever constantes ou variáveis no display LCD. A sintaxe deste comando é a seguinte:

`display(número da linha, número da coluna, constante ou variável)`

Em número da linha é informada a linha onde os dados serão apresentados, podendo ser neste caso 1 ou 2. Já em coluna, é escolhida a coluna onde os dados serão apresentados podendo variar de 1 a 16 e finalmente em constantes os caracteres a serem apresentados ou as variáveis byte ou word.

### *Exemplo:*

```
dirpin=%00000000 ;configura a direção dos pinos  
inic display ;inicializa o display  
display(1,1,Teste) ;mostra na linha 1 e coluna 1 a  
;constante “Teste”  
  
b1=10 ;apresenta na linha 2 o valor da  
display(2,1,b1) ;variável b1
```

---

## CLD

**Ação:**

Limpa o display LCD.

**Descrição:**

Limpa todo o conteúdo do display LCD.

**Exemplo:**

```
dirpin=%00000000 ;configura a direção dos pinos  
inic display ;inicializa o display  
display(1,1,Teste) ;mostra na linha 1 e coluna 1 a  
;constante “Teste”  
  
delay_seg(2); ;aguarda 2 segundos  
cld ;limpa o display  
b1=100 ;inicializa a variável b1  
display(2,1,b1) ;apresenta na linha 2 o valor da  
;variável b1
```

---

## ROTATE DISPLAY LEFT

**Ação:**

Rotaciona os caracteres do display para a esquerda.

**Descrição:**

Rotaciona o conteúdo do display para a esquerda aguardando um intervalo de tempo entre uma rotação e outra.

A sintaxe deste comando é a seguinte:

rotate display left (número de rotações, tempo entre rotações)

Em número de rotações é informado quantas vezes o display será rotacionado para a esquerda e em tempo entre rotações o intervalo de tempo entre uma rotação e outra.

**Exemplo:**

```
dirpin=%00000000 ;configura a direção dos pinos  
inic display ;inicializa o display  
display(1,10,Teste) ;mostra na linha 1 e coluna 10 a  
;constante “Teste”  
  
rotate display left (5,255) ;rotaciona para a esquerda o  
;display 5 vezes durante o intervalo  
;de tempo de 255 ms
```

---

## ROTATE DISPLAY RIGHT

**Ação:**

Rotaciona os caracteres do display para a direita.

**Descrição:**

Rotaciona o conteúdo do display para a direita aguardando um intervalo de tempo entre uma rotação e outra.

A sintaxe deste comando é a seguinte:

rotate display right (número de rotações, tempo entre rotações)

Em número de rotações é informado quantas vezes o display será rotacionado para a direita e em tempo entre rotações o intervalo de tempo entre uma rotação e outra.

**Exemplo:**

```
dirpin=%00000000 ;configura a direção dos pinos  
inic display ;inicializa o display  
display(1,10,Teste) ;mostra na linha 1 e coluna 10 a  
;constante “Teste”  
  
rotate display right (5,255) ;rotaciona para a direita o
```

;display 5 vezes durante o intervalo  
;de tempo de 255 ms

---

## WAITIO

### **Ação:**

Aguarda até que um pino de I/O alcance o nível lógico especificado.

### **Descrição:**

Este comando é utilizado para aguardar até que um determinado nível lógico seja atingido em um pino do microcontrolador. A sintaxe dele é a seguinte:

waitiox(nível)

Onde x pode variar de 1 a 8 dependendo do pino de I/O e nível pode ser 1 ou 0 aguardando até que este nível seja atingido.

### **Exemplo:**

dirpin=%0000001	;Configura todos os pinos como saída
novamente:	
waitio1(1)	;aguarda até que o pino fique em nível
	;lógico verdadeiro
high 6	;liga o pino 6
delay_ms(200)	;aguarda 200 ms
low 6	;desliga o pino 6
goto novamente	;volta para novamente

---

## INC

### **Ação:**

Incrementa o conteúdo de uma variável.

### **Descrição:**

O ato de incrementar consiste em somar 1 a alguma variável do tipo byte ou word. A sintaxe é a seguinte:

inc(var)

Onde var pode variar de b1 a b8 ou de w1 a w8 dependendo do tipo de variável utilizada.

### ***Exemplo:***

```
b1=10          ;inicializa a variável b1 com 10  
inc(b1)        ;incrementa o conteúdo de b1 tendo como resultado 11
```

---

## **DEC**

### ***Ação:***

Decrementa o conteúdo de uma variável.

