Prototipagem e Montagem de Placa de Circuito Impresso

Aula 04 - Roteiro Prático nº 01 - Uso do MPLAB e do CCS

Apresentação

Nesta aula, que será nossa primeira aula prática, nós veremos como gravar a lógica a ser executada no microcontrolador por meio do programa MPLAB. Veremos ainda, que anteriormente ao passo de gravação, essa lógica deve ser programada na linguagem C. Para a nossa prática, foi escolhido o compilador CCS para compilar o código em questão e gerar o arquivo a ser gravado no microcontrolador.

Objetivos

Ao final desta aula, você será capaz de:

- Rever alguns dos conteúdos vistos anteriormente: fluxo de desenvolvimento e execução de algoritmos em microcontroladores PIC, utilizando os programas CCS S e MPLAB.
- Analisar e compilar o código do microcontrolador que desenvolveremos nesta disciplina.

O Circuito

O circuito impresso que desenvolveremos nesta disciplina fará o controle de um 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.

O fluxo de projeto que seguiremos para desenvolver o nosso circuito impresso é o seguinte:

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

Etapas no Desenvolvimento do Circuito

A sequência que seguiremos no projeto e desenvolvimento do circuito que controlará o semáforo de dois tempos será:

I. **Programação do microcontrolador**

a. **Programação em C da lógica de funcionamento do 'semáforo de dois tempos', utilizando o compilador CCS C.**

b. **Gravar código (formato hex) no microcontrolador, utilizando o MPLAB e gravador da placa de desenvolvimento.**

Atenção!

Você verá primeiro no decorrer da aula como fazer 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), para então, verificar se não houve erros, caso exista, corrigir, senão, simular para verificar se a lógica programada condiz com o esperado, e caso afirmativo, utilizar 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**

a. **o projeto da placa começa com o desenvolvimento do protótipo em software;**

b. **depois, fazemos a simulação desse protótipo em software;**

c. **a última parte dessa etapa é a exportação do protótipo para o processo de fabricação.**

III. **Fabricação da placa de circuito impresso**

1. **a fabricação da placa começa com o processo de construção da placa;**

2. **depois, fazemos o processo de solda da placa.**

Agora, nesta aula prática, faremos a 1ª etapa do nosso circuito impresso, a **programação do microcontrolador**.

O MPLAB

O **MPLAB** permite escrever, compilar, depurar o código em tempo real e também gravar o código no microcontrolador.

Nesta aula, o MPLAB será utilizado apenas para importar o arquivo ".hex" gerado no CCS e colocá-lo no nosso microcontrolador.

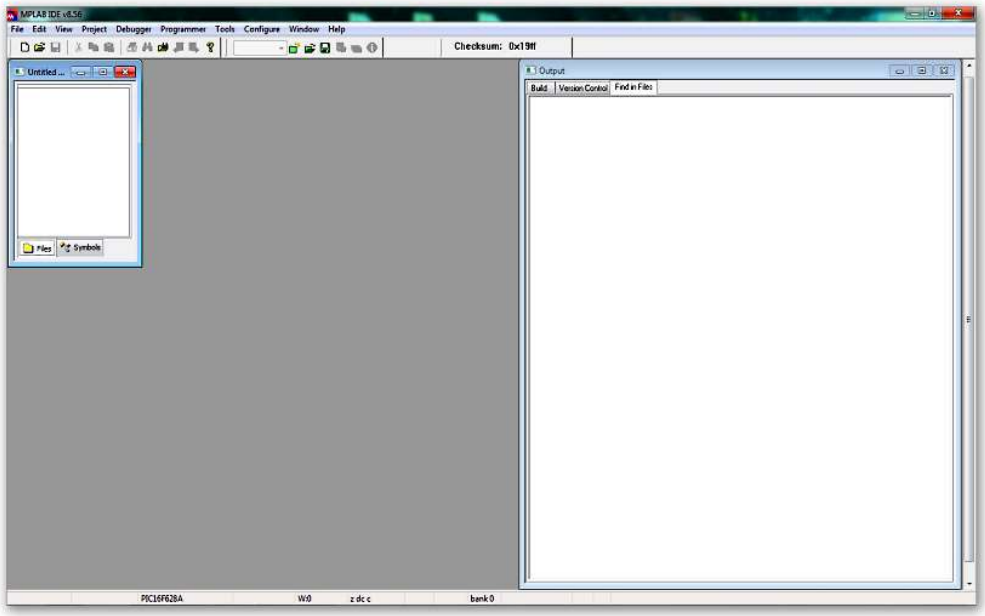
O microcontrolador que usaremos será o PIC 16F628A.

Algumas configurações devem ser feitas, são elas:

1. **escolher o microcontrolador;**
2. **importar o arquivo ".hex";**
3. **escolher o tipo de gravador;**
4. **definir as suas configurações.**

Na **Figura 1**, vemos a tela inicial do MPLAB.

