

**INTRODUÇÃO AO
FREESCALE PROCESSOR
EXPERT
PARA A DISCIPLINA EA076**

**Prof. Antonio A. F. Quevedo
FEEC / UNICAMP**

Revisado em setembro de 2014

INTRODUÇÃO

O CodeWarrior 10 (CW10) é o ambiente de desenvolvimento de *software* desenvolvido pela Freescale para os microcontroladores e microprocessadores por ela desenvolvidos. Ao contrário das versões anteriores, desenvolvidas de maneira totalmente proprietária, a versão 10 foi desenvolvida a partir do **Eclipse**.

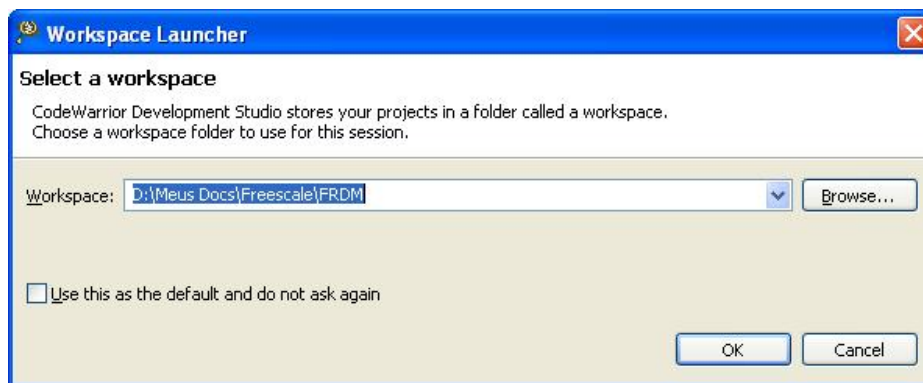
Um dos recursos presentes no CodeWarrior 10 é o **Processor Expert**. Esta ferramenta permite a geração automatizada de código de inicialização de periféricos e outros recursos, bem como funções de alto nível para o gerenciamento dos mesmos recursos. O sistema de *wizard* e GUI permite a rápida definição dos parâmetros de inicialização do periférico e a criação das funções de alto nível desejadas para sua utilização. Esta apostila parte do princípio que o leitor conhece o CodeWarrior 10, conforme mostrado na disciplina EA871. Para isto, existe uma apostila similar a esta, e recomenda-se a realização dos passos da mesma antes de experimentar o Processor Expert. Nesta apostila, as primeiras etapas serão idênticas às da outra apostila mencionada.

Neste passo-a-passo vamos mostrar como é fácil criar um programa que realiza o “Hello World” dos sistemas embarcados: fazer um LED piscar com frequência definida. Para isto, vamos criar dois **componentes**, que são “pacotes” compostos pela função de inicialização de um dado recurso e suas funções de alto nível.

Obs: A aparência de algumas janelas pode ser ligeiramente diferente em alguns computadores, mas as diferenças não são suficientes para prejudicar este tutorial.

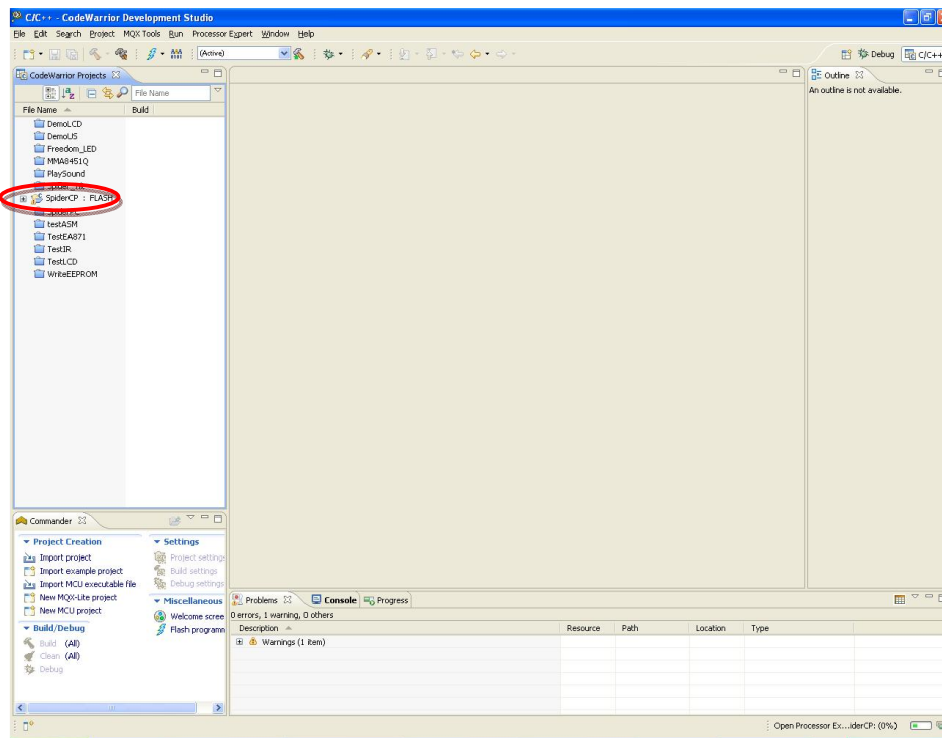
CRIANDO UM NOVO PROJETO

1. No Windows, vá ao menu Iniciar - Programas - Freescale CodeWarrior - CW for MCU v10.6 – CodeWarrior, ou chame o programa através do ícone do CodeWarrior 10.6 na área de trabalho.
2. Aparece o *Workspace Launcher*. Aqui você determina qual *workspace* você deseja utilizar. Clique em *Browse* para escolher uma pasta para ser seu *workspace* ou selecione uma já existente na lista *drop-down*. Clique em OK. Sugerimos que você crie seu próprio *workspace*.

