MIDDLE EAST TECHNICAL UNIVERSITY

ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT

EE447

Introduction to Microprocessors

Laboratory Manual

Course Instructors:
Prof. Dr. Gözde B. Akar
Prof. Dr. İlkay Ulusoy

Laboratory Assistants:
Doğu Erkan Arkadaş
Ferhat Gölbol
Kamil Sert

October 2022

Contents

1 General Information about Laboratory Work **3**
1.1 Experimental Setup . . . 3
1.2 Preliminary Work . . . 5
1.3 Laboratory Work Performance Session . . . 5
1.4 Feedback . . . 6
1.5 Evaluation . . . 7
1.5.1 Preliminary Work Reports . . . 7
1.6 Laboratory Work Performance Sessions . . . 7

2 Laboratory Regulations **8**
2.1 Rules . . . 8
2.2 Cheating . . . 8

3 Introducing Keil Microcontroller Development Kit (MDK) and Termite the Terminal Emulator **9**
3.1 Installation of Keil MDK . . . 9
3.2 Composing Code Project with Keil MDK . . . 12
3.2.1 Composing a Project . . . 13
3.2.2 Programming MCU with μVision . . . 14
3.3 Installation of Termite the Terminal Emulator . . . 14

4 Monitor Utility Subroutines **16**

Course Instructor

Name	e-mail	Room
Prof. Dr. Gözde B. Akar	bozdagi@eee.metu.edu.tr	D-121/1
Prof. Dr. İlkay Ulusoy	ilkay@metu.edu.tr	EA-410

Laboratory Assistants

Name	e-mail	Room
Doğu Erkan Arkadaş	arkadas@metu.edu.tr	EA-404
Ferhat Gölbol	ferhatg@metu.edu.tr	D-227
Kamil Sert	ksert@metu.edu.tr	EA-405

1 General Information about Laboratory Work

A laboratory work in *EE447 - Introduction to Microprocessors* course is composed of a preliminary work for an experiment and a performance session in which students are to fulfill some experiment related tasks in real-time using ARM MCU. The laboratory works aim to help the students understand most of the concepts given in the lectures.

There will be a total of 6 laboratory works, one of which will be a DEMO session, which has a lower grade then a regular lab session, as shown below. These laboratory works are based on practicing the utilization of the MCU via assembly and C languages and will be in the following subjects:

- Introduction to Laboratory Work with Assembly Programming (DEMO, 2.5 %, Assembly & C)
- Subroutines and Stack (4.5 %, Assembly)
- Input/Output (4.5 %, Assembly)
- Interrupts (4.5 %, C)
- Timer System (4.5 %, C)
- Analog-to-Digital Conversion of a Signal (4.5 %, C)

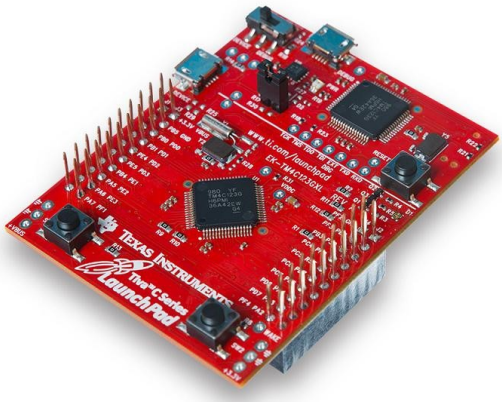
Each laboratory work has its own experimental work. The experiments will be carried out by students with the guidance of the corresponding laboratory work manual. A preliminary work report for each experiment will be prepared. In the laboratory work performance sessions, students will be given tasks that are related to the corresponding experimental works and will be expected to fulfill the tasks with implementations on ARM MCU.

Students will be able to get feedback from the assistants during their laboratory works except for laboratory sessions.

1.1 Experimental Setup

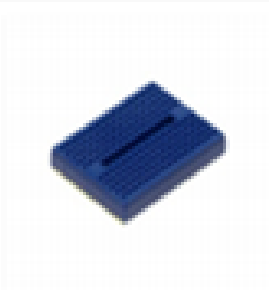
The experimental setup of EE447 Laboratory Work is composed of the following items:

- A personal computer with
  - Windows operating system
  - Keil MDK installed
- The TM4C123G LaunchPad Evaluation Kit (TM4C123G) - A board containing the MCU TM4C123GH6PM
  - TM4C123G is connected to the PC via USB data cable.
  - MCU(TM4C123GH6PM) is programmed via Keil MDK installed in the PC (to be explained in Section 3).
  - For some user interaction purposes, Termite which is a terminal emulator that provides serial communication between PC and MCU is used (to be explained in Section 3).
- Breadboard, M-F and F-F Jumper to construct external circuitry
- Keypad Module, Stepper Motor, Sonar Sensor and Potentiometer to be used in the experiments
- Nokia 5510 LCD Screen to be used in the project