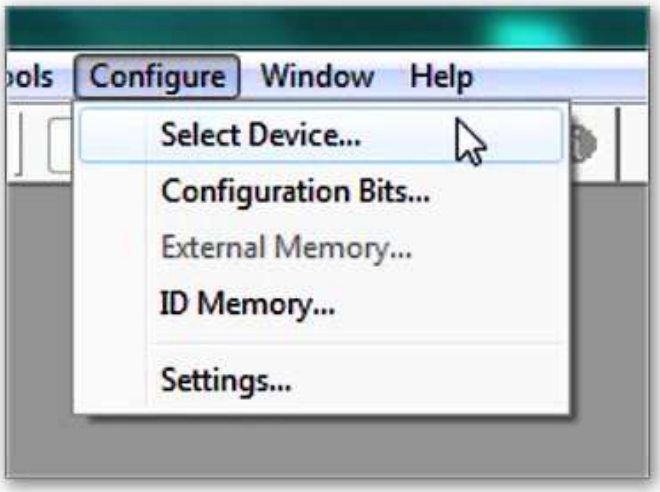
Figura 01 - MPLAB: Tela inicial



1. **Inicialmente, definiremos o nosso microcontrolador, fazendo os seguintes passos:**

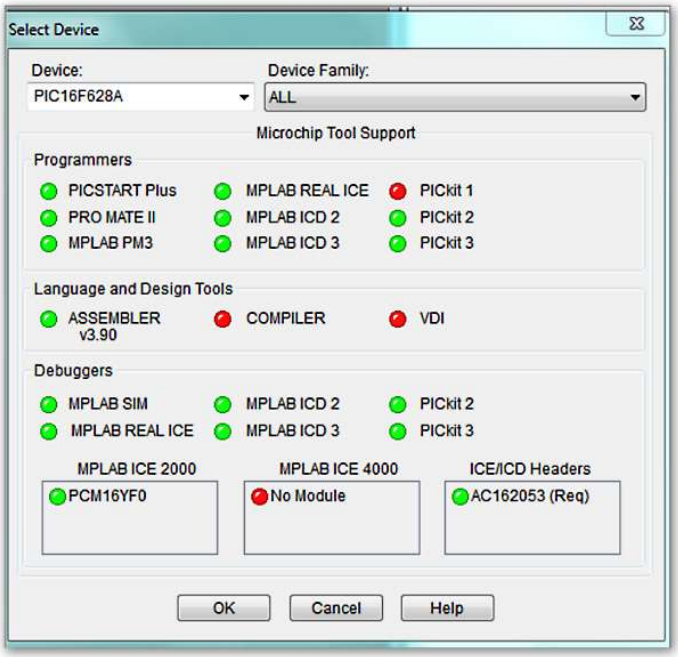
    - **Configure → Select Device... (Figura 2)**