### ***Descrição:***

O ato de decrementar consiste em subtrair 1 a alguma variável do tipo byte ou word. A sintaxe é a seguinte:

```
dec(var)
```

Onde var pode variar de b1 a b8 ou de w1 a w8 dependendo do tipo de variável utilizada.

### ***Exemplo:***

```
b1=10          ;inicializa a variável b1 com 10  
dec(b1)        ;decrementa o conteúdo de b1 tendo como resultado 9
```

---

## **NOT**

### ***Ação:***

Inverte o estado dos bits de uma variável.

### ***Descrição:***

Os bits que estão em nível 1 de uma variável ficarão em nível baixo e vice-versa. A sintaxe é a seguinte:

```
var= not var
```

Onde var pode ser de b1 a b8 referente as variáveis do tipo byte ou de w1 a w8 referente as variáveis do tipo word .

**Exemplo:**

b1=0	;inicializa a variável b1 com 0
b1=not b1	;b1 recebe a negação dele, neste caso 255
w1=65535	;inicializa a variável b1 com 65535
w1=not w1	;w1 recebe a negação dele, neste caso 0

---

## AND

**Ação:**

Operação lógica & entre variáveis byte ou word.

**Descrição:**

Executa uma operação & entre variáveis. Veja a sintaxe:

var1 = var1 and var2

Onde var1 e var2 podem ser de b1 a b8 referente as variáveis do tipo byte ou de w1 a w8 referente as variáveis do tipo word .

**Exemplo:**

b1=15	;inicializa a variável b1 com 15
b2=0	;inicializa a variável b2 com 0
b1=b1 and b2	;lógica & entre b1 e b2

---

## OR

**Ação:**

Operação lógica OR entre variáveis byte ou word.

**Descrição:**

Executa uma operação OR entre variáveis. Veja a sintaxe:

var1 = var1 OR var2

Onde var1 e var2 podem ser de b1 a b8 referente as variáveis do tipo byte ou de w1 a w8 referente as variáveis do tipo word .

**Exemplo:**

```
b1=15          ;inicializa a variável b1 com 15  
b2=0          ;inicializa a variável b2 com 0  
b1=b1 or b2   ;lógica or entre b1 e b2
```

---

## PULSOUT

**Ação:**

Gera pulsos de saída no microcontrolador.

**Descrição:**

Executa uma operação de saída de pulsos no microcontrolador.

pulsout (tempo dos pulsos, número de repetições)

Onde tempo dos pulsos pode variar de 1 a 255 assim como número de repetições. A base de tempo para tempo dos pulsos é de 1 ms. O pino de saída de pulsos é o I/O7.

**Exemplo:**

```
dirpin=0          ;configura os pinos como saída  
pulsout(5,5)     ;gera 5 pulsos com intervalo de 1 ms entre cada  
                  ;pulso
```

---

## PULSIN

**Ação:**

Lê uma variação externa.

**Descrição:**

Lê um pulso externo na base de tempo de 1 segundo. O pino utilizado para este fim é o I/O8. A sintaxe é a seguinte:

pulsin(estado do pino, variável)

Onde estado do pino informa em que nível o pulsos deverá começar a ser medido, sendo neste caso 1 ou 0. Já em variável , é informado a variável que

receberá a medição do pulso, devendo esta ser uma variável do tipo word de w1 a w8.

**Exemplo:**

```
dirpin=%00000010      ;configura os pinos de I/O  
pulsin(1,w1)          ;mede a duração do pulso no pino I/O1 e guarda  
                      ;o resultado em w1
```

---

## INPUT

**Ação:**

Transforma um determinado pino em entrada.

**Descrição:**

Transforma um dos I/Os em entrada. A sintaxe deste comando é a seguinte:

input pino

Onde pino pode variar de 1 a 8 dependendo do pino do microcontrolador.

**Exemplo:**

```
input 1                ;configura o pino 1 como entrada  
input 5                ;configura o pino 5 como entrada
```

---

## OUTPUT

**Ação:**

Transforma um determinado pino em saída.

**Descrição:**

Transforma um dos I/Os em saída. A sintaxe deste comando é a seguinte:

output pino

Onde pino pode variar de 1 a 8 dependendo do pino do microcontrolador.

**Exemplo:**

```
output 1              ;configura o pino 1 como saída
```

```
output 5 ;configura o pino 5 como saída
```

---

## REVERSE

**Ação:**

Inverte a direção de um pino.

**Descrição:**

Caso algum pino esteja configurado como entrada, passa a ser saída e vice-versa. A sintaxe deste comando é a seguinte:

```
reverse pino
```

Onde pino pode variar de 1 a 8 dependendo do pino do microcontrolador.

