

# Prototipagem e Montagem de Placa de Circuito Impresso

Aula 01 - Introdução à prototipagem e montagem de placa de circuito impresso

# Apresentação

Nesta aula, conheceremos as etapas do fluxo de desenvolvimento de um projeto para construir uma placa de circuito impresso. Aprenderemos a programar o microcontrolador que controlará o circuito da placa que desenvolveremos nesta disciplina.

Conheceremos também o ambiente MPLAB, responsável pela gravação do código na nossa placa, e o compilador CCS, que utilizaremos para programar e compilar em C o algoritmo que descreverá o que fará o microcontrolador da nossa placa.

## Objetivos

Ao final desta aula, você será capaz de:

- Compreender o fluxo de desenvolvimento de projeto de uma placa de circuito impresso e saber identificar suas etapas.
- Utilizar o compilador CCS para analisar e compilar o código do microcontrolador da placa de circuito impresso que desenvolveremos nesta disciplina.
- Utilizar o ambiente de desenvolvimento MPLAB para gravar o código na placa.
- Entender para que serve uma placa de circuito impresso e as suas vantagens.

# Prototipagem e montagem de placa de circuito impresso

Na disciplina de “Circuitos Eletrônicos”, você estudou o comportamento e o funcionamento de circuitos compostos por resistores, diodos, transistores e amplificadores. Nesta disciplina, estudaremos como fazer um protótipo e montar uma placa de circuito impresso para projetos que utilizam esses dispositivos eletrônicos, e até mesmo outros tipos de dispositivos, como microcontroladores, chips com funcionalidades específicas (ASICs).

Uma placa de circuito impresso é uma placa onde “imprimimos” um circuito eletrônico. Essa “impressão” consiste em traçar as trilhas que representam os fios do circuito que interligam todos os seus componentes como, por exemplo, resistores, capacitores, processador, etc. Esse processo traz a vantagem de poder tornar comercial determinado projeto, permitindo que um mesmo circuito eletrônico seja replicado tantas vezes quantas forem necessárias. Outras vantagens que são facilmente observáveis é obter o mesmo circuito de forma mais organizada, e realizar uma otimização no projeto, de forma que o circuito ocupe a menor área possível. De tal maneira, que aprenderemos, nesta disciplina, a projetar e montar essas placas de circuito impresso.

## Atividade 01

1. Pesquise na internet sobre placas de circuito impresso e dê exemplos conhecidos dessas placas que temos no nosso dia a dia.

## Circuito a ser desenvolvido

Nesta disciplina, desenvolveremos um circuito impresso que controlará um **semáforo de dois tempos**.

## Semáforo de dois tempos

Um semáforo de dois tempos consiste em dois semáforos interligados, de forma que, quando um está **verde**, o outro está **vermelho**. E vice-versa.

**Figura 01** - Semáforo de dois tempos



## Etapas no desenvolvimento do circuito

No desenvolvimento de um circuito impresso cujo projeto se faz necessário a utilização de um microcontrolador, o fluxo que costumamos seguir é o seguinte:

1. programação do microcontrolador;
2. projeto da placa de circuito impresso;
3. construção da placa.

Portanto, para casos de circuitos sem microcontrolador, não se faz necessária a primeira etapa, a programação do microcontrolador.

A sequência de passos e respectivos sub-passos que seguiremos no projeto de desenvolvimento do circuito controlador do semáforo de dois tempos será:

## I. Programação do microcontrolador:

1. Programação em C da lógica de funcionamento do 'semáforo de dois tempos', utilizando o compilador **CCS C**.
2. Gravar código (formato hex) no microcontrolador, utilizando o **MPLAB** e gravador da placa de desenvolvimento.

**Observação:** Você verá no decorrer da aula como realizar a gravação do código (formato 'hex') no microcontrolador utilizando o MPLAB antes de utilizar o compilador CCS C, isso porque o nosso código em C já está pronto, mas a ordem, na prática, é programar em C a lógica desejada, compilar (com o CCS C) e verificar se não houve erro. Caso exista o erro, corrigir, se não, simular para verificar se a lógica programada condiz com o esperado, em caso afirmativo utilizar o código em 'hex' gerado pelo compilador CCS C para gravar no microcontrolador utilizando o MPLAB. Portanto, em futuros projetos a ordem para programação do microcontrolador é a mesma descrita aqui: CCS C -> MPLAB.

## II. Projeto da placa de circuito impresso:

1. Desenvolvimento do protótipo em software.
2. Simulação do protótipo em software.
3. Exportação do protótipo para o processo de fabricação.

## III. Construção da placa:

1. Impressão da placa.
2. Soldagem dos componentes na placa.

Agora, veremos como fazer a primeira etapa de **programação do microcontrolador**. Antes, vamos aprender a usar o compilador CCS e o ambiente MPLAB. As demais etapas serão feitas nas demais aulas da disciplina.

# O ambiente de desenvolvimento MPLAB

O MPLAB é um software com um ambiente de desenvolvimento integrado (IDE, do inglês Integrated Development Environment), fornecido pela empresa Microchip Technology. O MPLAB permite escrever, compilar, simular, depurar o código em tempo real e também gravar o código em microcontroladores PIC da Microchip.

Além das explicações dadas na tabela anterior, temos mais três observações, as quais se encontram listadas a seguir.

## Para saber mais

Se você quiser saber maiores detalhes sobre o MPLAB, poderá verificar a opção **Help->About** no menu da tela do MPLAB, ou ainda, acessar o link: [<http://pt.wikipedia.org/wiki/MPLab>](http://pt.wikipedia.org/wiki/MPLab).

PIC é a abreviatura usada para *Peripheral Interface Controller* (controlador de interfaces periféricas ou, simplesmente, controlador de periféricos) — linha de microcontroladores fabricada exclusivamente pela Microchip Technology.