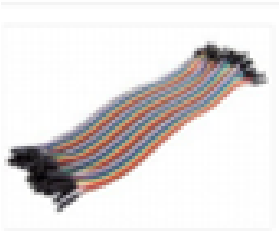
(a) TM4C123G Board

(b) Personal Computer with Windows OS

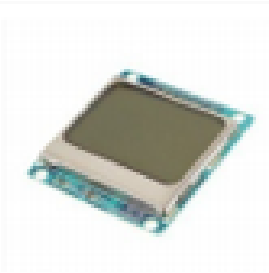


(c) Breadboard

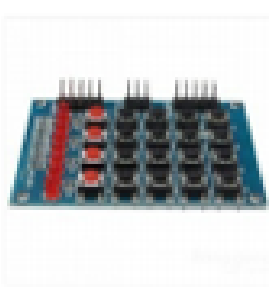
(d) M-F Jumper



(e) F-F Jumper

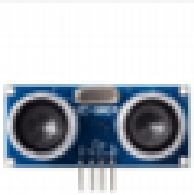


(f) Nokia5110LCD

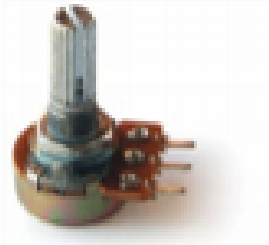


(g) Keypad

(h) Stepper Motor



(i) Sonar Sensor



(j) Potentiometer

Figure 1: Devices and gadgets used in the laboratory work

1.2 Preliminary Work

Preliminary works are experimental works given as exercises for related laboratory works so that students can get prepared for the performance session tasks by finding out and overcoming their weaknesses. Students are expected to have thoroughly learnt the laboratory work subject by the performance session.

Students are to complete a preliminary work for each laboratory work prior to performance session of the corresponding laboratory work. The preliminary works are experimental works and are to be detailed in the corresponding experiment manuals.

A preliminary work report (in PDF) should be prepared for each experimental work. Preliminary work requires implementation of different tasks. The report must include the related codes as texts (not screen shot images) and the illustrations showing the tasks in the *Experimental Work* part of the manual. Preliminary work reports will be collected at the beginning of the corresponding laboratory session week.

Preliminary work assessment is detailed in Sec. 1.5. In summary, codes should be well commented and the working codes for the *Experimental Work* tasks should be demonstrated with illustrations in the report.

Students who complete their preliminary work no more than 50% will not be eligible to take the corresponding laboratory work performance session. Preliminary work grade will not affect the corresponding laboratory work grade.

Preliminary Work Walkthrough

- ☐ Read the experiment manual (download from ODTUClass)
- ☐ Attempt to complete *Preliminary Work* part of the manual with your groupmate
- ☐ Perform *Experimental Work* part of the manual and take pictures, screen-shots or sequential pictures/screen-shots
- ☐ Prepare your preliminary work report in PDF
  - Put your implementations and your answers to the questions in the *Preliminary Work* part
  - Demonstrate *Experimental Work* items with illustrations and short explanations
- ☐ Submit your preliminary work report to ODTUClass by 10.00 on Monday of each lab session

1.3 Laboratory Work Performance Session

There will be a laboratory work performance session for each laboratory work in the corresponding week. Students will be given tasks to be fulfilled using ARM MCU and possibly related gadgets. The tasks will be related to corresponding preliminary work and will be coding problems. Students are expected to implement what is required in assembly/C language and finish the implementations within the laboratory work performance session hours.

No internet browsing will be allowed during the laboratory sessions. Students will be able to use the course materials available at ODTUClass (lecture notes, laboratory work materials, supplementary materials and their preliminary works). To this end, the students are advised to make printed copy of the related materials to use them offline. The codes you have uploaded to ODTUClass will be given to you by the lab assistant with a USB flash drive at the beginning of the experiment.

There will be no quiz in the laboratory session. Exam rules will be applied during laboratory sessions. At the beginning of laboratory sessions, the questions (tasks) will be asked to you by the assistant.