Figura 02 - Seleção do dispositivo



- **Escolhemos o PIC 16F628A e clicamos em "OK". (Figura 3)**

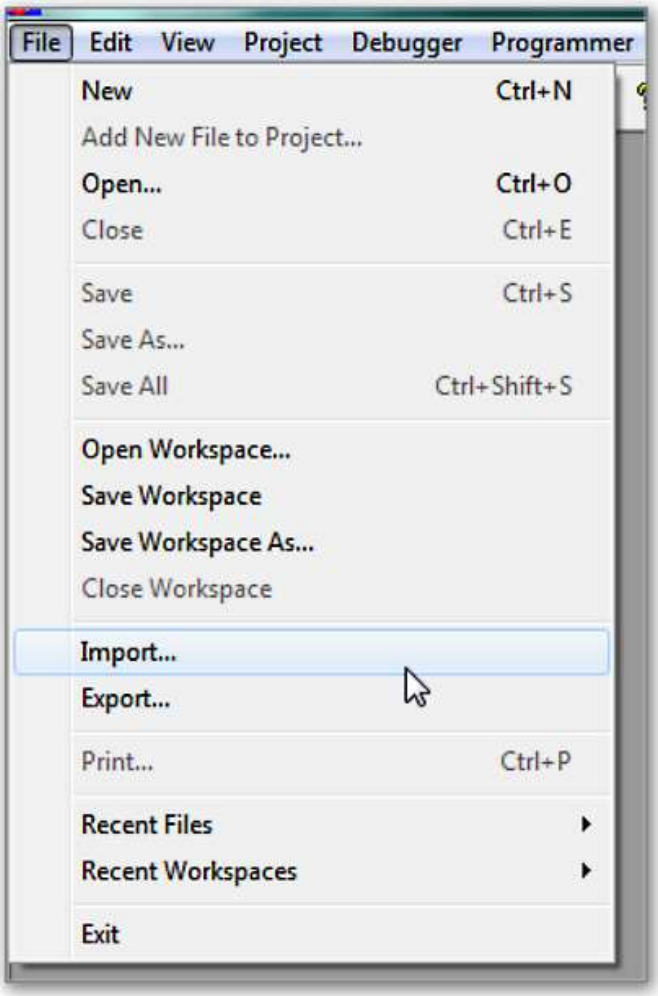
Figura 03 - Escolhendo o PIC 16F628A



2. **Agora, vamos importar o arquivo ".hex", executando os seguintes passos:**

- **Vamos em File → Import... (Figura 4).**

Figura 04 - Importando arquivo no MPLAB



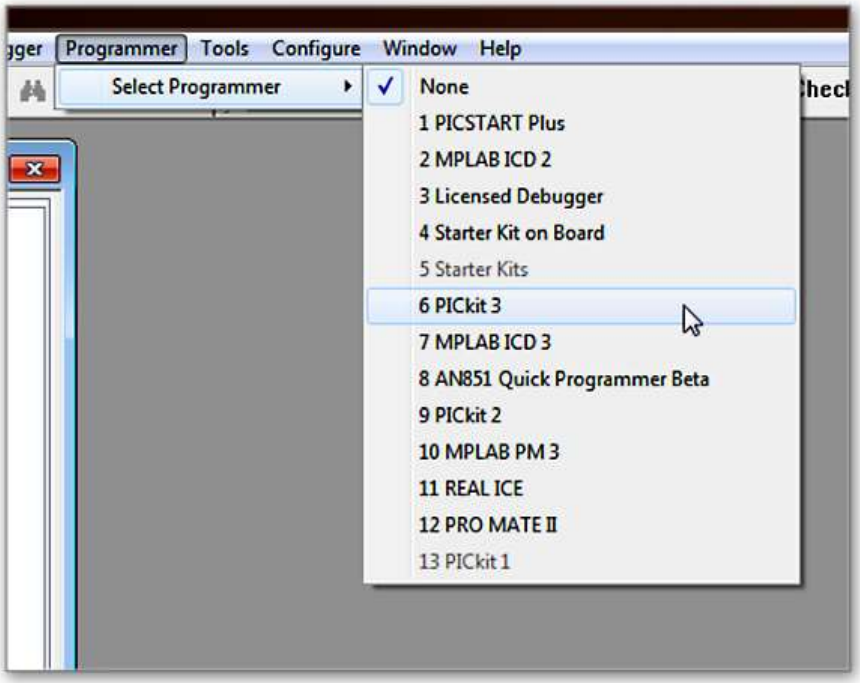
- **E escolhemos nosso arquivo ".hex".**
- **Podemos, então, verificar que ele foi carregado na tela de "Output" (Figura 5).**

Figura 05 - Tela de saída (*output*)

3. Vamos agora definir o tipo do gravador, que será o PICkit3.

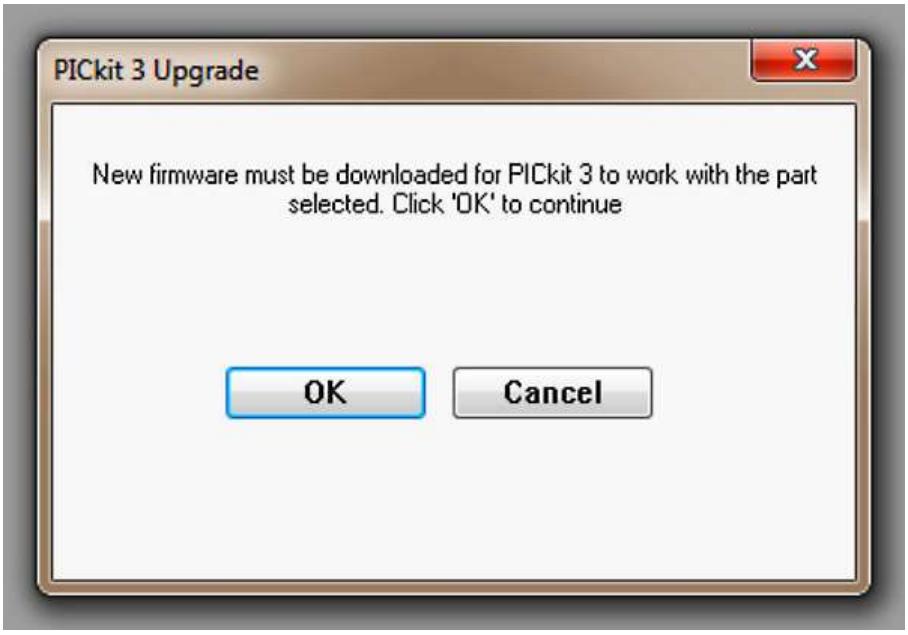
- **Selecione a opção de menu Programmer → Select Programmer → PICkit 3 (Figura 6).**