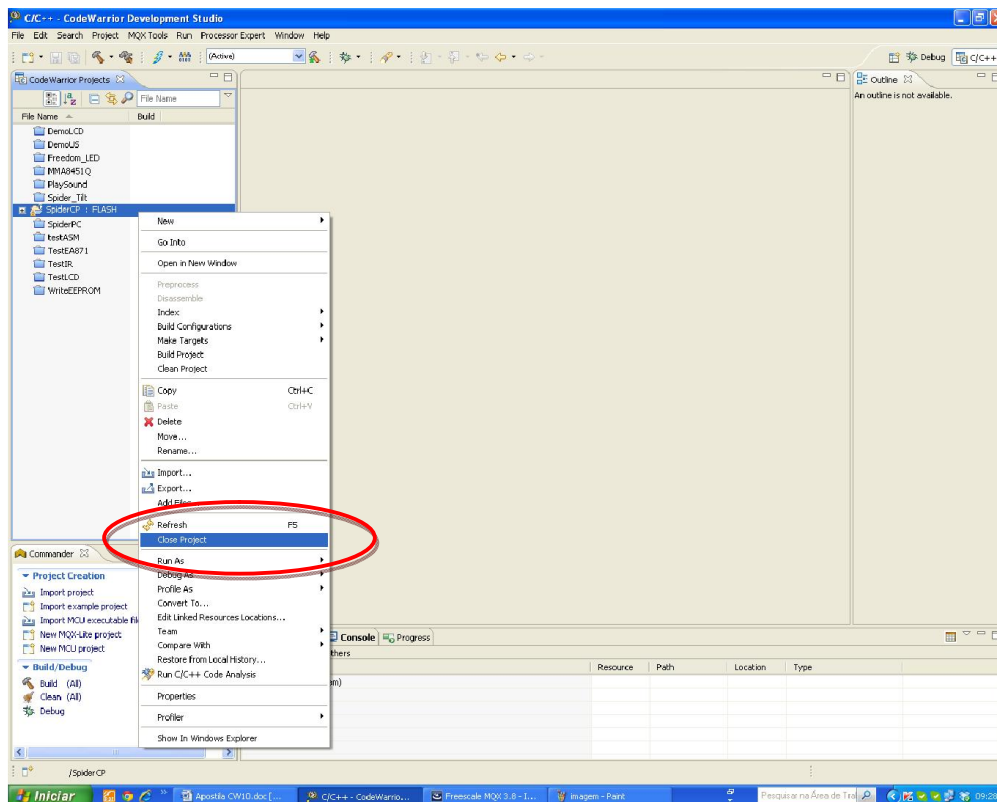


3. O CodeWarrior 10 vai abrir na perspectiva de programação. Pode-se ver a esquerda um painel com todos os projetos do *workspace*. Uma pasta aberta significa que aquele projeto está aberto.

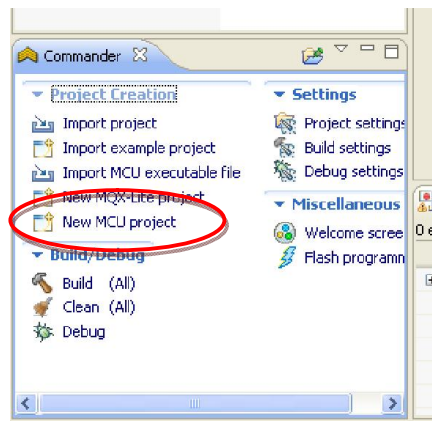
Na primeira utilização do *workspace*, ao invés da perspectiva de programação, pode aparecer um menu geral. Neste caso, basta escolher a opção “Go to Workspace”.



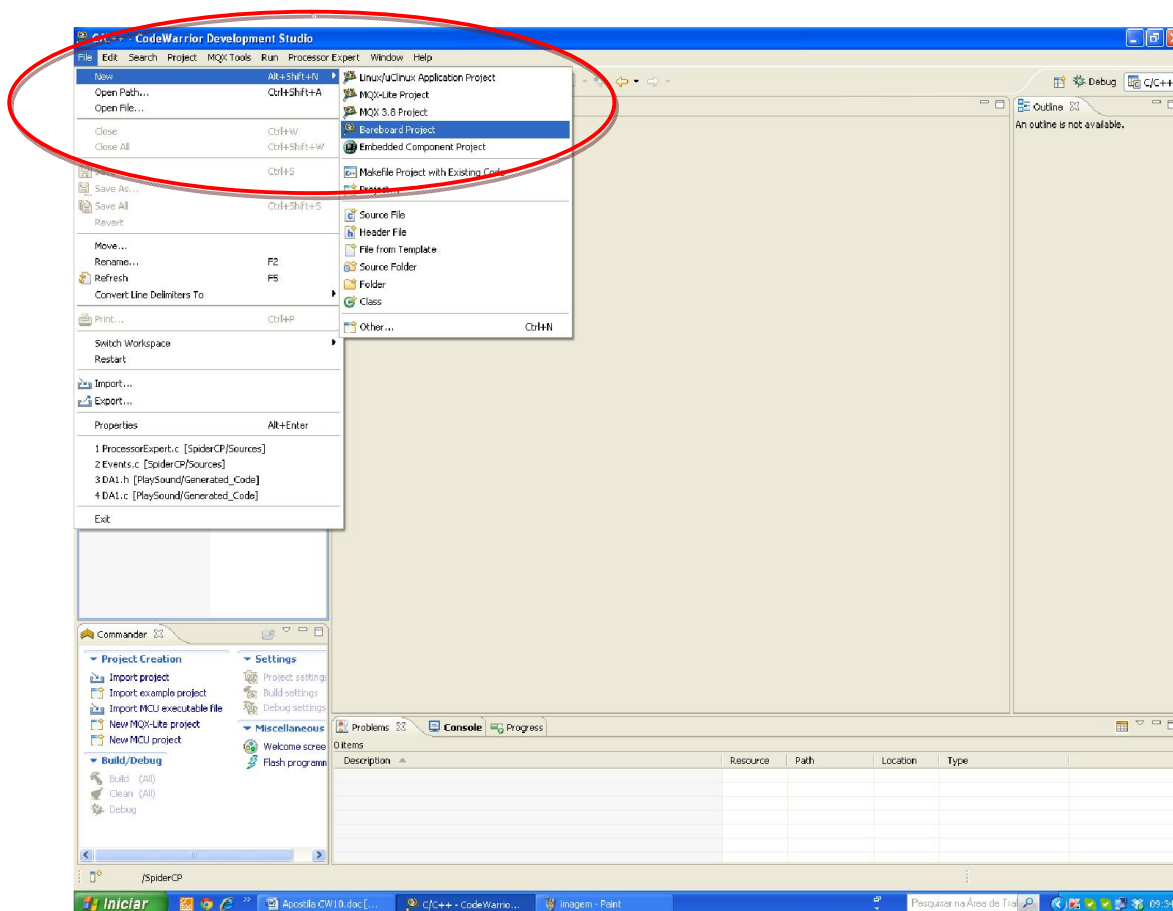
4. Para evitar problemas, evite ter mais de um projeto aberto de cada vez. Se houver diversos projetos abertos, feche os desnecessários clicando com o botão direito sobre a pasta do projeto aberto e escolhendo a opção "Close Project".