Você verá mais detalhes sobre esses microcontroladores na disciplina de “Projetos de Sistemas Microcontrolados”. Lá, você estudará um pouco sobre as características gerais dos microcontroladores da família 16F, bem como sua arquitetura interna. Também aprenderá sobre suas organizações da memória de programa e da memória de dados, sobre os registros de funções especiais e sobre as principais Interfaces periféricas presentes em sua organização, com o intuito de facilitar a programação e o projeto de sistemas digitais com o uso desses dispositivos.

## Saiba mais

Se você quiser saber maiores detalhes sobre o microcontrolador PIC que usaremos na nossa placa, é só acessar o link: [<http://pt.wikipedia.org/wiki/Pic16f628>](http://pt.wikipedia.org/wiki/Pic16f628).

Nesta aula, o MPLAB será utilizado apenas para importar o arquivo “.hex” gerado no compilador CCS, o **PIC 16F628A**, e para colocá-lo no nosso microcontrolador.

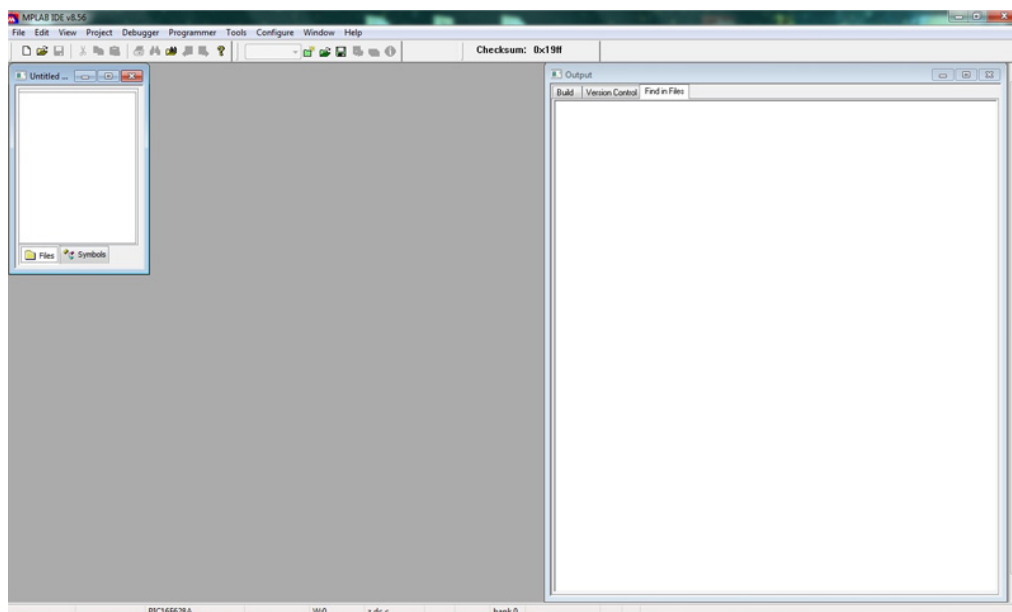
Algumas configurações que precisaremos fazer são:

- escolher o microcontrolador;
- importar o arquivo “.hex”;
- escolher o gravador e definir suas configurações.

## Usando o MPLAB

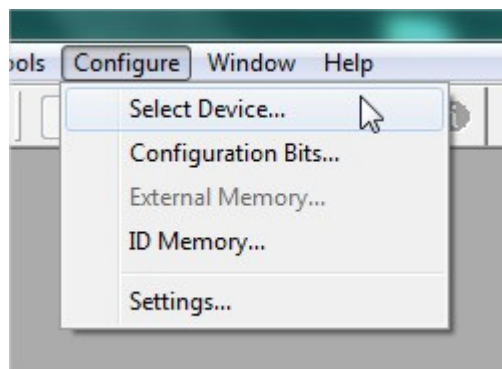
Ao iniciar o MPLAB, você terá uma tela igual a da **Figura 2**.

**Figura 02** - Tela inicial do MPLAB



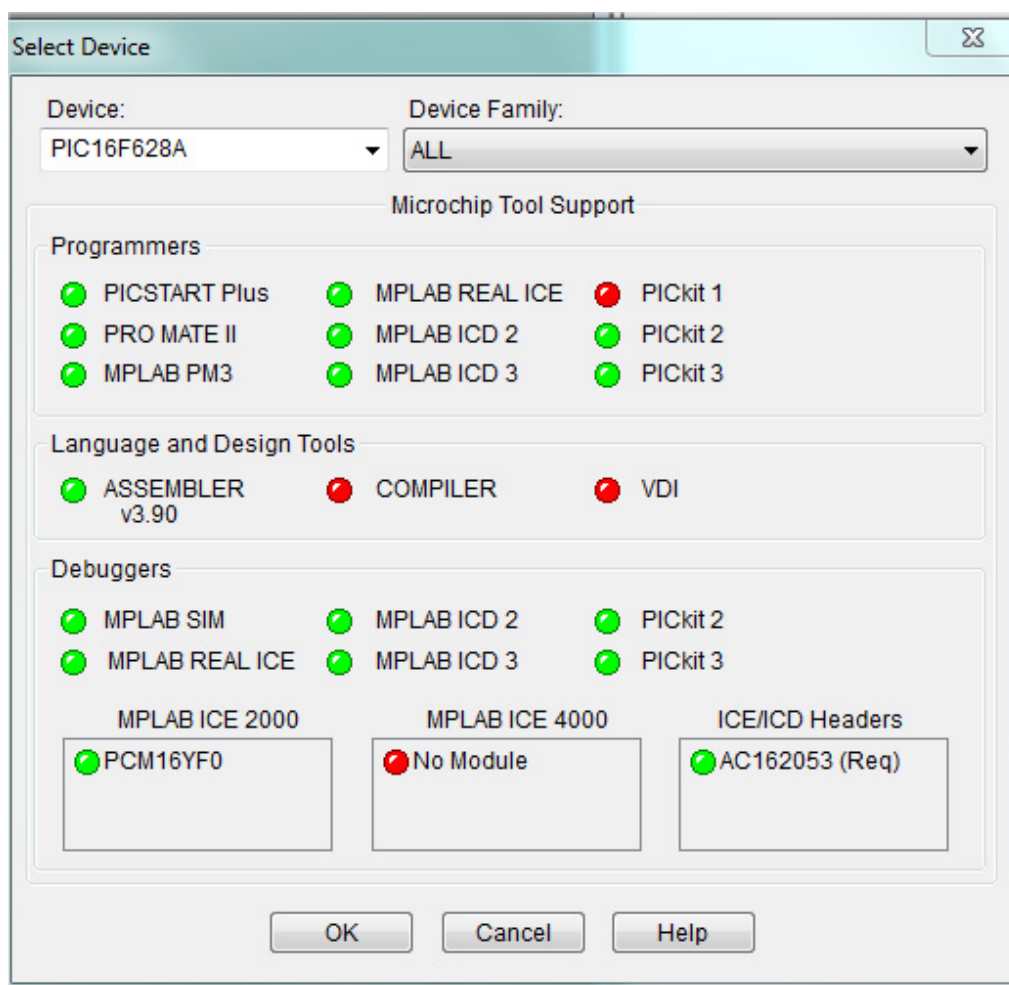
Inicialmente, vamos definir o microcontrolador do nosso circuito. Para isso, devemos selecionar no menu a opção: Configure → Select Device, como mostra a **Figura 3**.