Figura 06 - Selecionando o gravador



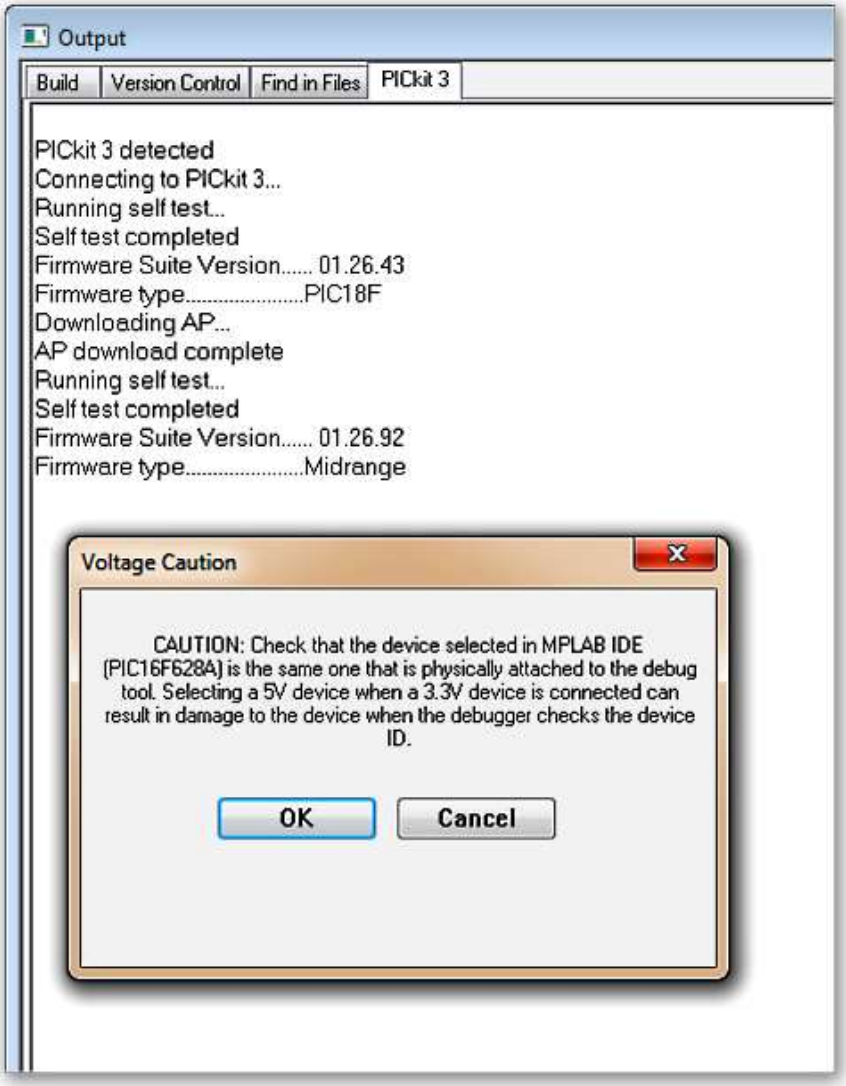
Uma mensagem, como a mostrada na Figura 7, pode aparecer. Ela está informando que precisamos atualizar o *firmware* do gravador para usá-lo com esse tipo de microcontrolador. Nesse caso, basta clicar em "OK" e esperar.