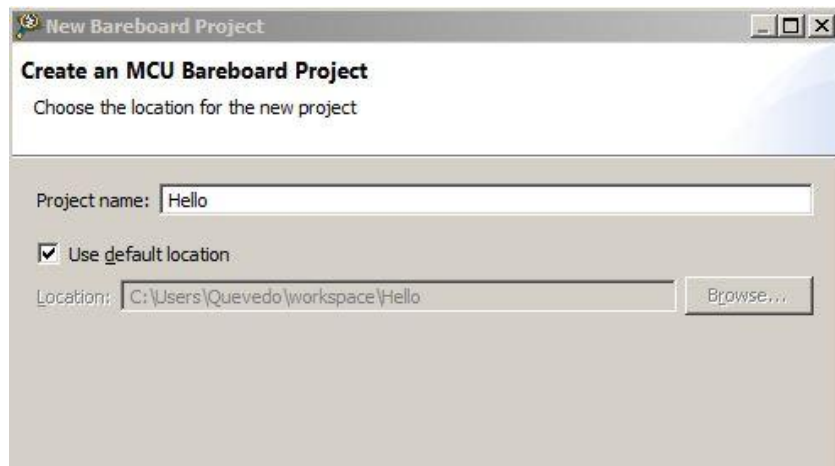
5. A janela *Commander* no canto inferior esquerdo da perspectiva agrupa os comandos mais comuns. Para criar um novo projeto, basta clicar em *New MCU project* nesta janela.



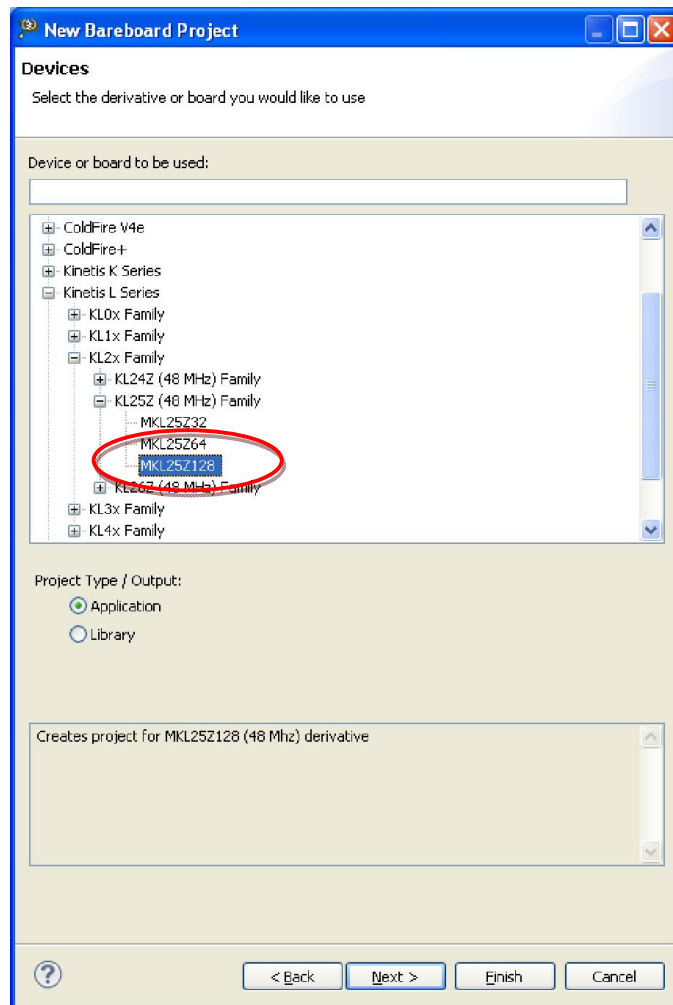
Alternativamente, no menu superior selecione *File - New - Bareboard Project*.



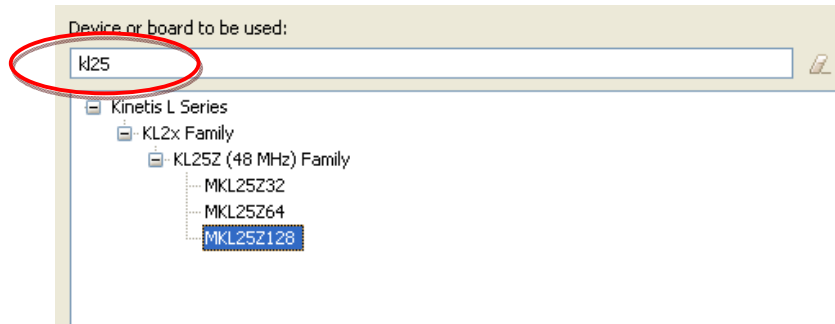
6. Na janela que se abre, escolha um nome para o projeto. No exemplo, foi escolhido o nome "Hello". Se a caixa de seleção "Use default location" estiver selecionada, os arquivos serão salvos na pasta do *workspace*, que pode ser vista logo abaixo da caixa. Desmarcando a caixa, pode-se escolher outra pasta para colocar os arquivos. Depois, clique no botão "Next".