**Figura 03** - Opção de seleção de dispositivo no MPLAB



Na tela de seleção do dispositivo (**Figura 4**), escolheremos o PIC 16F628A e clicamos no botão "OK".

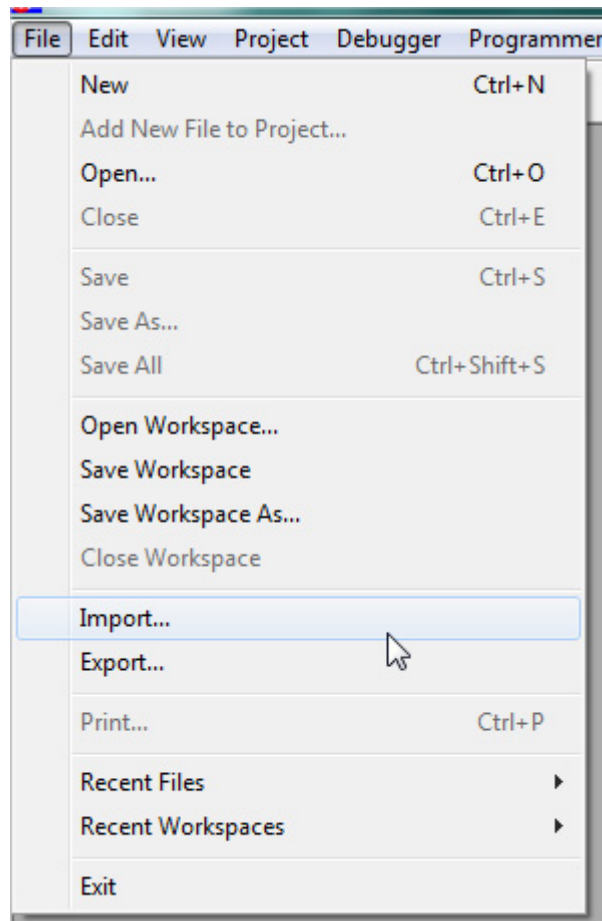
**Figura 04** - Seleção de dispositivo no MPLAB



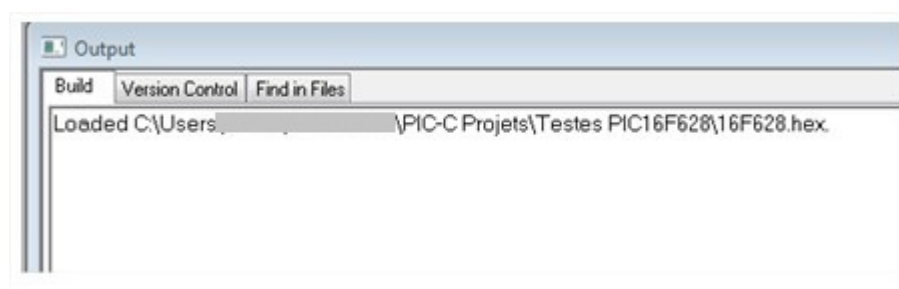


Agora que selecionamos o microcontrolador, o próximo passo é importar o arquivo “.hex”. Fazemos isso selecionando no menu a opção File → Import, como visto na **Figura 5**, e depois escolhemos nosso arquivo “.hex”. Podemos verificar que ele foi carregado na tela de “Output” (**Figura 6**).