Figura 07 - Possível mensagem de solicitação de atualização de *firmware*



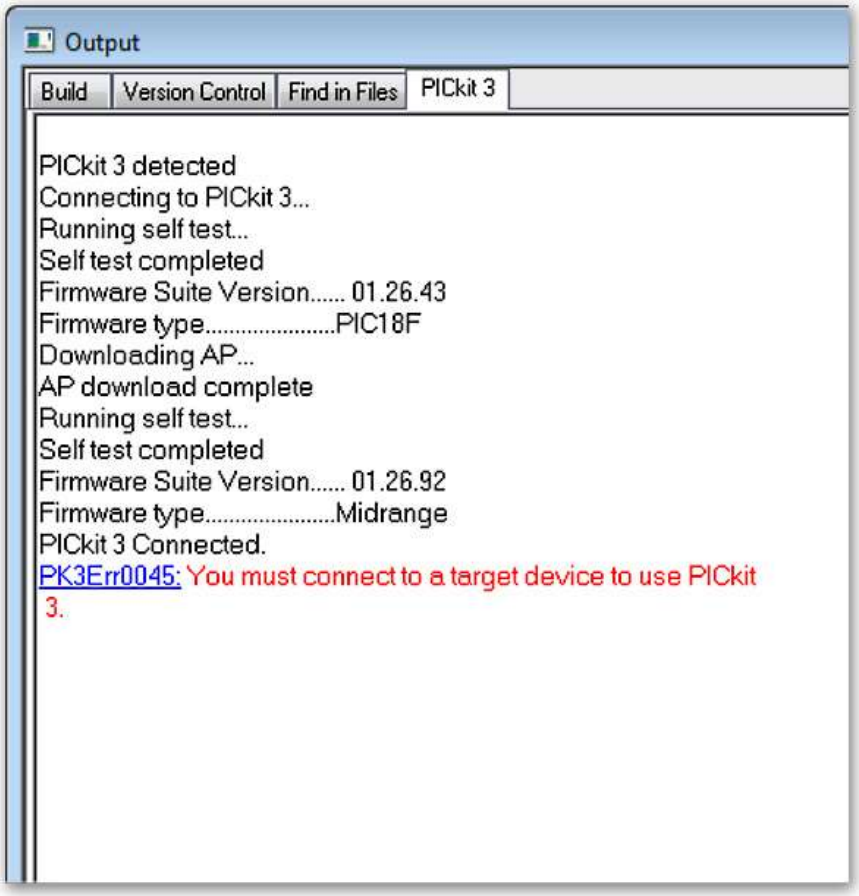
Outros avisos podem aparecer, como o mostrado na Figura 8. Esse aviso pede para tomarmos cuidado com as tensões que serão usadas para a placa, mas no nosso caso, não teremos problemas.

Figura 08 - Aviso sobre as tensões que serão usadas na placa



Ao término de toda a configuração do gravador, a tela da Figura 9 será mostrada.

Figura 09 - Tela de confirmação do término de configuração do gravador



A tela da Figura 9 está informando que não encontrou nenhum alvo (microcontrolador), mas está tudo pronto para a conexão.

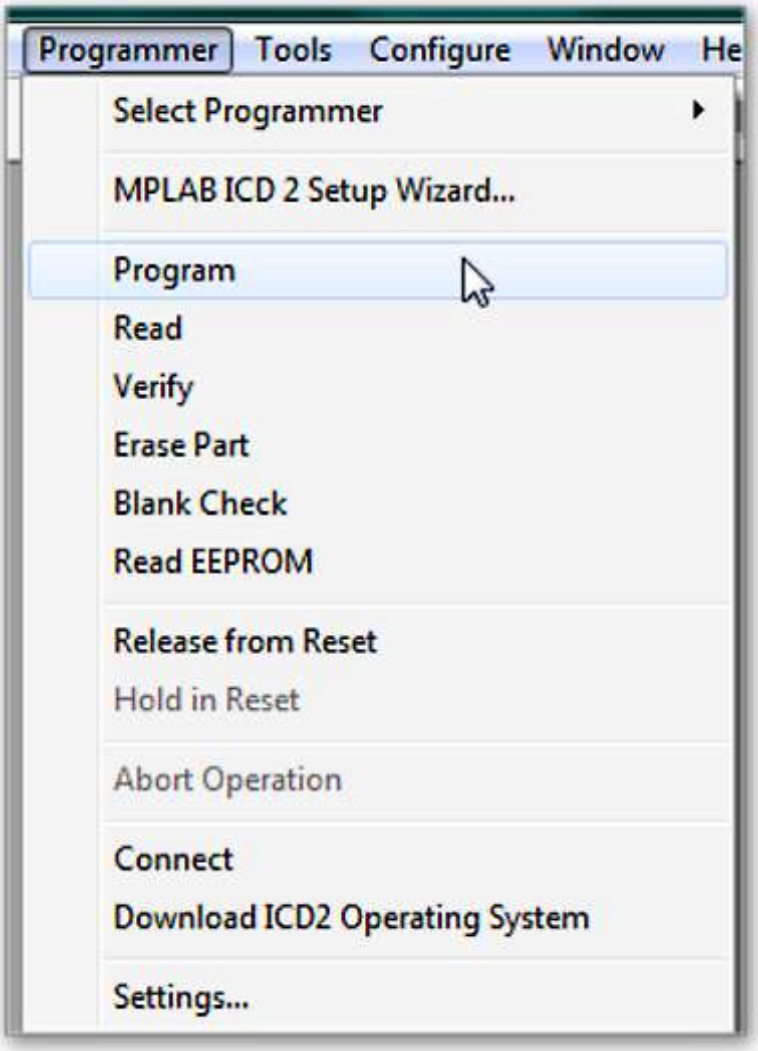
Quando a placa estiver com energia, automaticamente, o gravador irá reconhecer o microcontrolador e uma imagem, como a mostrada na Figura 10, deverá aparecer.

Figura 10 - Reconhecimento do dispositivo

Finalizamos a configuração de gravação do dispositivo.

