ASSIGNMENT I

MTRN3500 Computing Applications in Mechatronic Systems - 2020

James Stevens & Jay Katupitiya

1 General Instructions

This assignment has been designed to facilitate online learning allowing you to have more liberal access to the experimental set up which we will call "the PLC System". However, you can only access these equipment online through remote login. Please follow the document `MTRN3500 Remote Access Instructions.pdf` in Moodle Week 5 content, for instructions as to how to login. The computer you will be logging in will have one of the PLC Systems physically attached to it. The PLC system will respond to the software you will write. You will also have access to a camera which will allow you to see what happens on the PLC System.

2 Assignment Problem

This assignment requires you to develop an object oriented software package for a Programmable Logic Controller (PLC - and hence PLC System) which in this case is used as a general purpose interface device. PLCs like this are commonly used to control various robotic systems in the industry as they have more functionalities such as process control loops (PID loops). They are equipped with various digital inputs and outputs (for controlling ON/OFF systems, sensing ON/OFF signals and for communicating with encoders) as well as analogue I/O (for dealing with voltage-driven motors or other analogue equipment).

Six of these units have been setup, along with peripheral equipment (encoders, volt meters, LEDs, motors etc.) in the Mechatronics labs at UNSW. You are required to write the necessary software for controlling the PLC. A header file `Galil.h` has been provided for you. It declares all the functions which you are required to implement. You must create a C++ file called `Galil.cpp` (NOTE: The file must have this name EXACTLY) and it must contain all the function definitions you will write. You are allowed to add your own member functions or data, but you must not change the existing member functions and data. Having developed `Galil.h` and `Galil.cpp` you must be completely prepared to demonstrate the functionality of your object class by using it in a `main()` function.

You will be coding the assignment in Visual Studio (one of the best C++ IDE's) on the lab computers. This will require you to login remotely to the Mechatronics Lab computers. It is highly recommended that you write the code on your own personal computer and push it to the lab PCs whenever you make changes. This is because you will be **automatically logged out of the computers after one hour**. We also recommend that you set up the project in your UNSW H-Drive, so that it will be there when you log in to a different lab machine (you will not be able to choose a machine; you will be assigned one when you attempt a login, however, all computers and the experimental set ups are identical).

To start with please download the `Galil.h` file from the Moodle site under Week 5 content.

3 Overview of the Supplied Code

You can find the following in the Moodle site under Week 5 content.

- `Galil.h`: This header file declares all the necessary functions that you must write in your Galil.cpp. Read all the instructions carefully.
- `EmbeddedFunctions.h`: This header file wraps the Galil commands in a class structure. Whenever you send a command to the board, send it through this class. Failure to do so will result in mark deduction.
- **Other Files**: `gclib.lib`, `gclibo.lib`, `gclib.h`, `gclibo.h`, `gclib_errors.h`, `gclib_record.h`, `gclib.dll`, `gclibo.dll`, `GalilControl.lib`: These files contain all the dependencies required to use the Galil library. You must extract the zipped folder on Moodle and place these files in your project / solution folder. Full instructions for setting up the project are found in the next sections.

4 Creating a Visual Studio Solution

To set up the project, please follow the steps below:

1. Open visual Studio and create a new project. File←New←Project
2. Select 'Empty C++ Project'
3. Name your project, whatever you like, and choose a directory
4. Hit Create
5. In a file explorer, find the solution folder for your new project
6. Paste the contents of the zipped dependencies folder
7. In the Visual Studio Solution Explorer, right click on Header Files←Add←Existing. Select the header files that you pasted
8. Similarly, add GalilControl.lib in Resource Files
9. In the Solution Explorer, right click on the project←properties
10. Navigate to Configuration Properties←Advanced and enable Common Language Runtime support (set it to /clr)
11. In the same properties window, navigate to Linker←Input and select the dropdown for Additional Dependencies←Edit