**Figura 05** - Opção de importação de arquivo “.hex” no MPLAB

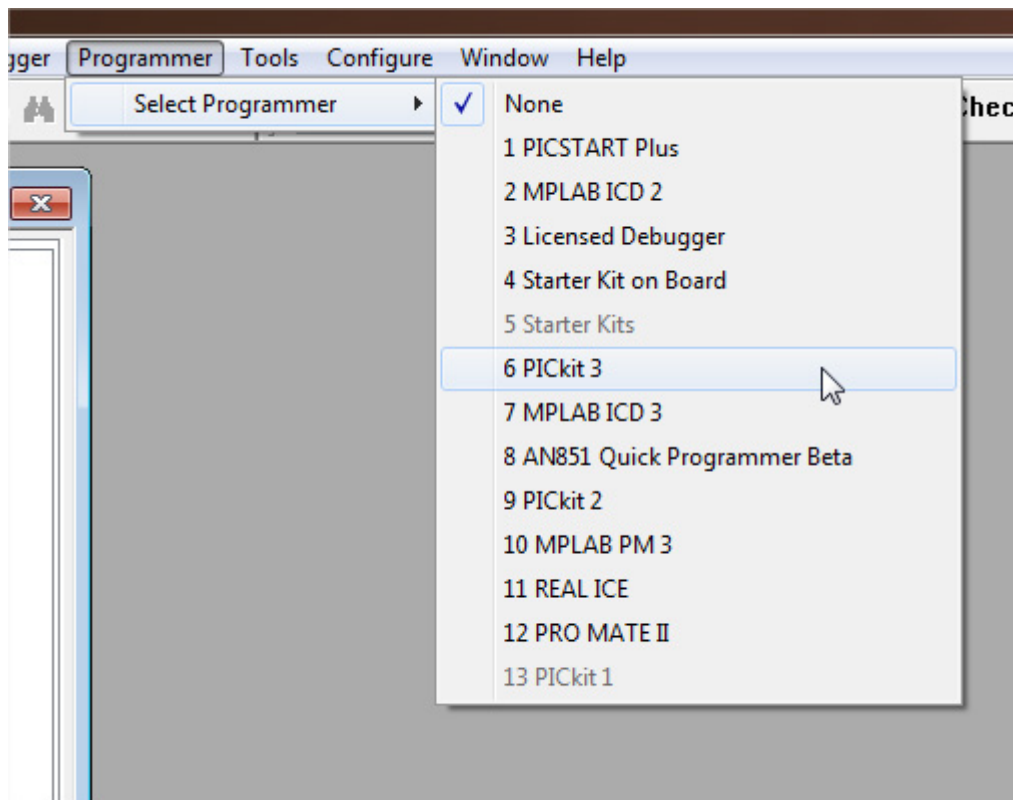


**Figura 06** - Tela de saída (output) do MPLAB



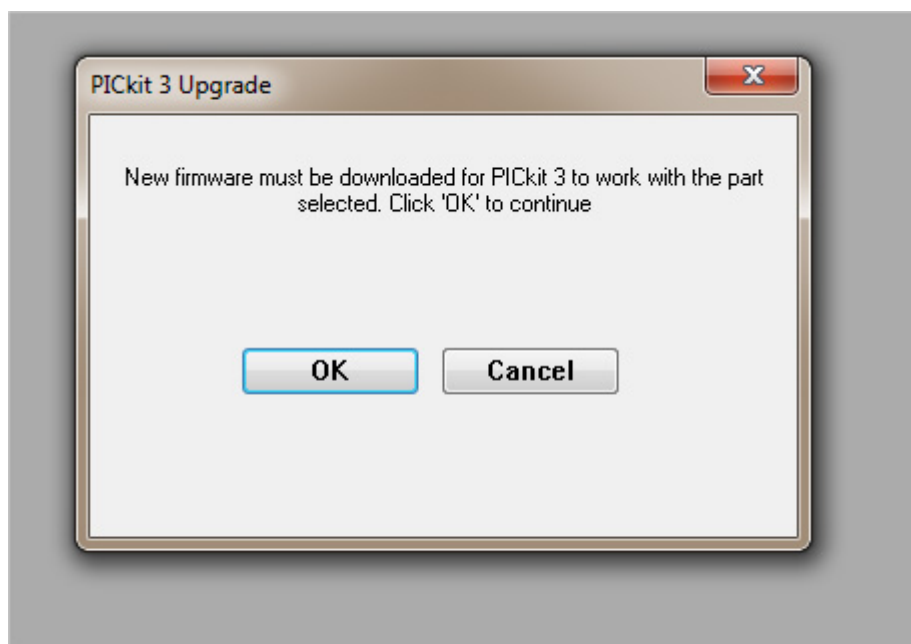
O próximo passo agora é definir o gravador. Usaremos o **PICKit3**. Para selecioná-lo, basta ir no menu, na opção Programmer → MPLAB PICKit3, como mostra a **Figura 7**.

**Figura 07** - Selecionando o gravador o PICkit3 no MPLAB



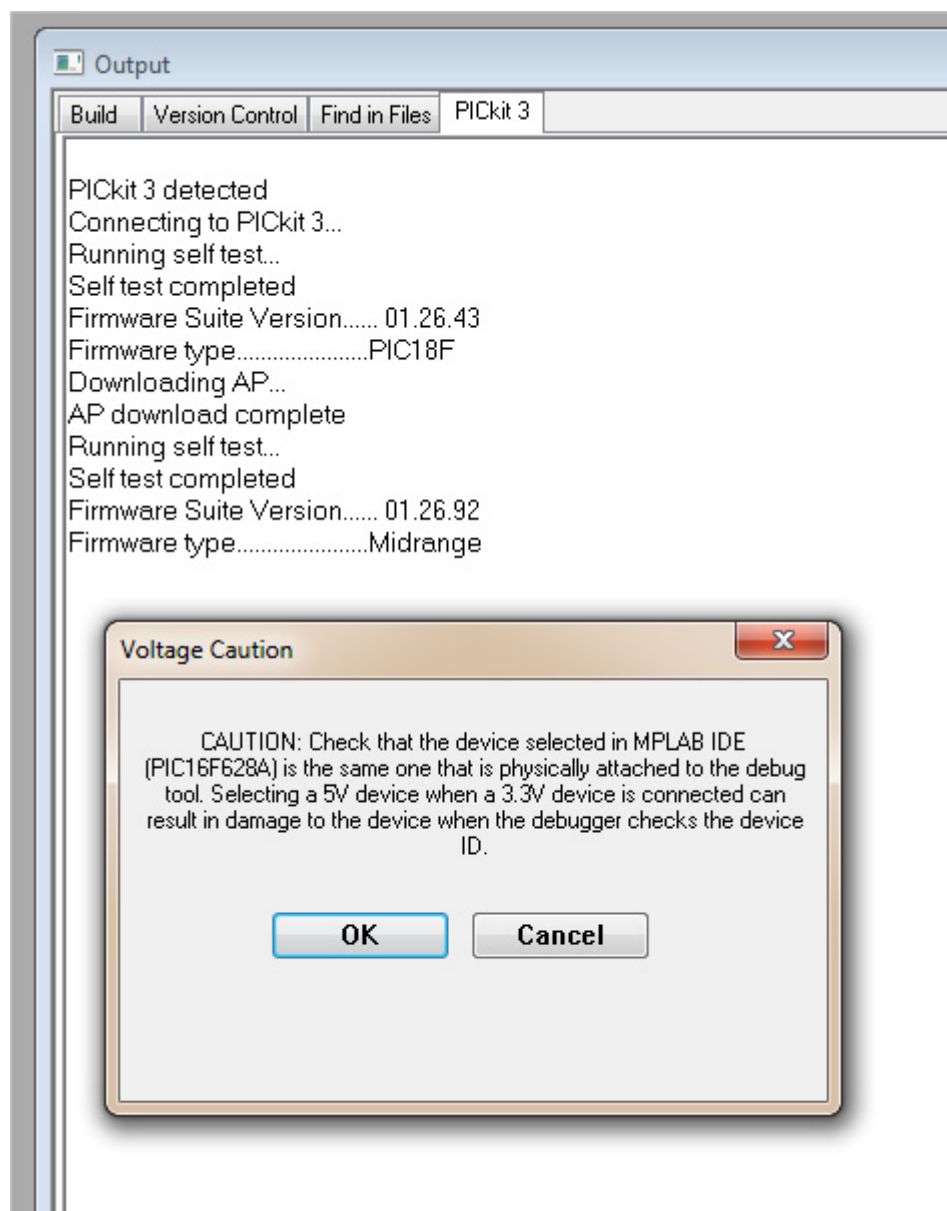
Quando você tiver selecionado o PICkit3, uma mensagem como a mostrada na **Figura 8** pode aparecer. Ela informa que precisamos atualizar o firmware do gravador para poder usá-lo com esse tipo de microcontrolador. Nesse caso basta clicar no botão “OK” e esperar.