Laboratory Work Performance Session Walkthrough

- ☐ Check your laboratory session group from ODTUClass
- ☐ Wait for the assistant to ask the questions (tasks)
- ☐ Read the instructions carefully

- ☐ Implement what is required during the session on your own without using any communication or browsing the internet
- ☐ Submit your implementations to the assistant at the end of the session

1.4 Feedback

Students will be able to get feedback from assistants about their preliminary work prior to corresponding laboratory session or about their performance on the previous laboratory work performance session tasks.

ODTUClass will be used to schedule individual feedback sessions during the office hours.

Students can - actually are expected to - practice their implementations during the preliminary work period and attend the office hours before attending the performance sessions. In this way, major bugs/misunderstandings that may cost too much time to fix can be eliminated and students are to be on the safer side to complete the tasks in the sessions within the required time slot.

Laboratory Work Feedback Session Walkthrough

- ☐ Check ODTUClass for available time slots of the following sessions
  - Preliminary Work Feedback
  - Laboratory Work Performance Feedback
- ☐ Choose a time slot available for you to schedule your feedback session
- ☐ Use the corresponding meeting url of the session to join your feedback session

1.5 Evaluation

Students' laboratory work grade is affected by the preliminary work grade and the laboratory session performance grade. The former has no explicit effect on the laboratory work grade. It is used to determine whether a student performs the required preliminary study prior to the performance session. The actual laboratory work grade is determined by the performance grade during the laboratory session.

1.5.1 Preliminary Work Reports

The evaluation is based on the functionality of the implementations and comprehensiveness of the answers to the questions. Each experiment manual has an *Experimental Work* part. The functionality of the implementations will be assessed by the demonstration of the tasks in the *Experimental Work* part as well as inspecting the actual implementation codes.

Moreover, implementations without comments will not be considered as valid implementations of the tasks and the corresponding task will not be evaluated. Students are expected to write down explanatory comments to their code. That does not mean one should write an explanation next to each code line. What is required is explaining the functionality of the code blocks and the functionality of the representative code lines where necessary.

Preliminary Work Report Evaluation Rubric

- ☐ Inspect demonstrations of the experimental works
- ☐ Inspect codes of the implementations
  - Check whether implementations satisfy required constraints
  - Check whether the codes are appropriately commented
- ☐ Check the answers to the conceptual questions if any
- ☐ Evaluate the attempted tasks according to aforementioned information

1.6 Laboratory Work Performance Sessions

The performance grade is according to the evaluation of the tasks that are considered in the laboratory session. In each laboratory work performance session, multiple tasks or a task open to partial implementation related to the preliminary work will be given. Students' performance will be evaluated according to the functionality of the implementations. Implementations that do not fulfill any task either completely or partially will get no grade. Namely, a working code should be submitted for a task or for the part of a task open to partial implementation in order to get points from the implementation. Incomplete implementations will not be evaluated.

Laboratory Work Performance Evaluation Rubric

- ☐ Check whether implementations satisfy required constraints
  - If no, penalty will be applied depending on the level of violation if the code is working
- ☐ Compile the code
  - Compile errors: No credit
- ☐ Run the code on MCU
  - Not working as desired: No credit
- ☐ Full credit for the working codes
  - Conditionally penalize (see first item)

2 Laboratory Regulations

This section is devoted to the rules for the regulation of *EE447 Laboratory*. These rules are strict; and by taking *EE447 - Introduction to Microprocessors* course, students will be considered that they have understood and accepted all the rules stated below. The rules for EE447 - Introduction to Microprocessors Laboratory is given below, please read thoroughly.

2.1 Rules

1. A preliminary work report is to be prepared for each laboratory work.
2. The preliminary works are to be collected on the last Monday (10 a.m.) before the corresponding laboratory work performance session.
3. Preliminary work is a crucial part of the laboratory work in both preparing for the laboratory work performance session and understanding the concepts to be covered in the corresponding experiment. Thus, **students will NOT be allowed** to attend the relevant laboratory session without any preliminary work. Partially done preliminary work reports are acceptable if and only if at least the 50% of the total work is completed. Redrawings or rewritings of the given material will not be considered within the aforementioned 50%.
4. Students are expected to prepare their preliminary work in groups of two.
5. Exam rules to be applied during the laboratory work performance sessions are:
   - Laboratory work performance sessions are to be performed individually.
   - Sharing any form of information among students during the session is strictly forbidden.
   - Students are only allowed to use printed copies of the course material available at ODTUClass and their preliminary works during the laboratory sessions. No other resource is allowed.
   - No extra time will be given to the latecomers and no one is allowed to take the laboratory session after 20 minutes past the beginning of the session.
   - Students are to follow the procedures explained in Sec. 1.3.
6. Cheating is an important issue regarding ALL steps of the laboratory work that involve preliminary work and laboratory performance. Disciplinary action is to be taken in case of any cheating and cheating attempt. Please read the subsection 2.2 for detailed information about cheating.
7. The codes in the preliminary works have to be well commented. The codes lacking of comments will not be evaluated. Please read the subsection 1.5 for detailed information.
8. Students with officially documented legal excuses will be allowed to take make-ups. Only academical permissions (given by University), signed confirmation from the instructors for any exam clashes, and medical reports will be regarded as valid documents for the right to take make-ups. Students may take at most 3 make-ups even if they have more than 3 officially documented legal excuses.
9. This document supersedes any other previous documents.

