ASSIGNMENT 1 - Specification

MTRN3500 Computing Applications in Mechatronic Systems - 2023

Alexander Cunio, James Stevens, & Jay Katupitiya

Change log:

- **11/09/23**: Original release

1 Assignment Aims

This assignment has been designed to introduce you to how you might interact with existing on-the-market hardware. It is conducted through an experimental hardware setup we will call "the PLC System".

By the end of the assignment you will have interacted with the hardware by writing a high-level interface and worked through finding the required methods of communicating with it through the command reference provided by the manufacturer.

You will also get practice reviewing hardware documentation and using this to learn how to interact with devices.

You will also get practice with writing tests over your code to test its functionality.

2 Assignment Problem

This assignment requires you to develop an object oriented software package for a Programmable Logic Controller (PLC - and hence the 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 amplifiers, actuators or other analogue equipment).

Twenty four 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. These are defined in `Galil.h`. You are allowed to add your own **private** member functions or data, but you must not change the existing **public** member functions and data. Having completed `Galil.h` and `Galil.cpp` you must be prepared to demonstrate the functionality of your object class by using calling individual functions where appropriate in a `main` function.

For testing outside of lab sessions, a simulator has been provided. This emulates the functionality of the physical devices as a means of continuing development. Please note that the in-person assessments will run on the physical hardware within the lab so ensure you test on these during your lab sessions.

You will be coding the assignment in Visual Studio 2022 (a popular C++ IDE produced by Microsoft). Most of the development can be conducted on your own computer, so you will need the software installed. Running the assignment requires connection to the PLC manufactured by Galil. This can be done in one of two ways, either the physical in-lab hardware or the provided simulator. Further information on running the assignment can be found in the provided 'Galil Use' document (found on Moodle).

3 What you must implement?

You must first implement ALL the member functions defined within `Galil.h`. The functionality of each of these is defined within the header files itself. These functions each define a way to interact with the hardware, including the digital and analogue inputs and outputs, and querying the PLC. Each function that must be completed is marked with a 'TODO' indicatior within the header file.

Alongside your implementation, you must also write tests over some of the functions. These will be written as unit tests (testing individual implementations) and are important in ensuring code functionality. You should write tests that have good coverage, meaning that they test for all different cases that may exist for the inputs to your functions, including edge cases.

Your tests should be implemented in the file `GalilTester.cpp`. The test structure and the required functions to implement have been defined within the provided header file `GalilTester.h` and you must read this for further information on testing expectations. This header also provides helper functions you MUST use during your test.

4 Overview of the Supplied Files

You can find the following in the Moodle site under Assignment 1. It is expected that you review and read these as they form part of this assignment specification.

- `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.
- `GalilTester.h`: This header file defines all the base functionality for your testing.
- **Other Files**: `gclib.h`, `gclibo.h`, `gclib_errors.h`, `gclib_record.h`, `gclib.lib`, `gclibo.lib`, `gclib.dll`, `gclibo.dll`, `GalilControl.lib`, `Embedded_funcs.lib`, and `GTester.lib`: These files contain all the dependencies required for the project. They

include those provided by Galil for use of their library along with some developed for assignment components. You must extract the zipped folder on Moodle and place these files in your solution folder. Some instructions for setting up the project are found in the next sections. Please note that you have been given multiple versions of each of the library files for different build configurations. Ensure you use the version related to your current build configuration. The build configurations will be explained further in your laboratory classes.

- **Command Reference**: This provides a manual for all the commands that can be sent to the Galil PLC as provided by the manufacturer. For this assignment these commands must be sent through the EmbeddedFunctions interface which you will need to interact with throughout the assignment.
- **Galil Use Instructions**: This supplementary document provides additional details on running the assignment using both the physical hardware in the lab and the provided simulator used for testing at home.

Throughout the supplied code files, the 'TODO' text indicates functions that you must implement.