**Figura 08** - Mensagem de solicitação de atualização de firmware



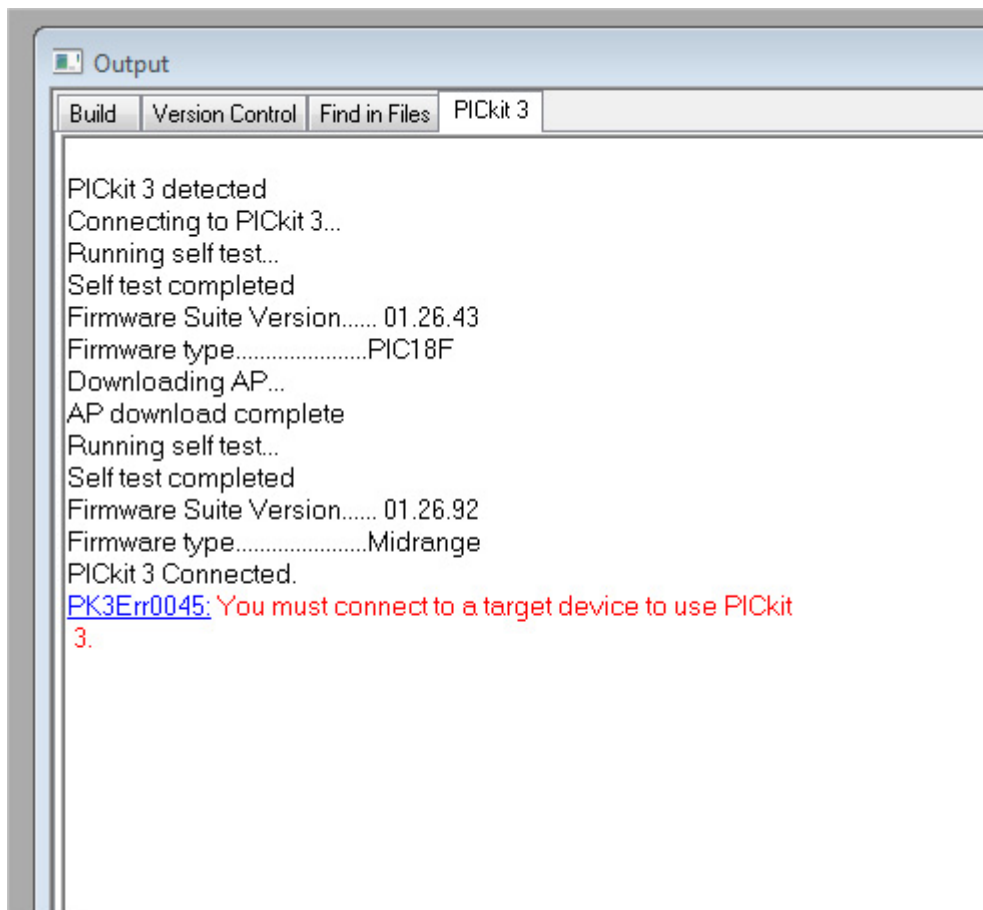
Outras mensagens podem aparecer, como a mostrada na **Figura 9**. Essa mensagem, por exemplo, pede para tomarmos cuidado com as tensões que serão usadas para a placa.

**Figura 09** - Aviso sobre as tensões que serão usadas na placa.



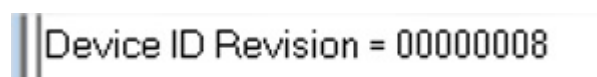
Ao término de toda a configuração do gravador, a tela da **Figura 10** será mostrada, informando que não encontrou nenhum alvo (microcontrolador), mas está tudo pronto para a conexão.