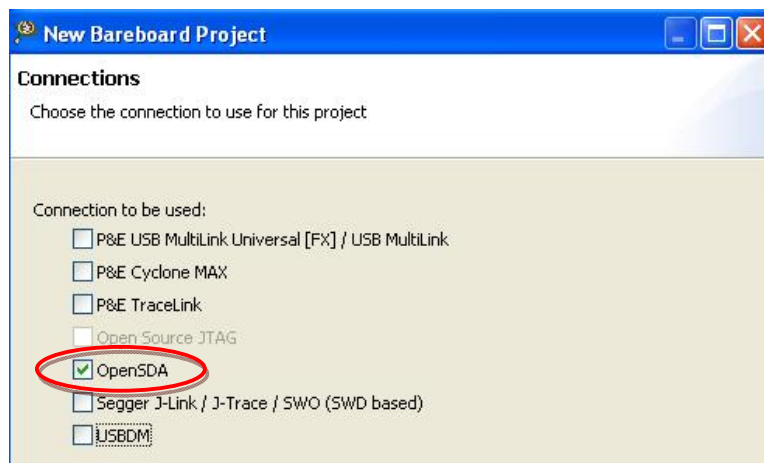
7. Agora deve-se escolher o microcontrolador ou a placa de desenvolvimento a ser utilizada. Neste curso usamos o controlador MKL25Z128, na placa FRDM-KL25. Na árvore de opções, abra o grupo "Kinetis L Series", depois o grupo "KL2x Family", depois "KL25Z (48MHz) Family" e seleccione "MKL25Z128" Clique "Next".



Alternativamente, pode-se digitar “KL25” na caixa de pesquisa (intitulada “Device or board to be used”). Neste caso, o programa filtra as opções, apresentando apenas a sub-árvore dos processadores KL25, e assim pode-se selecionar mais facilmente o processador-alvo.

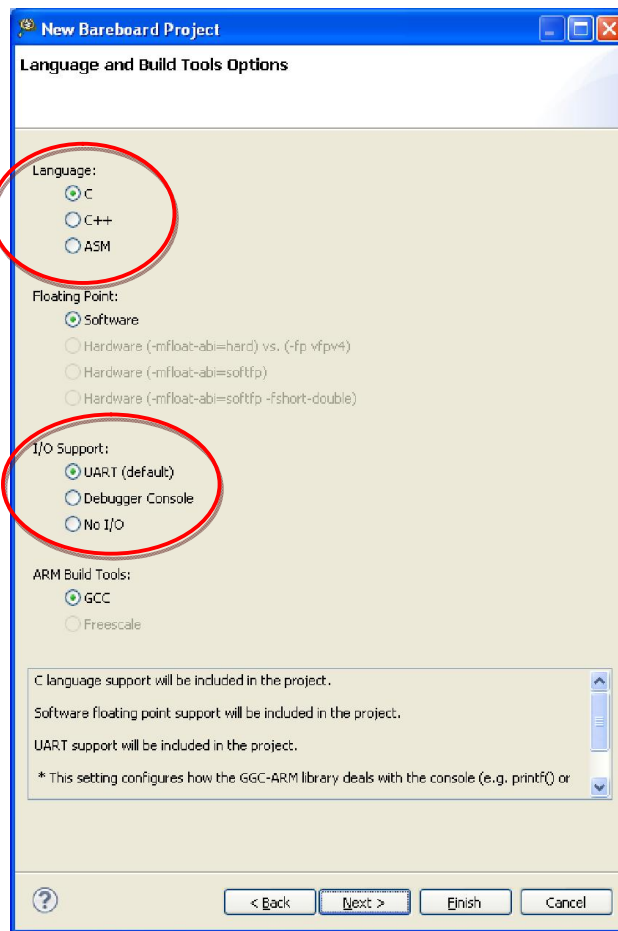


8. Agora, aparece uma lista de possíveis conexões. A conexão estabelecida entre o computador e a placa permite transferir os programas para a placa, bem como executar a depuração em tempo real. Neste caso, selecione a opção "OpenSDA", que é o *hardware* de conexão existente na placa de desenvolvimento. Desmarque quaisquer outras opções selecionadas por padrão, e depois clique em "Next".

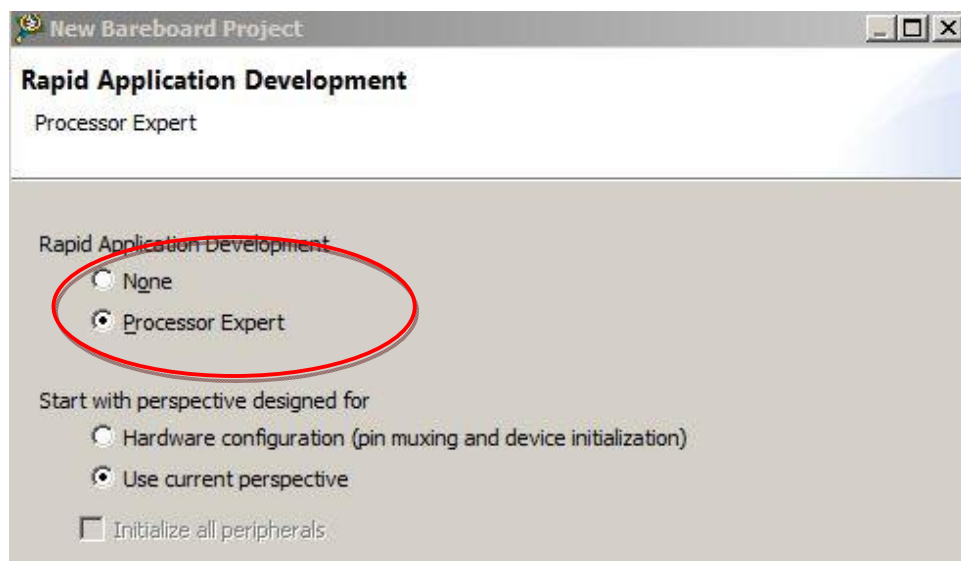


9. Na próxima janela, mantenha as opções no padrão e clique em "Next". Estas opções se referem à linguagem utilizada, uso de ponto flutuante, suporte a uso de porta serial ou console de debug, e compilador a ser usado.