**Exemplo:**

```
output 1 ;configura o pino 1 como saída  
reverse 1 ;configura o pino 1 como entrada já que inverteu  
;a sua direção
```

---

## READ\_EEPROM

**Ação:**

Lê um valor da memória EEPROM do microcontrolador.

**Descrição:**

Existe 1 byte de memória EEPROM que pode ser utilizado pelo desenvolvedor do sistema. Para ler este byte, siga a seguinte sintaxe:

```
var=read_eeprom
```

Onde var pode variar de b1 a b8.

**Exemplo:**

```
b4=read_eeprom
```

```
if b1=50 then  
    high 8  
else  
    low 8
```

end if

---

## WRITE\_EEPROM

**Ação:**

Escreve um valor da memória EEPROM do microcontrolador.

**Descrição:**

Escreve um byte na memória EEPROM do microcontrolador. Abaixo a sintaxe:

write\_eeprom(var)

Onde var pode variar de b1 a b8.

**Exemplo:**

b4=100	;inicializa b4 com o valor 100
write_eeprom(b4)	;escreve o conteúdo da variável na ;memória EEPROM

---

## RANDOM

**Ação:**

Retorna um valor randômico em uma variável do tipo word.

**Descrição:**

Retorna um valor randômico em uma variável do tipo word. A sintaxe é a seguinte:

var= rnd

Onde var pode variar de w1 a w8.

**Exemplo:**

dirpin=0	;configura a direção do pino
inic display	;inicializa o display
repete:	
w1=rnd	;w1 recebe o valor randômico
display(1,1,w1)	;apresenta na linha e coluna 1 o valor de w1
delay_ms(255)	;aguarda 255 ms
goto repete	;salta para repete

---

## FOR...NEXT

**Ação:**

Repete um bloco determinado número de vezes.

**Descrição:**

Com o FOR...NEXT você pode repetir um determinado número de vezes um bloco de comandos. A sintaxe desta estrutura é a seguinte;

```
for var= início to fim step passo
    comandos
next
```

A *var* é uma variável do tipo byte que será utilizada como controle do FOR...NEXT. O valor inicial desta variável é determinado através da constante *início* e o valor final é determinado pela constante *fim*. O passo que o valor inicial da variável será incrementado é definido através do *passo*.

**Exemplo:**

```
dirpin=0          ;configura os pinos como saída
for b1= 1 to 10 step 1 ;repete este bloco 10 vezes
    high 4        ;liga a saída
    delay_ms(200) ;aguarda 200 ms
    low 4         ;desliga a saída
    delay_ms(200) ;aguarda 200 ms
next
```

---

## WHILE...WEND

### **Ação:**

Repete um bloco determinado número de vezes.

### **Descrição:**

Com o WHILE...WEND você pode repetir um determinado número de vezes um bloco de comandos. A sintaxe desta estrutura é a seguinte;

```
while condição  
    comandos  
wend
```

Em condição é informado a condição para que o bloco se repita. Caso o teste logo no início seja falso, o bloco não é repetido. Enquanto a condição informada for verdadeira, o bloco se repetirá. A variável utilizada é do tipo byte e os operadores igual (=), maior (>) e menor (<) podem ser utilizados com esta variável.

### **Exemplo:**

```
dirpin=0          ;configura os pinos como saída  
  
while b1<10      ;enquanto b1 for menor que 10...  
    high 4        ;liga a saída  
    delay_ms(200)  ;aguarda 200 ms  
    low 4         ;desliga a saída  
    delay_ms(200)  ;aguarda 200 ms  
    inc(b1)        ;incrementa o conteúdo de b1  
wend
```

---

## DO...LOOP UNTIL

### **Ação:**

Repete um bloco determinado número de vezes até que a condição seja atendida.

### **Descrição:**

Com o DO...LOOP UNTIL você pode repetir um determinado número de vezes um bloco de comandos. A sintaxe desta estrutura é a seguinte;

```
do
    comandos
loop until condição
```

Neste caso, *comandos* repetirá até que a condição informada seja verdadeira. Caso a condição não seja verdadeira, *comandos* serão executados pelo menos 1 vez.

#### **Exemplo:**

```
dirpin=0          ;configura os pinos como saída
do
    high 4        ;liga a saída
    delay_ms(200) ;aguarda 200 ms
    low 4         ;desliga a saída
    delay_ms(200) ;aguarda 200 ms
    inc(b1)       ;incrementa o conteúdo de b1
loop until b1=10
```

## **Manipulação de Variáveis**

Ao todo, a EasySTEP suporta 16 variáveis, sendo 8 do tipo byte e 8 do tipo word. As variáveis byte possuem a faixa de 0 a 255 e as do tipo word de 0 a 65535. As variáveis byte já estão declaradas no sistema e são chamadas de b1 a b8 e as do tipo word de w1 a w8.