**Figura 10** - Tela de confirmação do término de configuração do gravador.



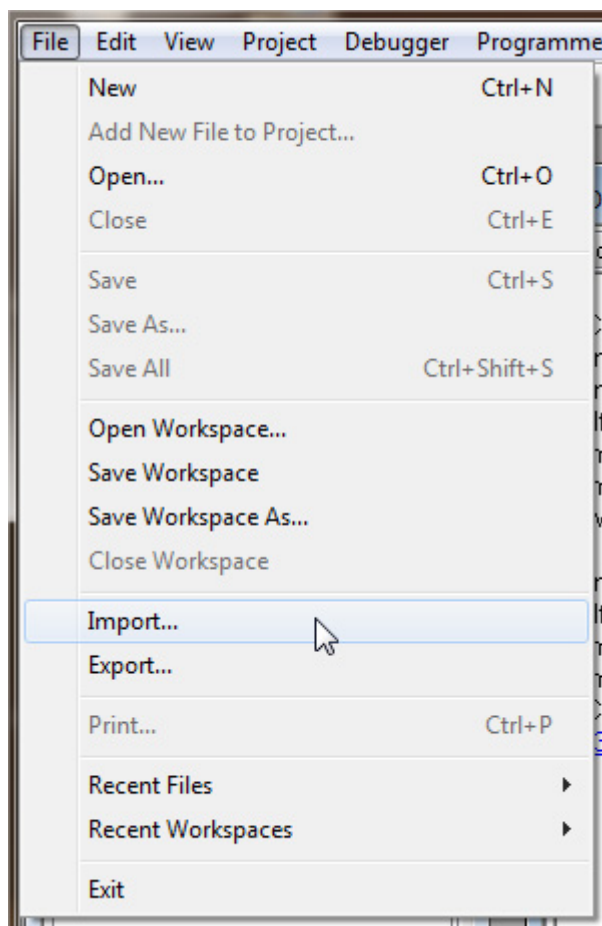
Quando a placa estiver com energia, automaticamente o gravador irá reconhecer o microcontrolador. E aí, uma imagem, como a mostrada na **Figura 11**, deverá aparecer.

**Figura 11** - Reconhecimento do dispositivo



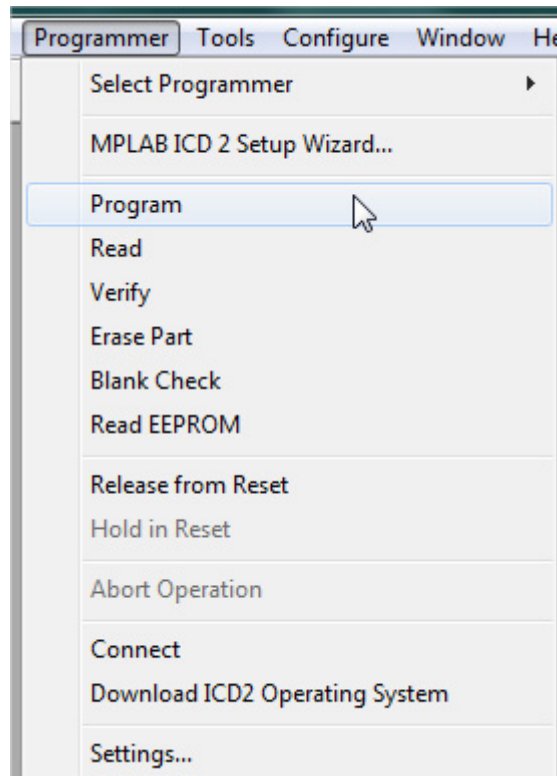
Escolhido o dispositivo, chegou a hora de gravar o nosso arquivo “.hex”, que será importado para a placa. Para isso, precisamos selecionar o arquivo que será importado, através da opção File → Import (**Figura 12**).

**Figura 12** - Selecionando o arquivo “.hex”



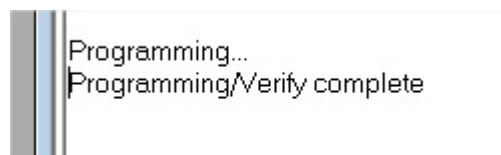
Selecionado o arquivo “.hex”, o próximo passo é programar o microcontrolador. Fazemos isso selecionando a opção do menu: **Programmer** → **Program** (Figura 13).

**Figura 13** - Programando o microcontrolador



Ao executarmos esse passo, poderemos ver na tela de “Output” (**Figura 14**) o resultado da gravação.

**Figura 14** - Resultado da gravação



É importante lembrar que o MPLAB IDE serve também para criar códigos e depurá-los, mas esse não será o objetivo dessa aula. Você pode estar se perguntando: E o arquivo “.hex”? De onde ele veio?

Esse é o nosso próximo passo no curso, vamos criar um código em C usando a lógica do semáforo e, ao compilá-lo, iremos perceber que serão gerados alguns arquivos de saída do nosso código, dentre eles teremos o “.hex”.

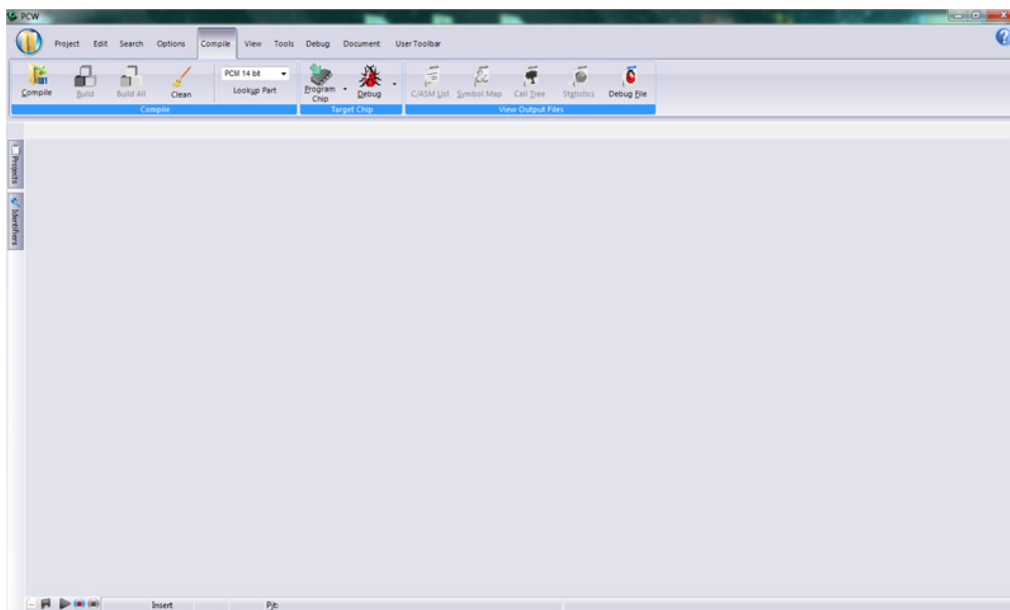
Para criar esse nosso código em C, vamos falar um pouco sobre CCS.