4. **Para programar o microcontrolador, vamos selecionar a opção do menu: Programmer → Program, como mostrado na Figura 11.**

Figura 11 - Programando o microcontrolador



- **Na tela de Output (Figura 12), podemos ver o resultado da gravação.**

Figura 12 - Resultado da gravação

Você deve lembrar que o MPLAB IDE serve também para criar e depurar códigos, mas esse não será o objetivo dessa nossa aula prática.

Você pode estar se perguntando: E o arquivo ".hex"? De onde ele veio?

Esse é o nosso próximo passo, 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 relembrar um pouco do CCS.

O CCS

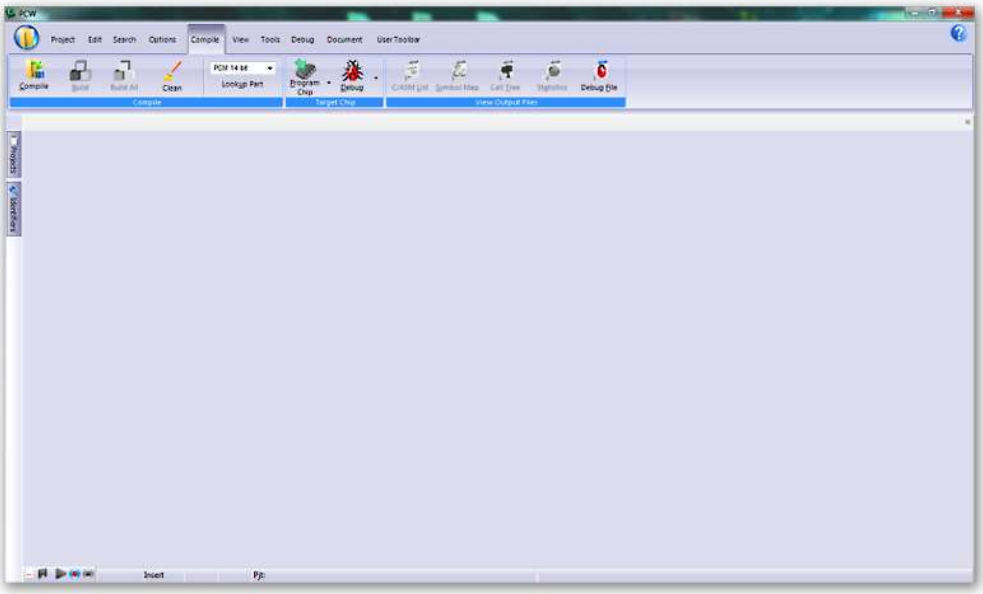
É um ambiente de desenvolvimento para microcrontroladores PIC.

Possui suporte para várias famílias de PIC.

Permite que o usuário escreva o programa em linguagem C.

A **Figura 13** mostra a tela inicial do CCS.

