

Projetos no IAR EWARM

Prof. Hugo Vieira Neto

2021/1

Familiarização com Kit e IDE

- Objetivo: colocar em funcionamento o projeto “simple_io_main_sp” da área de trabalho “TM4C1294_Bare_IAR9”
 - Verificar configurações da compilação
 - Verificar configurações da ligação
 - Compilar, ligar e executar o código da aplicação

Familiarização com Kit e IDE

- Clonar o repositório “TM4C1294_Bare_IAR9”
(https://github.com/ELF74-SisEmb/TM4C1294_Bare_IAR9)
- Abrir o arquivo `TM4C1294_Bare_IAR9.eww` no IAR EWARM
- Selecionar o projeto “simple_io_main_sp” e torná-lo ativo (botão direito do mouse)
- Compilar o projeto ativo
- Executar o projeto ativo no hardware do kit

Familiarização com o Depurador

- Experimentar o depurador com o simulador ou com o kit de desenvolvimento:
 - Configuração da conexão
 - Carregamento do código em memória flash
 - Execução controlada (passo-a-passo, breakpoints)
 - Execução do código em C e Disassembly
 - Inspeção de memória e de variáveis
 - Inspeção de registradores (CPU e periféricos)
 - Inspeção da pilha, terminal de I/O, etc.

Criação de um Novo Projeto

- Selecionar o subdiretório da área de trabalho “TM4C1294_Bare_IAR9”
- Criar um subdiretório para o novo projeto dentro do subdiretório “Projects”
- Criar um novo projeto dentro da área de trabalho da IDE
 - Selecionar Menu Project → Create new project...
 - Selecionar **Empty Project**

Criação de um Projeto

- Salvar o arquivo de projeto (extensão .ewp) no subdiretório de projeto recém-criado
 - Sugere-se utilizar o mesmo nome para o arquivo e para o subdiretório do projeto
- Criar um subdiretório “src” para armazenar os arquivos do código-fonte do novo projeto
- Copiar os arquivos do subdiretório “template” para o subdiretório “src” recém-criado

Arquivos do Projeto

- Clicar com o botão direito do mouse sobre o projeto recém criado na área de trabalho do EWARm e adicionar:
 - O arquivo de inicialização `startup_TM4C1294.s`
 - O arquivo de sistema `system_TM4C1294.c`
(se a biblioteca driverlib ***não for*** utilizada)
ou o arquivo de sistema `system_TM4C1294_TW.c` (se a biblioteca driverlib TivaWare ***for*** utilizada)
 - Seus próprios arquivos de código-fonte para a aplicação do projeto (ASM, C ou C++)

Arquivos do Projeto

- Se a biblioteca “driverlib” for utilizada no projeto, adicionar o seu código-objeto:
 - `driverlib.a`
- Observação: a localização do código-objeto da biblioteca pode ser encontrada no projeto “simple_io_main_sp”.

Opções do Projeto

- Clicar com o botão direito do mouse sobre o projeto recém criado e selecionar Options...
- General Options
 - Target → Device: Texas Instruments
TM4C1294NCPDT
 - Output file → Executable
 - Library Configuration → Library: Normal
 - Library Configuration → CMSIS: ☒ Use CMSIS

Opções do Projeto

- C/C++ Compiler
 - Preprocessor → Additional include directories:
`$PROJ_DIR$\..\..\TivaWare_C_Series-2.1.4.178`
- Linker
 - List: ☒ Generate linker map file
- Debugger
 - Setup → Driver: TI Stellaris
 - Setup → Download: ☒ Use flash loader(s)