# O compilador CCS

O compilador CCS é um ambiente de desenvolvimento para microcontroladores PIC que possui suporte para várias famílias de PIC, permitindo que o usuário escreva o programa em linguagem C.

A **Figura 15** mostra a tela inicial do ambiente de compilação CCS.

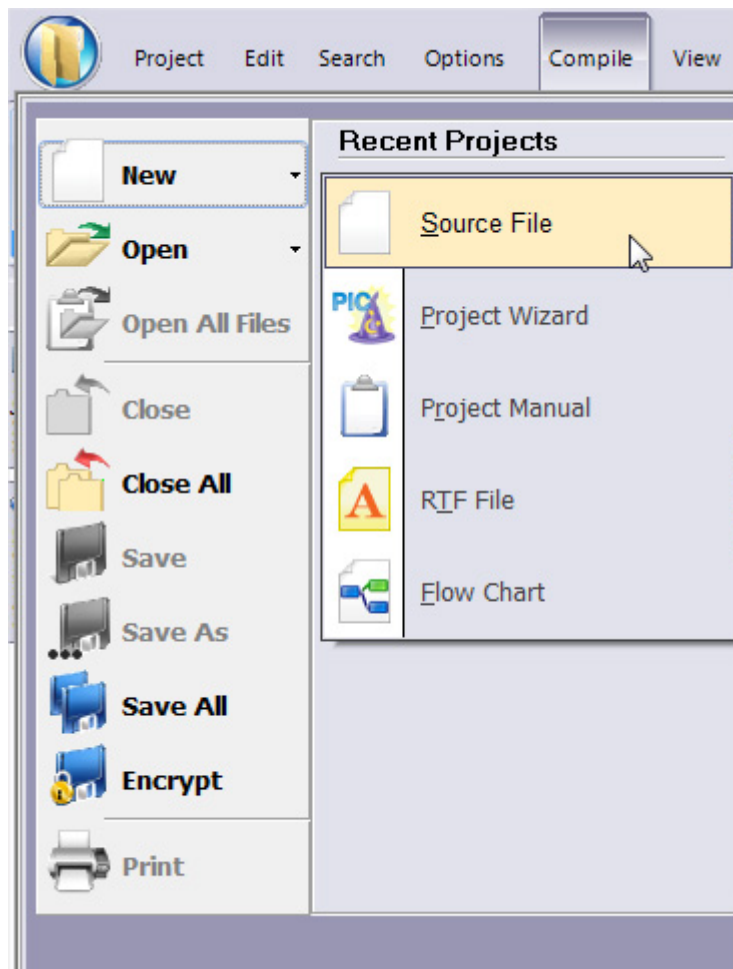
**Figura 15** - Tela inicial do CCS



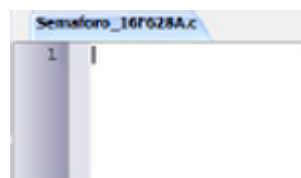
Inicialmente, vamos clicar no ícone  e depois na opção New → Source File, como mostra a **Figura 16**. Daí, escolhemos a pasta onde vamos salvar e criamos o nosso arquivo “Semaforo\_16F628A.c” (**Figura 17**).

O próximo passo é escrever o código responsável por executar a lógica desejada na linguagem C. No caso, para o projeto do semáforo, vamos adiantar o processo e utilizar o código que está descrito na **Listagem 01**, conforme mostrado na **Figura 18**.

**Figura 16** - Criação de um novo arquivo no CCS



**Figura 17** - Criação do arquivo "Semaforo\_16F628A.c"





**Figura 18** - Janela do CCS com o código C do arquivo

```

Semaforo_16f628a.c
1  #include <16f628a.h>           // Inclui o cabeçalho específico do pic a ser utilizado
2
3  #FUSES NOWDT                  //No Watch Dog Timer
4  #FUSES HS                     //High speed Osc (> 4mhz)
5  #FUSES NOPUT                  //No Power Up Timer
6  #FUSES NOPROTECT              //Code not protected from reading
7  #FUSES BROWNOUT               //Reset when brownout detected
8  #FUSES MCLR                   //Master Clear pin enabled
9  #FUSES NOCPD                  //No EE protection
10
11
12  #use delay(clock=20M)         // Seta o clock interno para 20Mhz
13
14  #define L_VERMELHO1 PIN_B4
15  #define L_VERMELHO2 PIN_B2
16
17  #define L_VERDE1 PIN_B1
18  #define L_VERDE2 PIN_B5
19
20  #define L_AMARELO1 PIN_B3
21  #define L_AMARELO2 PIN_B0
22
23
24  void main(void) // função principal
25  {
26      do {
27          /*
28              S1 S2
29              O X
30              O O
31              X O
32          */
33          output_high(L_VERMELHO2);
34          output_high(L_VERDE1);
35          output_low(L_AMARELO2);
36          output_low(L_VERMELHO1);
37          delay_ms(3000);
38          /*
39              S1 S2

```

```

1  #include <16f628a.h>
2  // Inclui cabeçalho específico do
3  PIC a ser utilizado
4
5  #FUSES NOWDT //No Watch Dog Timer
6  #FUSES HS //High speed Osc (> 4mhz)
7  #FUSES NOPUT //No Power Up Timer
8  #FUSES NOPROTECT //Code not protected from reading
9  #FUSES BROWNOUT //Reset when brownout detected
10 #FUSES MCLR //Master Clear pin enabled
11 #FUSES NOCPD //No EE protection
12
13 #use delay(clock=20M) // Seta o clock interno para 20Mhz
14
15 #define L_VERMELHO1 PIN_B4
16 #define L_VERMELHO2 PIN_B2
17
18 #define L_VERDE1 PIN_B1
19 #define L_VERDE2 PIN_B5
20
21 #define L_AMARELO1 PIN_B3
22 #define L_AMARELO2 PIN_B0