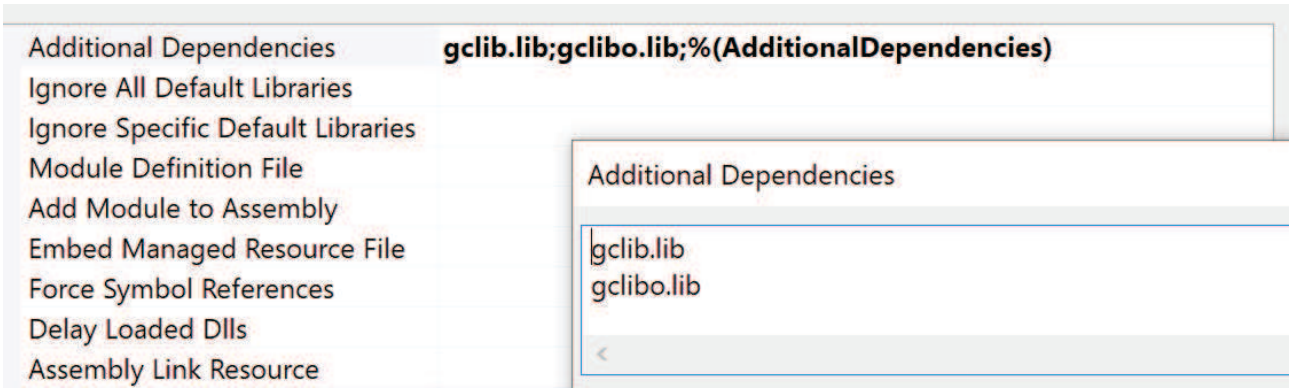

Figure 1: Including libraries

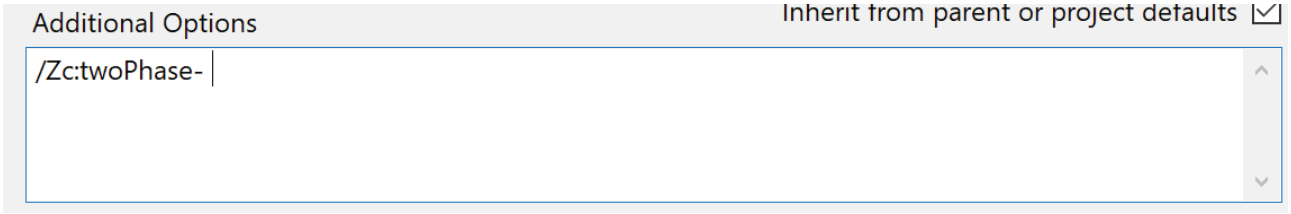

Figure 2: Command line changes

12. Add “gclib.lib” and “gclibo.lib”, each on a new line, as shown in Figure 1.
13. Navigate to C/C++←Command Line and type in “/Zc:twoPhase-”, as shown in Figure 2.
14. You can now create your own main file and start coding. Compiling and linking in VS is called “building”, and you can run either with or without debug functionality.

5 How You Will be Assessed

1. Your assessment will take place in week 5 in a Microsoft Teams meeting with your demonstrator. You need to make an appointment with your demonstrator before the beginning of Week 5.
2. This assignment is a take home assignment. Develop a complete package as described in this assignment and have it ready (with the `main()` function written and all objects instantiated) on the computer you will use for the assessment on the day of the assessment.
3. On the day of the assessment you will be asked to incorporate into your already prepared `main()` function, statements to carry out some tasks. Only a **maximum of 5 minutes** will be available for you to complete your task. This is why it is really important to have all the class instantiation completed and be really prepared to call any of the member functions to complete the specified task.
4. Marks will be awarded according to the following scheme:

(a) Making the program fully operational on the day of the assessment (5 marks) and answering questions posed to you by your demonstrator (3 marks).

(b) Style marking as per the following requirements.

i. **Modularity**:
- Logical breakdown into separate files,
- Well reasoned selection of data and function members of the classes and non-member functions (if any).

ii. **Structure**:
- Order of programming statements,
- Indentation,
- Use of braces and parenthesis,
- Consistent and well reasoned choice of constant/variable/function names.

iii. **Program constructs**:
- Well reasoned choice of data types,
- Proper choice of iterative loops,
- Orderly use of other constructs such as `switch`, `break`, `continue`, etc
- Logical selection of constructors
- Achieving best program logic with least number of lines.

Total for all three items above (6 marks).

(c) Auto-marking checking of the submitted files for the correctness of the solutions to each of the functions (6 marks).

(d) Your mark out of 20, will directly contribute to forming the final mark for this course.

6 What to Submit

Without zipping the files, submit your `galil.h` and `galil.cpp` files to Moodle by 11.59 pm of 16 October, 2020. The submitted code will be first checked for similarity scores and then by the style marker.

7 Penalties

1. If you are found to have plagiarised you will get zero marks for the assignment and you will be reported to School authorities.
2. If you have deleted or changed the existing code of the header files, or do not conform to the submission instructions (resulting in demonstrator intervention), 2 marks will be deducted.
3. Each day late past the online submission deadline will lose 20% of the total marks for the assignment as per School guidelines.
4. If you do not attend the assessment lab session you must apply for special considerations through myUNSW.