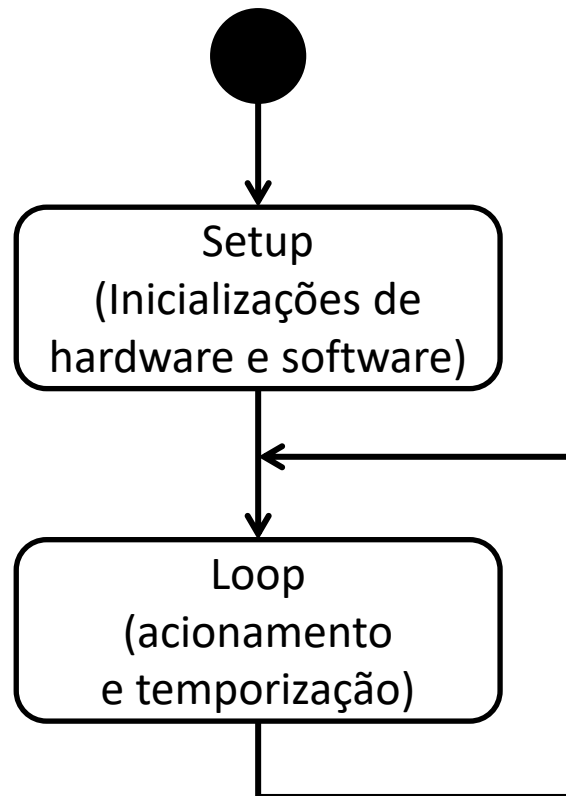
Laboratório 01

- Com base no projeto “simple_io_main_sp” da área de trabalho “TM4C1294_Bare_IAR9”, crie um novo projeto (nome: “Laboratório_01”) para uma aplicação com as seguintes especificações:
 - Frequência de clock (PLL) da CPU: 24MHz
 - Nível de otimização do compilador C: baixo (low)
 - O LED D4 deve trocar de estado a cada 1s
 - A temporização deve ser feita por software (laços de atraso), isto é, **sem o uso de qualquer mecanismo de interrupção por hardware**

Laboratório 01

- Depois de ter verificado a temporização por laços de atraso, teste o comportamento do sistema para as seguintes alterações:
 1. Diferentes níveis de otimização do compilador C
 2. Frequência de clock (PLL) de 120MHz
- Há variações na temporização por software para os casos acima? Quantifique-as.

Ideia Geral do Laboratório 01



Ideia Geral do Laboratório 01

- Setup:
 - Habilitar os GPIO ports (System Control)
 - Configurar os terminais de GPIO
- Loop:
 - Trocar estados dos terminais de GPIO
 - Gerar atrasos por software (laços)
 - Repetir o processo
- Calibrar as constantes dos laços de atraso para aproximadamente 1s

Importante

- Para entendimento adequado do uso das funções da biblioteca driverlib utilizadas no projeto “simple_io_main_sp”, consulte o manual TivaWare driverlib, especialmente:
 - Capítulo 1 (Introduction)
 - Capítulo 2 (Programming Model)
 - Capítulo 14 (GPIO)
 - Capítulo 26 (System Control)

Biblioteca TivaWare

- Diretório “TivaWare_C_Series-2.1.4.178”
- Analise o conteúdo dos arquivos:
 - `inc/hw_memmap.h`
 - `inc/hw_gpio.h`
 - `inc/hw_sysctl.h`

Driverlib – GPIO

- **API:**
 - `driverlib/gpio.h`
- **Principais funções:**
 - `GPIOPinTypeGPIOInput`
 - `GPIOPinTypeGPIOOutput`
 - `GPIOPadConfigSet`
 - `GPIOPinRead`
 - `GPIOPinWrite`

Driverlib – SYSCTL

- **API:**
 - `driverlib/sysctl.h`
- **Principais funções:**
 - `SysCtlClockFreqSet`
 - `SysCtlPeripheralEnable`
 - `SysCtlPeripheralReady`

Clareza e Legibilidade

- Os seguintes trechos de código são equivalentes:
 - `GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);`
 - `GPIOPinWrite(0x40025000, 0x00000010, 0x00000010);`
- Qual dos trechos de código acima é mais legível e fácil de se compreender?

Clareza e Legibilidade

- Os seguintes trechos de código são equivalentes:
 - `GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 | GPIO_PIN_4);`
 - `GPIOPinTypeGPIOOutput(0x40025000, 0x00000011);`
- Qual dos trechos de código acima é mais legível e fácil de se compreender?
- **Obs:** `GPIO_PIN_0=0x01; GPIO_PIN_4=0x10`