Obs: Na opção “I/O Support”, a opção *default* (UART) e a opção *Debugger Console* acrescentam código para suporte a funções *printf* e *scanf* através da porta serial ou de uma janela de console. Quando estas funções não são necessárias, pode-se marcar a opção “No I/O” para reduzir o tamanho do código final e se evitar inicializações desnecessárias. Sugerimos usar a terceira opção, a menos que o experimento use as funções acima citadas (não é o caso no curso EA871).

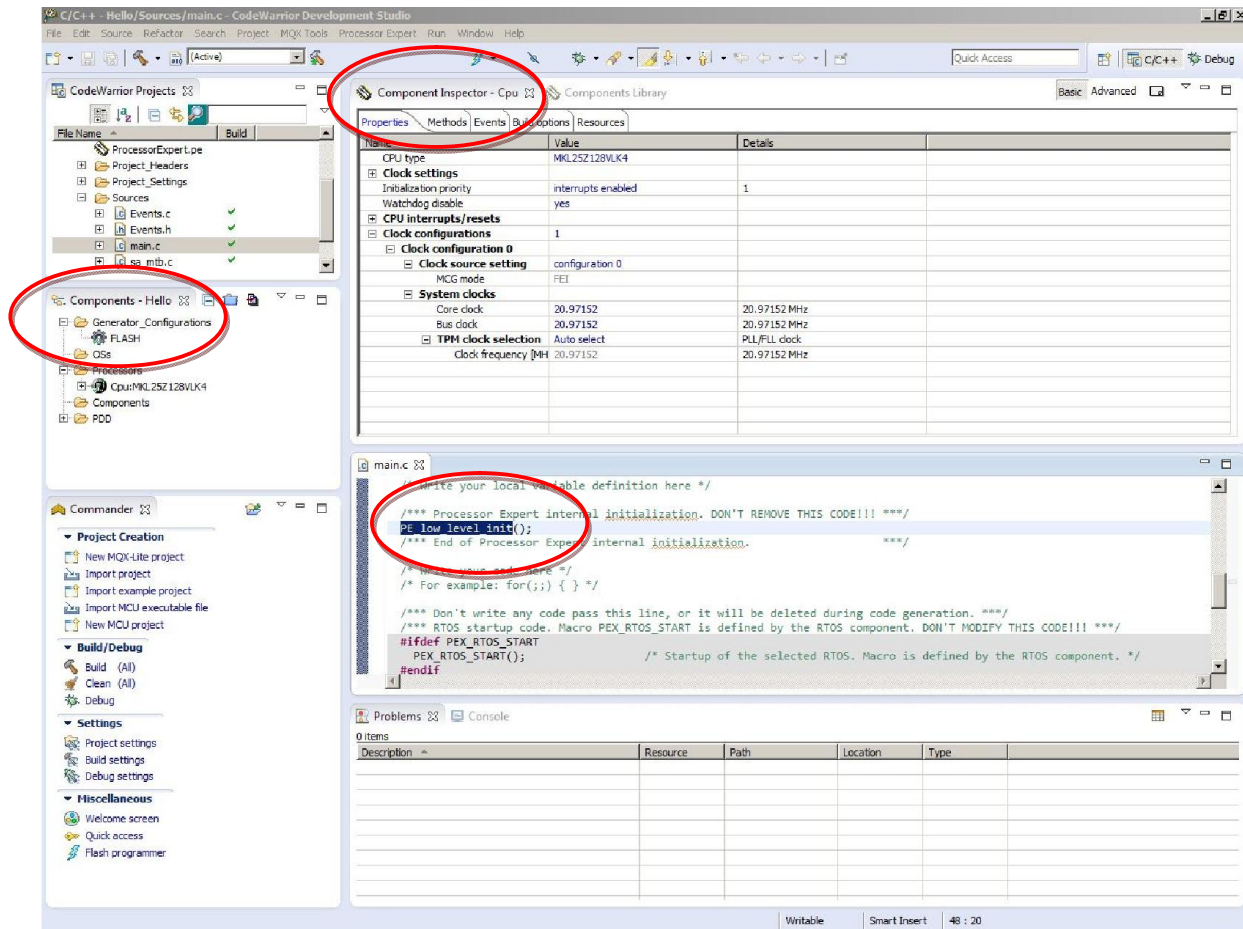


10. Nesta próxima janela, definiremos se o CodeWarrior usará ou não o *Processor Expert*, que é o nosso objetivo nesta apostila. Selecione a opção "Processor Expert" e clique em "Finish".

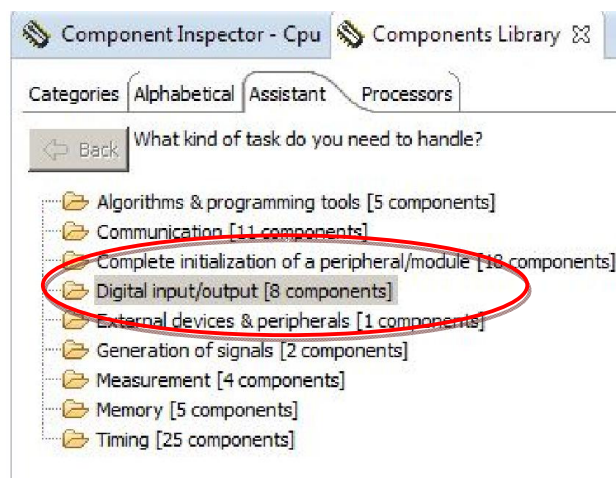
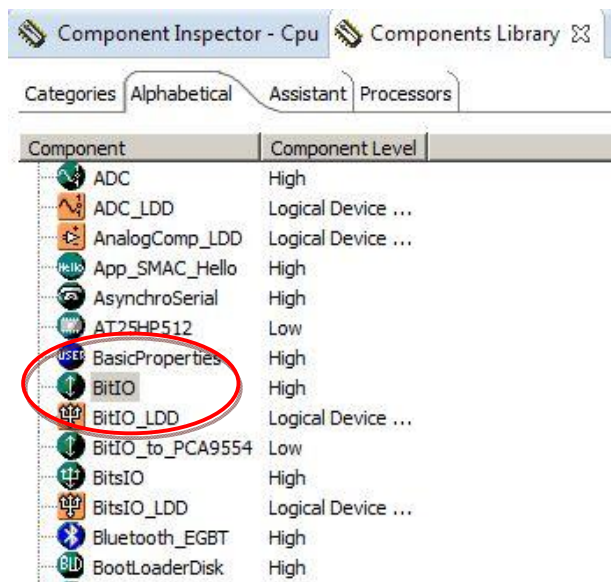
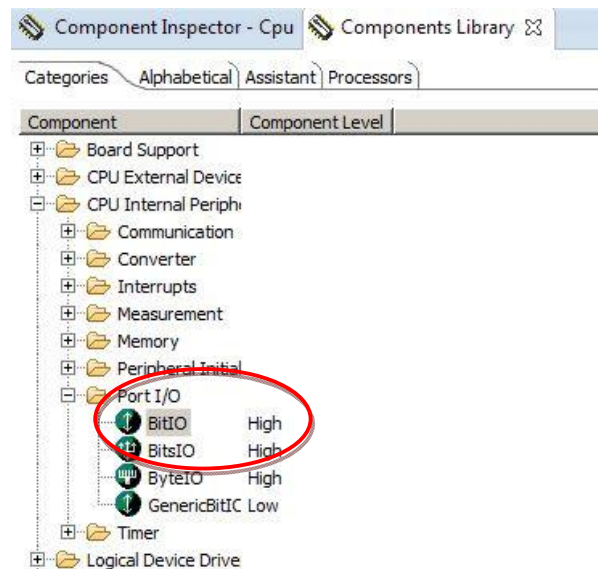


