Configuration of New Projects in IAR EWARM

Prof. Hugo Vieira Neto

2020/1

Creation of a Project

- Create a folder with the name of the project in the "Projects" folder
- Create the new project in the "EK-TM4C1294_IAR8" workspace
  - Select Menu Project → Create new project...
    - Select **Empty Project**
- Save the project file inside the recently created folder

Creation of a Project

- Create a "src" folder to store the source-code files of the new project
- Copy the files from the "template" folder to the recently created "src" folder

Project Files

- Click with the right mouse button over the recently created project in the workspace and add the files :
  - Startup file `startup_TM4C1294.s`
  - **Either** system file `system_TM4C1294.c` (if the driverlib library ***will not*** be used) **or** system file `system_TM4C1294_TW.c` (if the driverlib library ***will*** to be used)
  - Your own source-code files for the project application (ASM, C or C++)

Project Files

- If the "driverlib" library will be used in the project, add its object-code:
  - `driverlib.a`
- Note: the library's object-code location may be found in the "simple_io_main_sp" project.

Project Options

- Click with the right mouse button over the recently created projecto and select Options...
- General Options
  - Target → Device: Texas Instruments TM4C1294NCPDT
  - Output file → Executable
  - Library Configuration → Library: Normal
  - Library Configuration → CMSIS: ☑Use CMSIS

Project Options

- 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)

Pre Lab Work 1 Exercise

- Having the "simple_io_main_sp" project from the "EK-TM4C1294_IAR8" workspace as basis, create a new project for an application with the following specifications:
  - CPU clock frequency (PLL): 24MHz
  - C compiler optimization level: low
  - LED D4 must change state every 500ms
  - Timing must be performed by software (delay loops), that is, without using any hardware interrupt mechanism

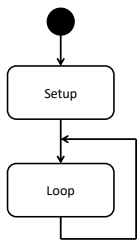
Pre Lab Work 1 Exercise

- In order to measure driving times accurately, besides driving LED D4, simultaneously drive some other pin from GPIO Port K (available in the kit's BoosterPack 2 interface connectors)
- With the aid of an oscilloscope connected to the pin from GPIO Port K, calibrate the software delay loops in order to obtain the highest possible accuracy on timing

Pre Lab Work 1 Exercise

- After having calibrated the delay loops, redo timing measurements for the following cases:
  1. Different C compiler optimization levels
  2. CPU clock frequency (PLL) of 120MHz
- Are there variations in software timing for the cases above? Quantify them.

Pre Lab Work 1 General Idea



Pre Lab Work 1 General Idea

- Setup:
  - Enable GPIO ports (System Control)
  - Configure GPIO pins
- Loop:
  - Change states of GPIO pins
  - Generate software delays (loops)
  - Repeat the process
- Calibrate the constants of the delay loops with the aid of the oscilloscope

Important

- For proper understanding of driverlib library functions used in the “simple_io_main_sp” project, check the TivaWare driverlib manual, especially:
  - Chapter 1 (Introduction)
  - Chapter 2 (Programming Model)
  - Chapter 14 (GPIO)
  - Chapter 26 (System Control)

TivaWare Library

- “TivaWare_C_Series-2.1.4.178” folder
- Analyze the contents of the files:
  - `inc/hw_memmap.h`
  - `inc/hw_gpio.h`
  - `inc/hw_sysctl.h`

Driverlib – GPIO

- API:
  - `driverlib/gpio.h`
- Main functions:
  - `GPIOPinTypeGPIOInput`
  - `GPIOPinTypeGPIOOutput`
  - `GPIOPadConfigSet`
  - `GPIOPinRead`
  - `GPIOPinWrite`

Driverlib – SYSCTL

- API:
  - `driverlib/sysctl.h`
- Main functions:
  - `SysCtlClockFreqSet`
  - `SysCtlPeripheralEnable`
  - `SysCtlPeripheralReady`

Clarity and Legibility

- The following code snippets are equivalent:
  - `GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);`
  - `GPIOPinWrite(0x40025000, 0x00000010, 0x00000010)`
- Which of the code snippets above is more legible and easier to understand?

Clarity and Legibility

- The following code snippets are equivalent :
  - `GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 | GPIO_PIN_4);`
  - `GPIOPinTypeGPIOOutput(0x40025000, 0x00000011)`
- Which of the code snippets above is more legible and easier to understand?
- Note: `GPIO_PIN_0=0x01;GPIO_PIN_4=0x10`