2.2 Cheating

You are considered to be graduate and become engineers and colleagues in 1 or 2 years time. Hence, you are expected to act according to the professionalism that is required as a METU graduate.

Copying a work from any other resource (web-page, your friend's report, older resources you have found, etc.) or sharing information, know-how or code files during sessions are considered as cheating.

Helping your friends, studying together, or any form of cooperation is encouraged -since it both fosters your relationships with others and helps you learn the topic better- as long as YOU do your own work. Creating only one report is not studying together or is not cooperation, and will NOT be accepted.

3 Introducing Keil Microcontroller Development Kit (MDK) and Termite the Terminal Emulator

This section introduces Keil Microcontroller Development Kit (MDK) and Termite the Terminal Emulator, which are software to be used throughout the experiments to program the MCU and to provide user interaction, respectively. The organization of the section is as follows.

First, walkthrough of the installation of Keil MDK is presented for those who consider to purchase their own TM4C123GXL LaunchPad Development Kit and practicing in their own PCs. You may skip reading that section if you are not planning to purchase your own board. However, purchasing a TM4C123GXL board might be beneficial to you in case you consider to use it for your further purposes such as EE493 - EE494 project.

The next subsection explains how to compose code projects in Keil MDK and how to program MCU in TM4C123G board via Keil MDK.

The last subsection introduces Termite terminal emulator that is to be used for some user interaction tasks in some experiments.

3.1 Installation of Keil MDK

The installation is only for Windows operating systems. The MDK contains μVision 5 IDE (Integrated Development Environment) with debugger, flash programmer and the ARM compiler toolchain. The software is free, but limited to compile, assemble, or link programs that are less than 32 Kbytes of code. The installation is to be explained step by step. Before the installation, make sure that your TM4C123GXL board is not connected to your PC since, driver installation is required prior to make a connection.

1. Download Keil MDK-ARM
   - Go to the Keil MDK-ARM product page: `https://www.keil.com/download/product/`
   - Click *MDK-ARM* (Figure 2)

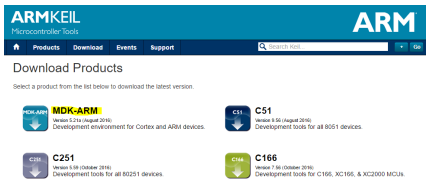


Figure 2: Click *MDK-ARM*

   - Complete the form and proceed (Figure 3)

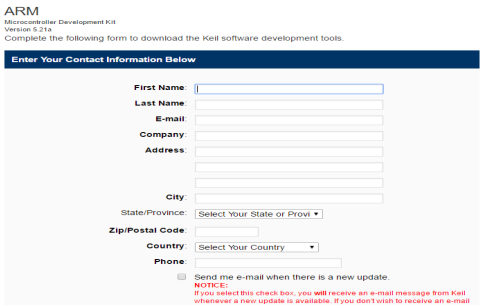


Figure 3: Fill out the form

   - Click the MDK5xx.EXE link, where xx represents the current software version (Figure 4)

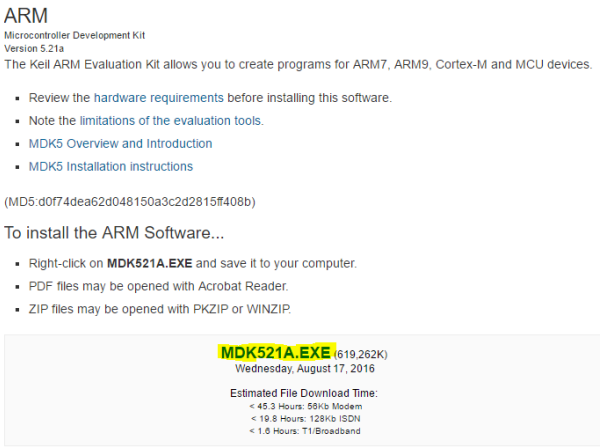


Figure 4: Click the EXE link emphasized by yellow (You might see a different version which is current)

2. Install Keil MDK
   - Run the downloaded setup file
   - Click next on the initial screen
   - Agree to the term of license and proceed
   - Optionally choose where you want to install the software
   - Fill out the required information and start the installation by clicking next
   - If you get a security prompt about installing the device software, click install
   - Installation should be completed
3. Pack Installer
   The Pack Installer is a tool to install the correct support files for the processor / device you are using. Keil μVision can be used to create programs for MANY devices other than the TI TM4C. The Pack Installer is where you choose the packs needed to support your device. **Internet connection is required at this step**.