11. Veja que agora na lista de projetos, temos o projeto "Hello" criado e aberto. Clicando no símbolo de "+" (ou no pequeno triângulo se estiver usando Windows 7/8) à esquerda do folder, podemos expandir o projeto e ver seus componentes pré-criados pelo CodeWarrior. Dentro do folder "Sources", encontraremos o arquivo "main.c". Dando um duplo-clique neste arquivo, abrimos

o editor e podemos ver uma estrutura básica de código já feita pelo CodeWarrior, onde são incluídas algumas bibliotecas e a função “main” chama a função “PE_low_level_init()”. Esta função está definida em outro arquivo, e conterà chamadas a todas as funções de inicialização (uma para cada componente do PE). Acima da janela de edição, pode-se ver a GUI do Processor Expert, na aba “Component Inspector”. À esquerda, existe uma pequena janela chamada “Components”, que possui a relação de todos os componentes implementados. Pode-se ver que já foi criado um componente chamado CPU, e que o Component Inspector está apresentando as opções para este componente. Aqui pode-se definir alguns parâmetros básicos de operação da CPU, como, por exemplo, a fonte e frequência de *clock*. Neste projeto, deixaremos este componente inalterado.

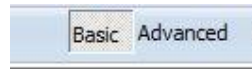


12. Vamos agora criar os componentes necessários para o projeto. Para tanto, deve-se clicar na aba “Components Library”. Aparecerá uma listagem de componentes disponíveis. No primeiro uso, os componentes são apresentados por categoria. Pode-se escolher outras formas de classificação ou utilizar um assistente, dependendo da aba. Para controlar o LED da placa, vamos criar um pino de saída digital, usando o componente “BitIO”. Podemos encontrá-lo na aba por categorias, sob “CPU Internal Peripherals” – “Port I/O”. Pode-se ainda localizá-lo por ordem alfabética, ou usar o assistente respondendo às perguntas feitas pelo mesmo.



13. Após a seleção, será criado o componente “Bit1” do tipo “BitIO” (veja no painel de componentes à esquerda). O nome fornecido é padrão e pode ser modificado na GUI. Vamos definir os parâmetros e funções do componente. Selecione o componente no painel correspondente, e clique na aba do Component Inspector (veja que no título da aba aparece um “*”, indicando que os parâmetros ainda não foram salvos).

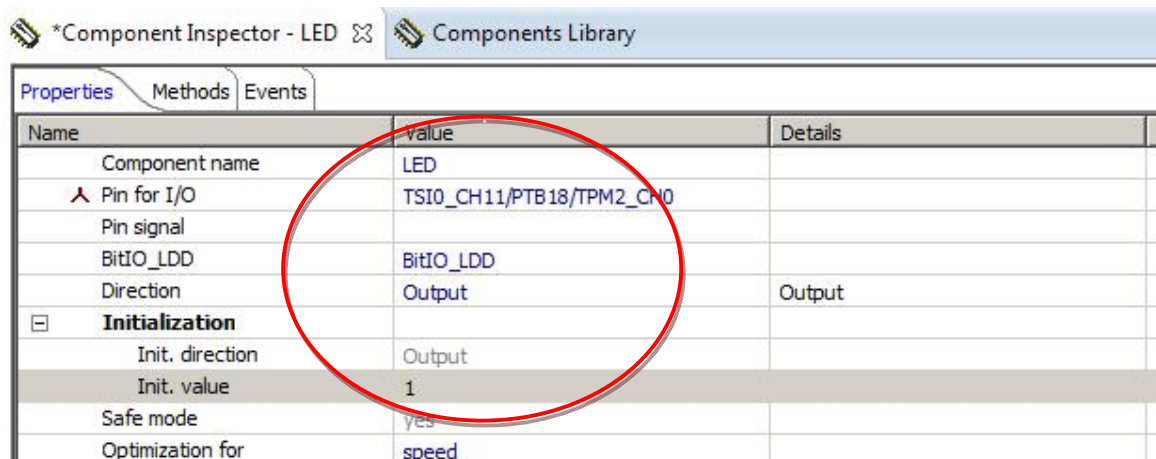
14. No canto superior direito do Component Inspector, existem 2 botões que definem a configuração básica ou avançada. No modo básico não podemos mudar o nome do componente, assim usaremos o modo avançado, com mais recursos.



15. Veja que o Inspector possui 3 sub-abas: Propriedades, Métodos e Eventos. Vamos inicialmente definir as propriedades do componente. Clique no campo ao lado de “Component name” e mude o nome para “LED”. Outras opções exibem caixas do tipo “drop-down” quando selecionadas.