```

```

1 void main(void) // função principal
2 {
3     do {
4         /*
5          S1 S2
6          O X
7          O O
8          X O
9          */
10        output_high(L_VERMELHO2);
11        output_high(L_VERDE1);
12        output_low(L_AMARELO2);
13        output_low(L_VERMELHO1);
14        delay_ms(3000);
15        /*
16        S1 S2
17        O X
18        X O
19        O O
20        */
21        output_low(L_VERDE1);
22        output_high(L_AMARELO1);
23        delay_ms(1000);
24
25        output_low(L_VERMELHO2);
26        output_high(L_VERDE2);
27        output_low(L_AMARELO1);
28        output_high(L_VERMELHO1);
29        delay_ms(3000);
30        /*
31        S1 S2
32        X O
33        O X
34        O O
35        */
36        output_low(L_VERDE2);
37        output_high(L_AMARELO2);
38        delay_ms(1000);
39    } while (TRUE); // mantém o laço de
40        repetição rodando em loop infinito
41 }


```

**Listagem 01** – Código C do nosso semáforo (Semaforo\_16F628A.c)

Na disciplina “Projetos de Sistemas Microcontrolados”, você verá, de forma mais detalhada, o que representa cada parte do código da **Figura 19**: definição do PIC a ser utilizado; comportamento dos fuses para o nosso projeto; do cristal; das portas; dos comentários e das funções principais do nosso semáforo.

Nesse código, você pode verificar que as portas de saída estão relacionadas a flags, como por exemplo: na porta B, o pino 1 está relacionado à flag L\_VERDE1, informando que é o Led verde do semáforo 1.

Após criarmos todo o nosso código (**Listagem 01**), clicamos na opção Compile ->

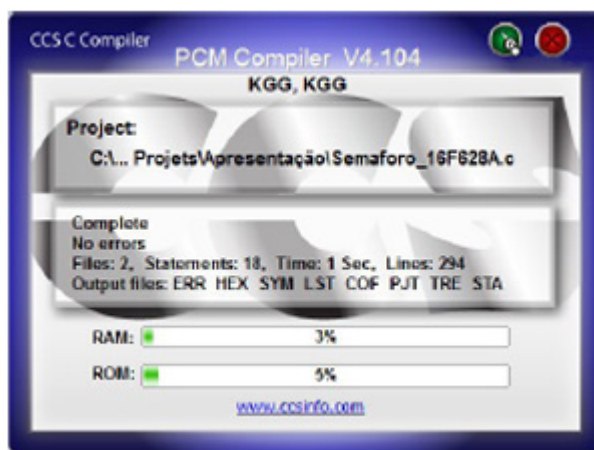
Compile  para que nosso código seja compilado.

Lembra dos arquivos de saída? Um deles é o “.hex” de que falamos tanto! Pronto, é justamente nesse ponto que ele será criado.

Caso seu código não tenha nenhum erro, uma tela, como a da Figura 19, será mostrada. É nessa tela que podemos ver os arquivos que foram gerados: “.ERR”, “.HEX”, “.SYM”, dentre outros. Esses arquivos foram gerados dentro da mesma pasta em que você salvou o código em C e apresentam, por padrão, o mesmo nome em que foi salvo o código em C, diferenciados apenas, pelas extensão (".ERR", ".HEX", ".SYM", etc).

Esse arquivo “.hex” é o que utilizamos para gravar o nosso microcontrolador, os outros arquivos tem funções específicas, como debugger, que não serão abordadas aqui.

**Figura 19** - Tela de resultado da compilação no CCS



Por fim, na barra de “Output” (**Figura 20**), podemos ver também um relatório da nossa compilação, já que a outra tela (**Figura 19**) desaparece rapidamente.

**Figura 20** - Relatório de resultado da compilação no CCS

Output
Memory usage: ROM=5%    RAM=3% - 3% 0 Errors, 0 Warnings.

## Leitura Complementar

Se quiser saber mais sobre os microcontroladores PIC, você pode procurar esses livros:

PEREIRA, Fábio. **Microcontroladores PIC**: técnicas avançadas. São Paulo: Editora Érica, 2002.

SOUZA, David José de. **Desbravando o PIC**. São Paulo: Editora Érica, 2000.

E se quiser saber mais sobre o MPLAB, consulte o seguinte link:

WIKIPÉDIA. **MPLAB**. Disponível em: <<http://pt.wikipedia.org/wiki/MPLab>>. Acesso em: 11 jul. 2012.

## Resumo

Nesta aula, vimos como programar o microcontrolador do nosso circuito usando as ferramentas MPLAB e CCS. Usamos o MPLAB para importar o arquivo “.hex” e programar nossa placa. Além disso, usamos o CCS para criar nosso arquivo em C.

## Autoavaliação

1. Quais são as etapas que devemos seguir no fluxo de projeto e desenvolvimento de uma placa de circuito impresso?
2. Que ferramenta utilizamos para gravar o código na nossa placa?
3. E qual é a ferramenta usada para programar microcontroladores PIC?

## Referências

MICROCHIP. **Microchip Technology Inc.**

Disponível em: <<http://www.microchip.com//pagehandler/en-us/family/mplabx>>.

Acesso em: 11 jul. 2012.

WIKIPÉDIA. MPLAB. Disponível em: <<http://pt.wikipedia.org/wiki/MPLab>>. Acesso em: 11 jul. 2012a.

\_\_\_\_\_. **PIC16F628**. Disponível em: <<http://pt.wikipedia.org/wiki/Pic16f628>>. Acesso em: 11 jul. 2012b.