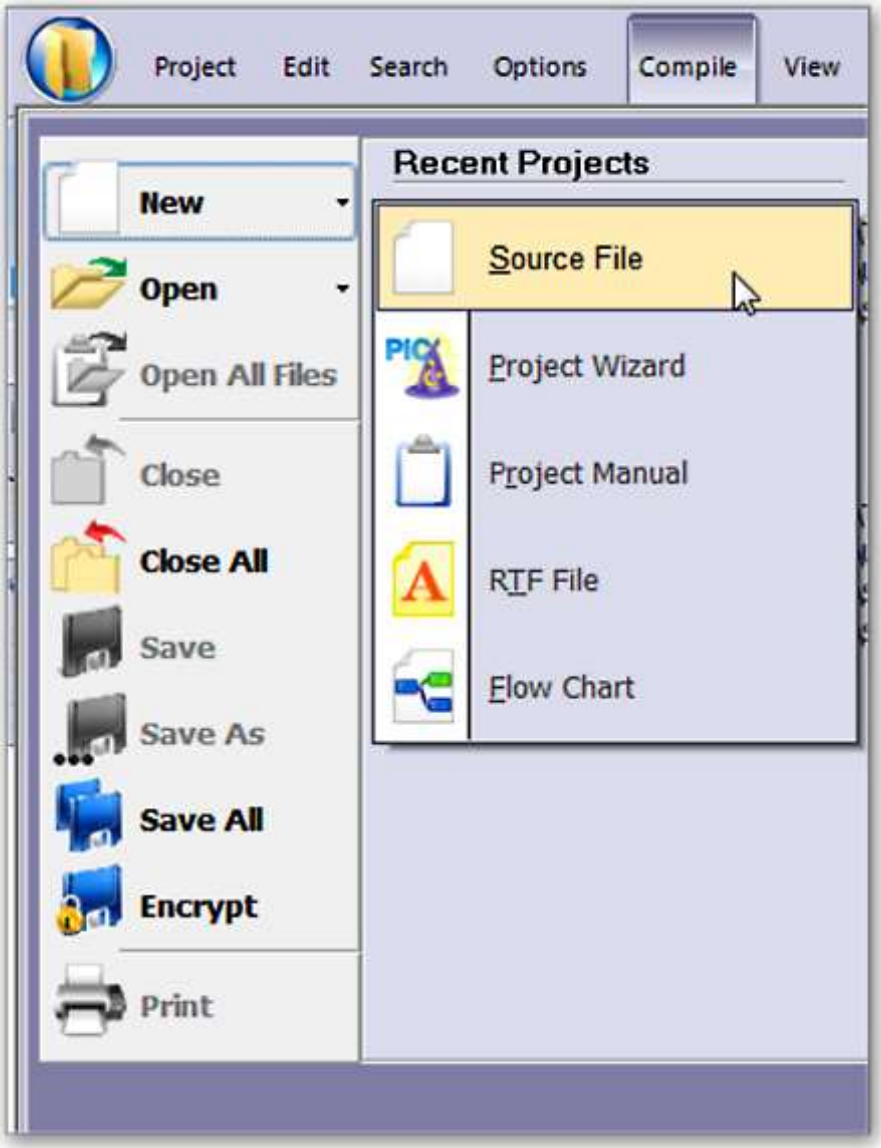
Figura 13 - Tela inicial do CSS



Inicialmente vamos em

e clicamos em **New** → **Source File** (**Figura 14**).

Figura 14 - Selecionando o arquivo fonte



- Escolhemos em que pasta vamos salvar e criamos o nosso arquivo “Semaforo_16F628A.c”.

- Agora, vamos digitar o nosso código em C, como exemplificado na **Figura 15**.

Figura 15 - Código do semáforo no CSS

Semaforo_16F628A.c *

```
#include <16f628a.h>            // Inclui o cabeçalho especifico do pic a ser utilizado

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz)
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES MCLR                     //Master Clear pin enabled
#FUSES NOCPD                    //No EE protection


#use delay(clock=20M)           // Seta o clock interno para 20Mhz

#define L_VERMELHO1  PIN_B4
#define L_VERMELHO2  PIN_B2

#define L_VERDE1     PIN_B1
#define L_VERDE2     PIN_B5

#define L_AMARELO1   PIN_B3
#define L_AMARELO2   PIN_B0


void main(void) // função principal
{
   do {
         /*
         S1 S2
         O  X
         O  O
         X  O
         */
         output_high(L_VERMELHO2);
         output_high(L_VERDE1);
         output_low(L_AMARELO2);
         output_low(L_VERMELHO1);
         delay_ms(3000);
         /*
         S1 S2
```

- Você deve escrever o código, conforme mostrado a seguir.

```
#include <16f628a.h>  // Inclui o cabeçalho especifico do pic a ser usado

#FUSES NOWDT         //No Watch Dog Timer
#FUSES HS           //High speed Osc (> 4mhz)
#FUSES NOPUT        //No Power Up Timer
#FUSES NOPROTECT         //Code not protected from reading
#FUSES BROWNOUT     //Reset when brownout detected
#FUSES MCLR         //Master Clear pin enabled
#FUSES NOCPD        //No EE protection
#use delay(clock=20M)     // Seta o clock interno para 20Mhz

#define L_VERMELHO1  PIN_B4
#define L_VERMELHO2  PIN_B2
#define L_VERDE1     PIN_B1
#define L_VERDE2     PIN_B5
#define L_AMARELO1   PIN_B3
#define L_AMARELO2   PIN_B0


void main(void) // função principal

{
    do {
        /*

        S1 S2
        O  X
        O  O
        X  O

        */

        output_high(L_VERMELHO2);
        output_high(L_VERDE1);
        output_low(L_AMARELO2);
        output_low(L_VERMELHO1);
        delay_ms(3000);

        /*

        S1 S2
        O  X
        X  O
        O  O

        */

        output_low(L_VERDE1);
        output_high(L_AMARELO1);
        delay_ms(1000);

```

```
52          /*
53
54          S1 S2
55          X  O
56          O  O
57          O  X
58
59          */
60
61          output_low(L_VERMELHO2);
62          output_high(L_VERDE2);
63          output_low(L_AMARELO1);
64          output_high(L_VERMELHO1);
65          delay_ms(3000);
66
67          /*
68
69          S1 S2
70          X  O
71          O  X
72          O  O
73          */
74
75          output_low(L_VERDE2);
76          output_high(L_AMARELO2);
77          delay_ms(1000);
78
79      } while (TRUE); // mantem o laço de repetição rodando em loop infinito
80  }
81
82  // FIM DO CÓDIGO
```

Quando estiver digitando esse código, você vai conseguir identificar:

1. **definição do PIC a ser utilizado;**
2. **comportamento dos fuses para o nosso projeto;**
3. **cristal;**
4. **portas de IO;**
5. **comentários;**
6. **função principal do nosso semáforo.**

Nesse código, verificamos que as portas de saída estão relacionadas a flags. 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, clicamos em

para que ele seja compilado.