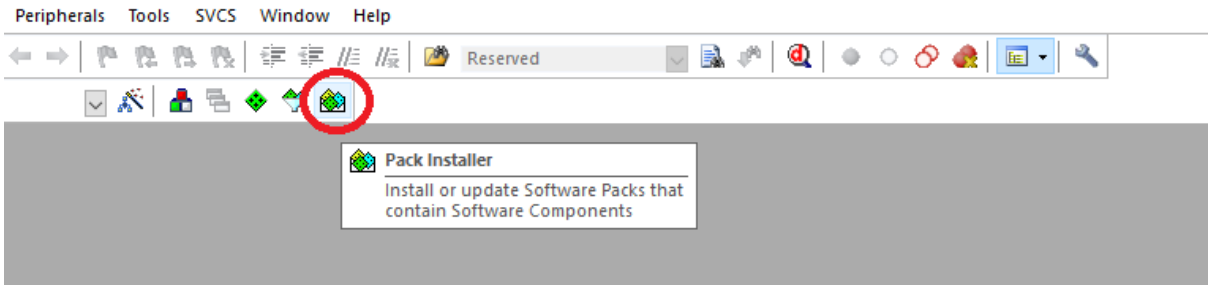


Figure 5: Click the encircled icon to open Pack Installer

You can open Pack Installer from Keil μVision by clicking the corresponding icon (Figure 5. Once it is opened, click *OK* on the first dialog and a large list of packs that support various features on various devices appears. For the first install, it might take a while for the list to fully populate. There exists a button next to each pack. The button might represent different actions to be taken according to the status of the corresponding pack. There are three typical options:

- **Up to Date:** The pack is installed and it is up to date. No action is to be taken upon clicking.

- **Install:** The pack is not installed. Clicking the button will install the pack.
- **Update:** The pack is installed and there is an update available. Clicking the button will fetch and install the required updates for the pack.

Now, you are ready to install the packs required for the TM4C123G board.

- Open Pack Installer and proceed by clicking *OK* on the first dialog
- You should see an interface similar to one provided in Figure 6

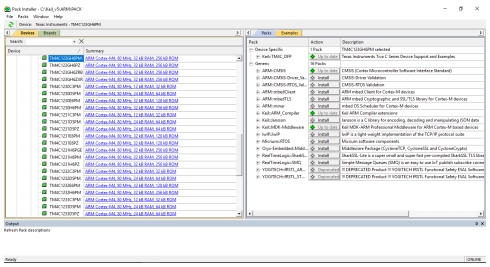


Figure 6: Layout of the Pack Installer interface

- Under *Devices* tab, seek for the TM4C123GH6PM: All Devices > Texas Instruments > Tiva C Series > TM4C123x Series > TM4C123GH6PM (Figure 7)

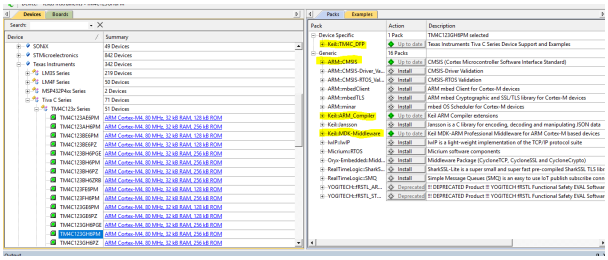


Figure 7: Packs to be installed are emphasized by yellow

- Install the following packs under the *Packs* tab: $Keil :: TM4C\_DFP$, $ARM :: CMSIS$, $Keil :: ARM\_Compiler$, $Keil :: MDK - Middleware$ (Figure 7)
- Under *Devices* tab, seek for the CM4xx (or CM41x) Mixed Signal Control Processors: All Devices > Analog Devices > CM4xx (or CM41x) Mixed Signal Control Processors (Figure 8)