Queremos um pino que seja exclusivamente de saída, ligado ao LED vermelho da placa FRDM (olhando no manual da placa, vemos que o PTB18 é ligado no catodo do LED vermelho). Sabemos que o LED acende se o pino estiver em 0, então queremos que ao iniciar o mesmo fique no nível 1 para manter o LED apagado. Assim, definiremos o seguintes valores:

- Pin for I/O: PTB18 (o nome do pino é mais complexo, mas pode-se clicar no campo de valor e escrever “ptb18” que a opção correta aparecerá).
- Direction: Output
- Initialization – Init. Value: 1

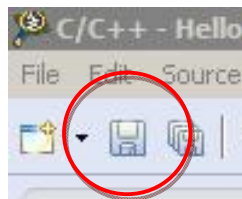


16. Vamos agora definir quais as funções de alto nível (aqui chamadas de **métodos**) que serão criadas pelo PE. Clique na aba de Métodos e aparecerá uma lista de possíveis funções, tendo ao lado a informação se elas devem ou não ser criadas. Ao colocar o cursor sobre o nome de cada função, é apresentada uma janela mostrando a sua utilização. Nosso exemplo demanda apenas a função “NegVal”, que inverte o estado do pino. Selecione “generate code” para esta função e “don’t generate code” para todas as outras. Não há problemas em gerar uma função que não será usada, mas isto produzirá um “warning” na compilação, e a geração do código para esta função irá

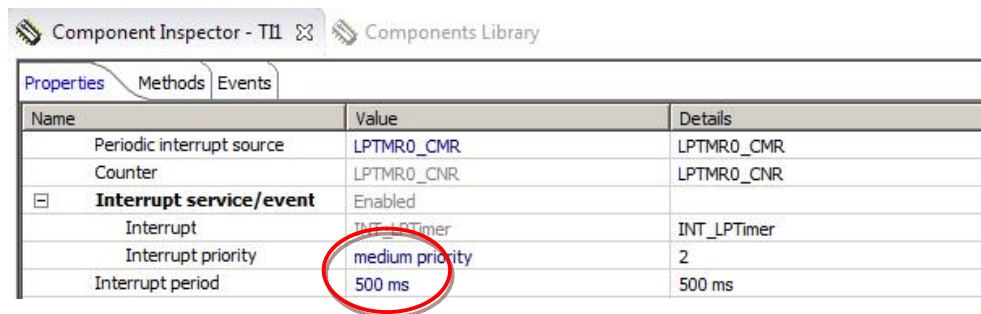
aumentar o tamanho do código-objeto. Por isso, é uma boa prática desativar as funções desnecessárias.

Properties Methods Events		
Name	Value	
GetDir	don't generate code	
SetDir	don't generate code	
SetInput	don't generate code	
SetOutput	don't generate code	
GetVal	don't generate code	
PutVal	don't generate code	
ClrVal	don't generate code	
SetVal	don't generate code	
NegVal	generate code	
ConnectPin	don't generate code	
GetRawVal	don't generate code	

17. Para preservar as configurações feitas, vamos salvar o componente. Para isto, clique no botão “Save” (símbolo de disquete) no canto superior esquerdo. Veja que o asterisco na aba do Inspector sumiu.

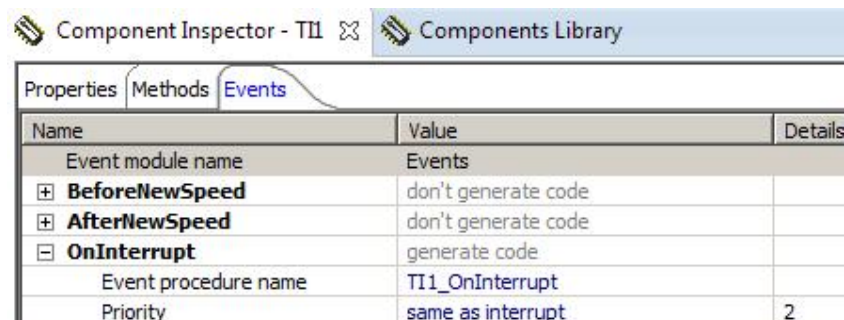


18. Vamos criar agora o segundo componente, que é um *timer* que gera uma interrupção periódica. Assim, a cada interrupção, iremos chamar a função “NegVal” para inverter o estado do LED. Na Components Library, selecione o componente “TimerInt”. Voltando ao Component Inspector, selecione a aba de Propriedades, e veja que o nome do componente foi definido como “TI1”. Podemos deixá-lo assim. Veja que no painel dos componentes, o TI1 aparece com um “x” vermelho, e também aparece uma pasta de “Referenced_components”, também com o “x” vermelho. Isto acontece porque já parâmetros obrigatórios não definidos em nosso componente. Além disso, este componente é criado sobre outro componente (“referenced”), que é um LDD (Low-level Device Driver) para o timer. Alguns componentes exigem a configuração de seu LDD, mas no geral a configuração do componente de mais alto nível já configura o LDD, o que é este caso. Assim, tendo selecionado o componente TI1 e a aba de propriedades no Inspector, vamos configurá-lo. Podemos selecionar qual o *timer* a ser usado (“periodic interrupt source”), mas neste caso vamos manter o que foi definido pelo PE. Em “Interrupt period” selecione “500ms”, para que o LED mude de estado a cada meio segundo, gerando assim um período de 1s. Pode-se experimentar outros valores se desejado. Veja que ao se definir o período, os “x” vermelhos desaparecem.

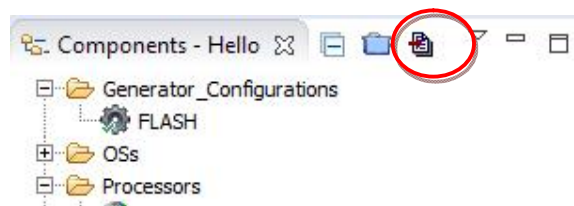


19. Não vamos mexer na aba de Métodos, pois o padrão é não gerar nenhuma função, e elas não serão necessárias. Pode-se, por exemplo, definir funções para habilitar e desabilitar a interrupção periódica. Como nas propriedades está definido (por padrão, podendo-se mudar o parâmetro) que a interrupção periódica é habilitada na inicialização, estas funções não são necessárias.