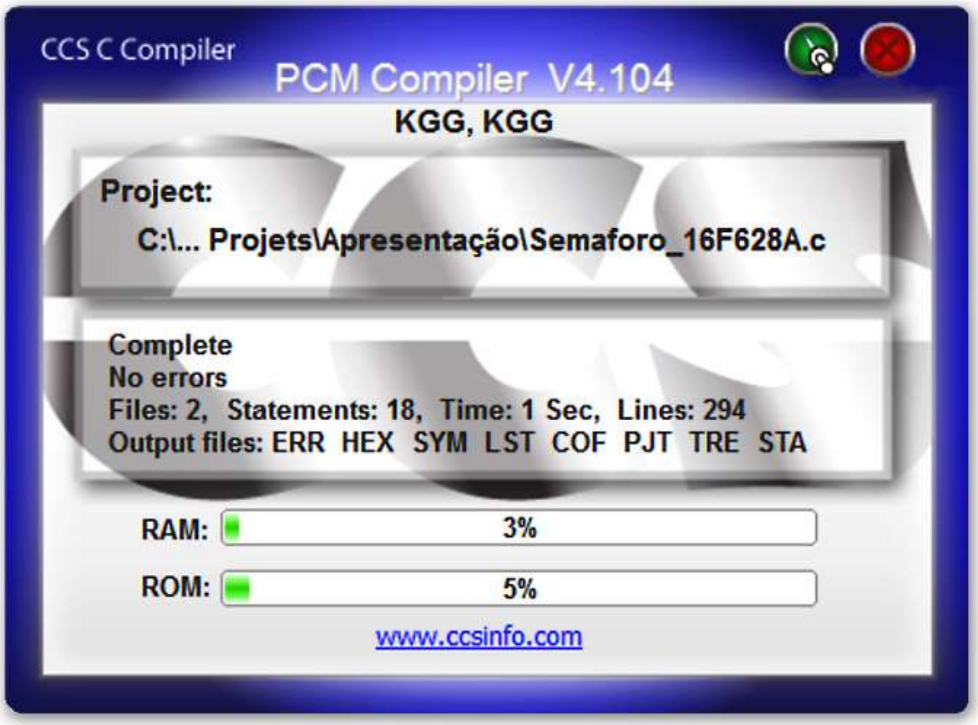
Arquivos de Saída

Você lembra dos arquivos de saída?

Aquele ".hex" que falamos tanto. Pronto! É justamente nesse ponto que ele será criado.

Caso o código digitado não tenha nenhum erro, uma mensagem igual a da **Figura 16** será mostrada.

Figura 16 - Resultado de compilação bem sucedida no CCS

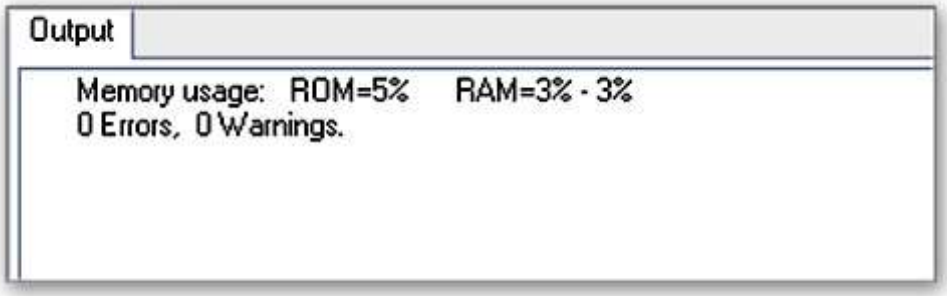


Na tela mostrada na **Figura 16**, podemos ver os arquivos que foram gerados: “.ERR”, “.HEX”, “.SYM”, dentre outros.

Esse arquivo “.hex” é o que utilizamos para gravar o nosso microcontrolador. Os outros arquivos têm funções específicas, como debugger (por exemplo), mas que não serão abordados aqui.

Por fim, na barra de *Output* (**Figura 17**), podemos ver também um relatório da nossa compilação, já que a outra tela anterior (**Figura 16**) desaparece rapidamente.

Figura 17 - Barra de saída (output) do CCS



Resumo

Nesta aula prática, usamos o MPLAB para importar o arquivo “.hex” e para programar nossa placa. Além disso, usamos o CCS para criar nosso arquivo em C, que implementa o semáforo. Vimos também como criar um exemplo de código para programar o microcontrolador do nosso circuito da forma como definimos.

Referências

CCS: custom computer services, Inc. Disponível em: <http://www.ccsinfo.com>. Acesso em: 24 out. 2012.

MICROCHIP. Disponível em: <http://www.microchip.com>. Acesso em: 24 out. 2012.