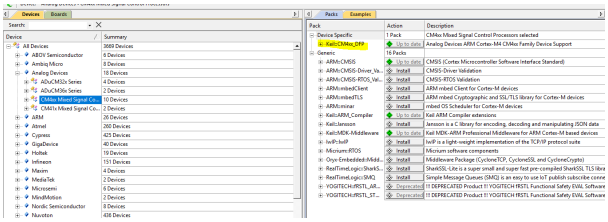


Figure 8: Packs to be installed are emphasized by yellow (Depending on the Keil version selected device under the device tab might be CM41x)

- Install the pack $Keil :: TM4xx\_DFP$ under the *Packs* tab (Figure 8)

4. MDK Stellaris ICDI Add-on: For MDK v5.29 or above, you should install an add-on to bring back the support for Stellaris ICDI. Go to `https://www.keil.com/support/docs/4196.htm` to download and install the add-on.

5. Download and Install In-Circuit Debug Interface (ICDI) Drivers The TM4C123G development board contains an on-board debugger that can be used to debug programs in Keil MDK. Thus, the PC needs a driver to create a communication link between Keil MDK and the development board.
    - Go to TI's Stellaris ICDI drivers page: `http://www.ti.com/tool/stellaris_icdi_drivers`
    - Download the latest drivers (Figure 9)

Part Number	Buy from Texas Instruments or Third Party	Alert Me	Status	Current Version	Version Date
SW-ICDI-DRIVERS: Stellaris® ICDI Drivers - Current	Download	Alert Me	ACTIVE	v1.0	21-JAN-2016
SW-ICDI-DRIVERS-AR01: Stellaris® ICDI Drivers - OLD	Download		ACTIVE		

Figure 9: Download the latest drivers

- Extract the downloaded files to a particular location, which is to be browsed during driver installation
- Plug TM4C123G board into a USB port
- Open Device Manager from Control Panel in Windows
- Expand the *Other Devices* branch to observe *In-Circuit Debug Interface devices*
- Right-click on each ICDI device listed and select *Update Driver Software..*
- On the update wizard screen select *Browse my computer for driver software* option
- Provide the path to where the drivers are extracted to and click *Next* by *Including subfolders*
- In case of a security warning about verification of the publisher of the drivers, proceed with *Choose to Install this driver software anyway*
- Once the driver is installed, the individual ICDI device will be removed from the Other Devices branch
- Repeat the driver install for each remaining ICDI device
- At this point, you also possibly have a device under *Ports (COM & LPT)* branch
- Update this port device according to aforementioned steps, as well

Once the aforementioned steps are performed successfully, the installation is complete.

3.2 Composing Code Project with Keil MDK

This subsection introduces creating code projects and programming MCU with Keil MDK. You will be familiar with those concepts once you complete your first laboratory session. However, for a quick reference, a brief documentation is also to be presented.

First of all, you should create a folder for each project you are going to create since, Keil MDK needs a location to place the auxiliary files. When you are ready, run the μVision application, making sure you see the 2 sub-windows: the Project window on the left, the Build Output window along the bottom (Figure 10). The Build Output and Project windows can be made visible or invisible using the View menu at the top of the program.

All files associated with each of your projects, debug, and build settings are organized and saved using Project files. Each project will contain all of the assembly files created for an individual laboratory or project.