Vejamos algumas operações de atribuição com a variável byte e word:

```
b1= 10           ;inicializa b1 com 10
w1= 2500         ;inicializa w1 com 2500
```

Abaixo uma operação para verificar o conteúdo de uma variável:

```
b1=10           ;inicializa b1 com 10
b2=10           ;inicializa b2 com 10

if b1=b2 then   ;se b1 for igual a b2...
    high 5      ;liga a saída 5
else            ;senão...
```

```

        low 5          ;desliga a saída 5
end if           ;fim do se

```

Abaixo uma operação para verificar o conteúdo de uma variável:

```

b1=100          ;inicializa b1 com 100
b2=10          ;inicializa b2 com 10

if b1>b2 then      ;se b1 for maior que b2...
    high 5        ;liga a saída 5
else             ;senão...
    low 5         ;desliga a saída 5
end if           ;fim do se

```

Abaixo uma operação para verificar o conteúdo de uma variável:

```

b1=100          ;inicializa b1 com 100
b2=10          ;inicializa b2 com 10

if b1>b2 then      ;se b1 for maior que b2...
    high 5        ;liga a saída 5
else             ;senão...
    low 5         ;desliga a saída 5
end if           ;fim do se

```

Abaixo uma operação para verificar o conteúdo de uma variável:

```

b1=100          ;inicializa b1 com 100
b2=10          ;inicializa b2 com 10

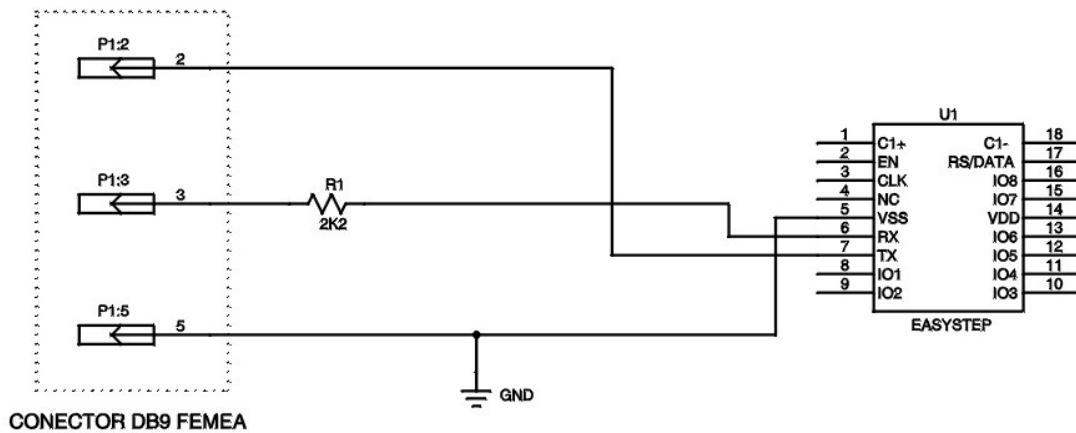
if b1<b2 then      ;se b1 for menor que b2...
    high 5        ;liga a saída 5
else             ;senão...
    low 5         ;desliga a saída 5
end if           ;fim do se

```

# Esquemas

## Cabo de Gravação

Para que o microcontrolador possa ser gravado pelo PC, é necessário a montagem de um cabo que seja montado da seguinte forma:



## Exemplos

---

### 1. Acionamento de Saídas

*;Este programa visa carregar um valor binário  
;na saída da EasyStep*

```
dirpin=%00000000      ;configura os pinos como saída  
ios=%10101010          ;aciona as saídas de io
```

---

### 2. Pisca-Pisca

*;A idéia deste programa é  
;criar um pisca-pisca em todo o  
;os pinos de I/O do microcontrolador*

```
dirpin=%00000000      ;configura a direção do pino  
  
repete_:  
  
    ios=%11111111      ;carrega um valor nos I/Os  
    delay_seg(1)        ;aguarda 1 segundo  
    ios=%00000000      ;carrega um valor nos I/Os  
    delay_seg(1)        ;aguarda 1 segundo  
    goto repete_        ;salta para repete
```