20. Neste componente, vamos explorar a aba de Eventos. Um evento é todo processo que pode ser gerado pelo componente, geralmente uma interrupção. Vamos manter as opções padrão, sabendo-se que podem ser alteradas. Veja que o “Event module name” se chama “Events”. Isto significa que será criado um arquivo “Events.c” que conterá a função de tratamento dos eventos deste componente. Veja ainda que será gerado código para a condição “On Interrupt”, ou seja, para o tratamento da interrupção gerada pelo componente. No campo de “Event procedure name” temos “TI1_OnInterrupt”, que é o nome padrão para esta função (composto pelo nome do componente e pelo tipo de evento).

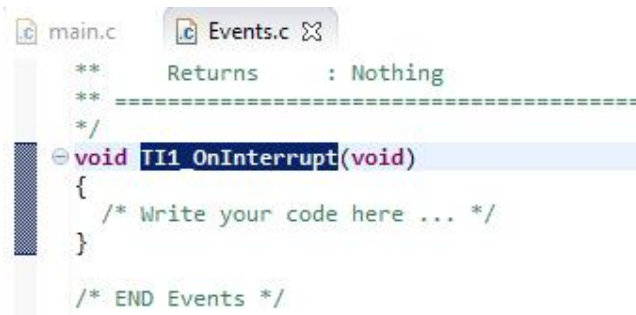


21. Agora que temos todos os componentes definidos, precisamos fazer com que o PE crie o código. Para tanto, no painel de componentes, clique no botão que simboliza uma lista com uma seta vermelha (*Generate Processor Expert Code*). Os códigos-fonte de inicialização, dos métodos e das funções dos eventos serão gerados.



22. Após a geração do código, o arquivo “main.c” permanece o mesmo. Veja que na pasta “Sources” aparecem agora outros arquivos. O arquivo “Events.c” contém a função de tratamento da interrupção. Dê um duplo clique sobre ele para abri-lo no editor.

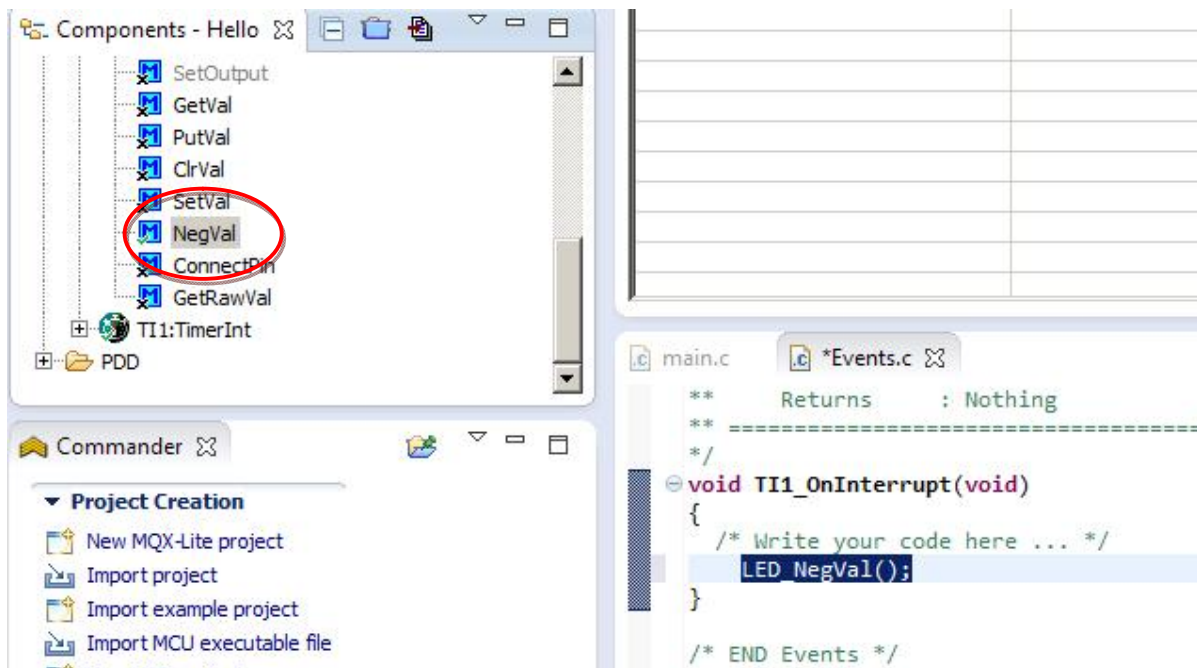
23. Na listagem, procure a função “TI1_OnInterrupt”. Veja que ela está vazia, e há um comentário para que o usuário coloque seu código ali. A única coisa a ser feita por esta função (a limpeza do *flag* de interrupção é feita automaticamente pelo código gerado) é chamar a função “NegVal” do componente LED (ou seja, “LED_NegVal”). Podemos simplesmente escrever “LED_NegVal();” dentro da função de tratamento, mas há um método mais simples.



```
main.c  Events.c
**      Returns      : Nothing
**      =====
*/
void TI1_OnInterrupt(void)
{
    /* Write your code here ... */
}

/* END Events */
```

24. No painel dos componentes, expanda o componente LED. Você verá todas as funções possíveis para este componente. Entre elas, está a “NegVal”. Clique sobre ela e arraste o cursor com o botão apertado, arrastando a função para dentro da função “TI1_OnInterrupt”. Verifique se o cursor está dentro da função OnInterrupt antes de soltar o botão. Ao soltar, será colocada a chamada da função “LED_NegVal” no local do cursor.



25. Pronto! Já temos tudo o que precisamos para fazer o programa funcionar. Agora basta compilar o projeto, carregar na placa e executar o programa, conforme já visto na apostila do CodeWarrior!

Revisado pelo Prof. Antonio Quevedo em 04/09/2014

As janelas apresentadas são de telas do CodeWarrior 10.4, executando em Windows XP, e do CodeWarrior 10.6, executado em Windows 7. Pequenas diferenças podem aparecer em futuras versões do programa, e em outros sistemas operacionais.