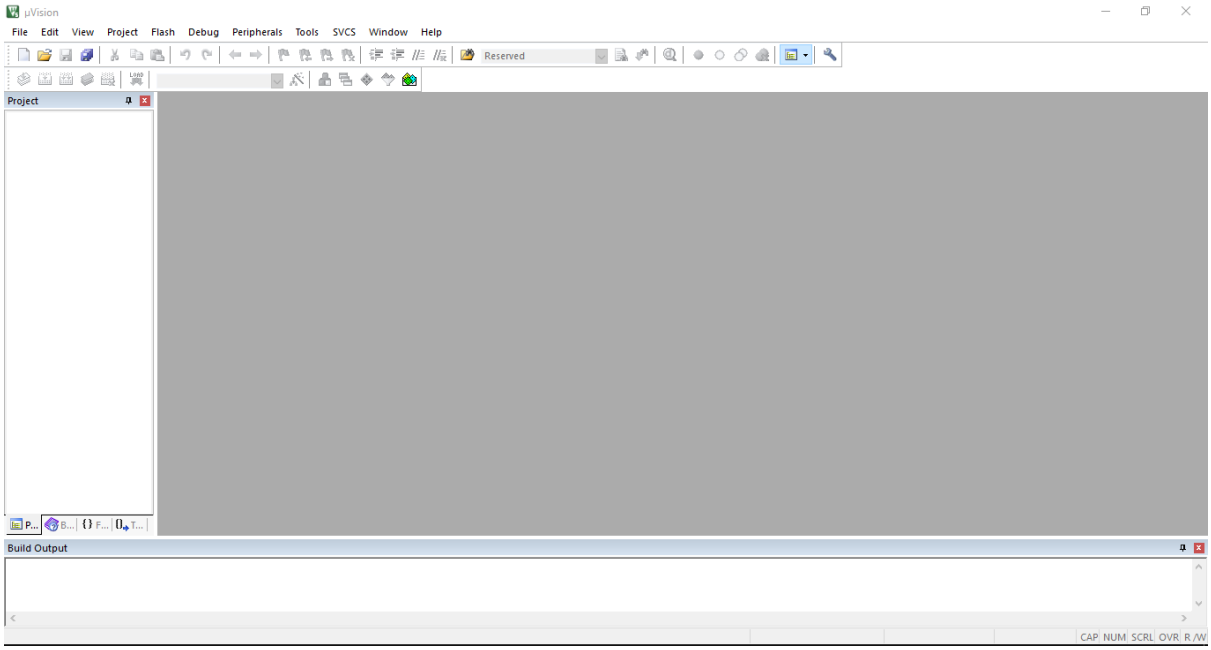


Figure 10: μVision and its sub-windows

3.2.1 Composing a Project

To start a new project:

- Go to the Project menu > New μVision Project
- Create a project folder, then navigate to the Project Folder you just created and select a name for your project
- Next, select the device you are using: In our case, the TI TM4C123GXL development board uses the TM4C123GH6PM microcontroller
- Expand the tree inside of the Target window to find the correct processor: *Texas Instruments* > *Tiva C Series* > *TM4C123x Series* > Scroll down unti you find the *TM4C123GH6PM*
- Select it and click *OK*
- The next window to pop up is the Manage Run-Time Environment window: For an assembly project, nothing needs to be configured here, so simply click *OK*

You will notice that the Project window now has a tree structure. Expanding the Target1 folder shows the Source Group 1 folder. The Source Group 1 folder is where all of your assembly files will be added. There are two ways to add assembly files to your project. Right-click on the Source Group folder and either select *Add New Item to Group* or *Add Existing Files to Group*.

The Add **Existing** option is for when you already have an existing assembly file that you would like to use in your project.

The Add **New Item** option is used when starting an assembly file from scratch. Selecting this option walks you through creating a new assembly file and opens the new file as a new tab in the Editor.

Any new or old program file can be opened in the Editor (like any word processor) using either the File menu or double clicking it in the Project window. Saving the files can be done by selecting *File* > *Save*, or the save button in the menu.

REMARK: If you want a file to be used in your project it MUST be added to the Source Group folder. Otherwise, μVision does NOT consider it during build time.

3.2.2 Programming MCU with μVision

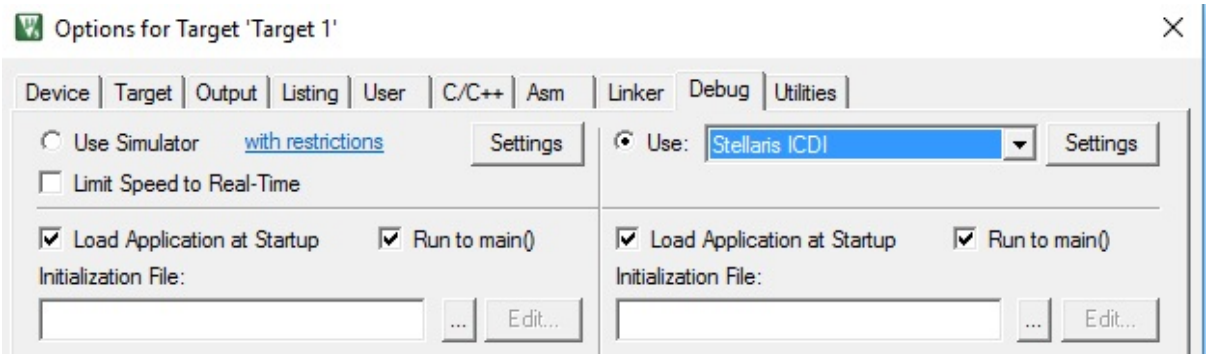
TM4C123GH6PM microcontroller requires an initialization procedure in order to run your programs properly. This initialization is also a program that runs before your main program and performs certain tasks for the functionality of MCU. Generally, microcontroller vendors provide users with the required initialization program. You can download yours -***StartUp.s***- from ODTUCLASS. **You should add this file to your assembly projects**. For now, you might not follow what is going on with those lines of codes, however you will be able to understand each line of the ***StartUp.s*** file by the end of the semester.