---

### 3. Pisca-Pisca II

*;Faremos com que o led pisque somente  
;10 vezes neste exemplo utilizando a estrutura  
;for ... next*

```
dirpin=%00000000      ;configura a direção dos pinos  
  
for b1=1 to 10 step 1  ;laço de 1 até 10  
    ios=%11111111      ;carrega um valor na saída de io  
    delay_seg(1)        ;aguarda 1 segundo  
    ios=%00000000      ;carrega um valor na saída de io  
    delay_seg(1)        ;aguarda 1 segundo  
next
```

#### 4. Pisca-Pisca III

```

;A intenção deste projeto
;é fazer com que o led oscile
;com utilizando para isso a estrutura
;do...loop until

dirpin=%00000000          ;configura os pinos como saída
b1=0                      ;inicializa a variável b1

do
    toggle                ;inverte o estado dos pinos de I/O
    delay_seg(1)           ;aguarda 1 segundo
    toggle                ;inverte o estado dos pinos de I/O
    delay_seg(1)           ;aguarda 1 segundo
    inc(b1)                ;incrementa a variável b1
loop until b1=10           ;laço até que b1 seja igual a 10

```

---

#### 5. Pisca-Pisca IV

```

;A intenção deste projeto
;é fazer com que o led oscile
;com utilizando para isso a estrutura
;while...wend

dirpin=%00000000          ;configura os pinos como saída
b1=0                      ;inicializa a variável b1

while b1<>10
    toggle                ;inverte o estado dos pinos de I/O
    delay_seg(1)           ;aguarda 1 segundo
    toggle                ;inverte o estado dos pinos de I/O
    delay_seg(1)           ;aguarda 1 segundo
    inc(b1)                ;incrementa a variável b1
wend

```

---

#### 6. Chamada de Rotinas

```

;A intenção deste projeto
;é demonstrar a utilização do gosub
;para oscilar um led

dirpin=%00000000          ;configura os pinos como saída

```

```

loop:
    gosub liga_led      ;chama a rotina liga_led
    gosub aguarda_tempo ;chama a rotina para aguardar um tempo
    gosub desliga_led   ;chama a rotina para desligar um led
    gosub aguarda_tempo ;chama a rotina para aguardar um tempo

    goto loop            ;volta para loop

liga_led:
    high 8              ;liga a saída 8
    return               ;retorna

desliga_led:
    low 8                ;desliga a saída 8
    return               ;retorna

aguarda_tempo:
    delay_seg(1)         ;aguarda 1 segundo
    return               ;retorna

```

---

## 7. Controle de PWM

```

;este exemplo irá apresentar a utilização da
;recepção serial assim como do controle
;de PWM através da USART

dirpin=%00000000          ;configura a direção da porta

teste:
    if rxdata=="a" then      ;se recebeu caracter...
        pwm(50)              ;configura o PWM
    end if                   ;fim do se

    if rxdata=="b" then      ;se recebeu caracter...
        pwm(100)              ;configura o PWM
    end if                   ;fim do se

    if rxdata=="c" then      ;se recebeu caracter...
        pwm(150)              ;configura o PWM

```

```

end if ;fim do se

if rxdata=="d" then ;se recebeu caracter...
    pwm(200)
end if ;configura o PWM
;fim do se

if rxdata=="e" then ;se recebeu caracter...
    pwm(250)
end if ;configura o PWM
;fim do se

goto teste ;salta para teste

```

---

## 8. Controle de Display

```

;este exemplo irá apresentar o controle
;de um display lcd de 16 colunas por 2 linhas

dirpin=%00000000 ;inicializa os ios do microcontrolador
inic display ;inicializa o display
display(1,1,EasySTEP o microc.)
;mostra mensagem na primeira linha do
;display
display(2,1,do FUTURO!!!)
;mostra mensagem na segunda linha do
;display

```

---

## 9. Desenvolvendo um contador

```

;neste exemplo será apresentado os passos
;para se desenvolver um contador com o display
;e uma entrada de io

dirpin=%00000001 ;configura a direção do pino
inic display ;inicializa o display
display(1,1,***Contador***) ;mostra uma mensagem no display

loop:

if iol=1 then ;se a entrada estiver pressionada...
    inc(w1) ;incrementa a variável
    display(2,5,w1) ;mostra mensagem no display
    waitiol(0) ;aguarda o botão ficar solto

```

```
end if  
  
goto loop ;volta para loop
```

---

## 10. Testando o Pulsout

*;este exemplo irá demonstrar a utilização do comando  
;pulsout. A idéia é oscilar 10 vezes em um intervalo de  
;tempo de 255 ms uma carga conectada na saída do IO7*

*pulsout(255, 10)*