Now, assuming that you have added ***StartUp.s*** file to your project and have written your own assembly code, e.g. ***Lab1.s***; the codes in your project need to be built (assembled) in order to be translated into machine code. This can be done using *Project > Build Target*. When the program is built, other files with the same name, but with other file extensions are created. The .lst file is a common listing file, which provides physical address information with operation and operand information. It also gives the error information above the line having the error. Make sure to edit the main program (*.s) and not the .lst file. Upon a successful build, the Build Output window will display something similar to the following:

```
Build target 'Target 1'
assembling Lab1.s...
linking...
Program Size: Code=40 RO-data=0 RW-data=0 ZI-data=1024
".\Objects\Lab1.axf" - 0 Error(s), 0 Warning(s).
```

Having built your program, the next step in the process is to download it to your board. Enter the drop-down menu *Flash* at the top and click *Download*. This can be done also from the download icon on the toolbar or by pressing F8. The Build Output window shows messages about the download progress. If the pop-up window shown below on left appears and/or Build Output window shows message on right, the driver may not be set up properly.

In order to use the correct driver for TM4C123GH6PM microcontroller select *Project → Options for target > Debug* and choose Stellaris ICDI from drop down menu like shown below.



3.3 Installation of Termite the Terminal Emulator

In several experiments, MCU will interact with the user through serial communication. The terminal emulator is similar to a console screen where output from the TM4C123G board can be displayed in and user input to be sent to the board from keyboard can be given to. Hence, throughout the experiments, Termite from Compuphase is utilized as the terminal emulator.

To install Termite:

- Go to the Compuphase Termite page: `http://www.compuphase.com/software_termite.htm`
- Download the latest program only version (Figure 11

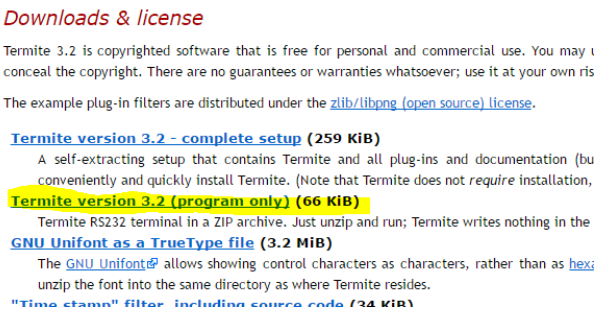


Figure 11: Download the latest program only version

- Extract the downloaded **.zip* file
- Now, you are ready to run the emulator

After the installation, the emulator should be set up to communicate with the development board. To set up the emulator:

- First find the communication port (COM Port) your board is using on your PC (from *Device Manager*)
  - Connect your board to the PC using the supplied USB cable.
  - Go to *Control Panel* and open *Device Manager*
  - Expand Ports (COM & LPT) and find Stellaris Virtual Serial Port (COM x) where x is the number of the communication port
- Open Termite and change the Settings to use the correct COM port
- Click *Settings* and under *Port*, select the number your board is using
- Set Transmitted text to append nothing.
- The rest of the settings should remain default at 9600 baud, 8-bit, 1-stop, no parity

At this point, you are ready to use monitor utility subroutines to have an interaction with the board. Monitor utility subroutines are explained in Section 4.

Once you complete the installation of both Keil MDK and Termite, you can start practicing on your own board and computer.

4 Monitor Utility Subroutines

The monitor utility subroutines are a set of routines that allow communication between the TM4C123G board and a serial window - i.e. Termite 3.3. Those subroutines are to be used in several experiments and the source codes are to be provided to you via ODTUCLASS. You have to add the source codes of the subroutines that you are going to use to your project. Though the explanations of the subroutines are brief in this section, you will understand them better as you proceed in your laboratory sessions. This documentation is for quick reference.

InChar

The InChar subroutine inputs the ASCII character from the keyboard of the PC to register R0. For example, the upper case letter 'A' is loaded into register R0 as 0x41. The subroutine InChar loops until you press a key.

OutChar

The OutChar subroutine can be considered as the reverse of the InChar subroutine. The OutChar subroutine sends the ASCII character value of which is placed in register R0 to the PC through UART0 (serial communication).

OutStr

The other useful subroutine is OutStr. This subroutine outputs a string of ASCII bytes pointed to by the address in index register R0 until the end of transmission character, 0x04, is reached. If you want to display different messages to the terminal screen when certain conditions are met, you can have the string stored in memory with the 0x04 after the last character.