5 Creating a Visual Studio Solution

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

1. Open Visual Studio 2022 and create a new project. File→New→Project
2. Select 'CLR Empty Project .NET Framework' allowing you to create a C++ Windows Console application.
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 and unzip it.
7. Add "gclib.lib", "gclibo.lib", "GalilControl.lib", and "Embedded_funcs.lib" as libraries.
8. 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.
9. IP address to be used to connect to hardware: 192.168.0.120.
10. Set up your code in a git repository (you will first need a Github account for this) to be able to transfer files between computers and have version control. You MUST use the repository set up for you in GitHub classroom.

A video of the setup process for this assignment has been supplied on Moodle.

6 How You Will be Assessed

Warning: You must attend your designated weekly lab class to progress and get support with this assignment. **You will be assessed on the physical hardware in the lab classes**, so make sure you test your code here beforehand.

This assignment is a take home assignment. You should develop a complete package as described in this assignment. You will be assessed based on this software as well as an in-person assessment as described below.

1. In-person assessment (4 marks)

    - Your assessment will take place in Week 5, during your scheduled tutorial. It will constitute a short 15-minute assessment where you will be provided with a question that defines functionality that is desired from the PLC. Using your interface that you have already written, you must call the functions from the public interface to make the PLC act as specified. Please see the submission section below for opportunities for bonus marks submitting early.
    - You will also be asked **two** theoretical questions related to the PLC and will be given 10 minutes to answer these questions after the completion of the first component. These will be submitted via an online form.
    - 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. **Only one attempt is allowed.**
    - You should have your implementation ready for this along with a `main()` function ready to use. Within this you can have written all objects instantiated ready for use but no other implementation is permitted.
    - The functionality of this task will be assessed in-person by your demonstrator during your lab session directly after the 25 minutes is complete.
    - 2 marks will be awarded for the functionality of your program and 2 marks for the questions asked.

2. Auto-marking (8 marks)

    - Checking of the submitted files for the correctness of the solutions to each of the functions
    - Checking of implementation of Galil functions defined in `Galil.h` (6 marks).
    - Checking of tests as described in `GalilTester.h` against template solution (2 marks).

3. Style marking based on the following requirements (8 marks):

    - **Structure (2 marks)**:
        - Logical breakdown into separate files,
        - Well-reasoned selection of data and function members of the classes and non-member functions.

  - Good application of programming paradigms including DRY (don't repeat yourself) and KISS (keep it simple).
  - Usage of modern C++ principles where appropriate.
- **Layout (2 marks)**:
  - Order of programming statements,
  - Indentation,
  - Use of braces and parenthesis,
  - Consistent and well reasoned choice of constant/variable/function names that enhances readability.
- **Program constructs (2 marks)**:
  - Well reasoned choice of data types,
  - Proper choice of iterative loops,
  - Orderly use of other constructs such as `switch`, `break`, `continue`, etc
  - Achieving best program logic with least amount of coding.
- **Tests (2 mark)**:
  - Well-reasoned testing methodology,
  - Good test coverage of edge cases

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

7 Submission

The assignment is submitted in two components: file submission for style and auto-marking and completion of prescribed task.

The first submission will be for code style and auto-marking. Without zipping the files, submit your `Galil.h`, `Galil.cpp`, `GalilTester.h`, and `GalilTester.cpp` files to Moodle in the 'Assignment 1 - Code Submission' submission box. The submitted code will be first checked for similarity scores prior to the assessment outcomes.

Submission must be by 11.59 pm of Friday of week 5 **(13 October, 2023)**. However, early submission can be conducted to gain bonus marks. Submitting one week early (by 11.59 pm of Friday of week 4) will grant one bonus mark and submitting two weeks early (by 11.59 pm of Friday of week 3) will grant two bonus marks.

The second submission is conducted during your practical assessment in your lab class. This will be assessed in person by your demonstrator for the completion of the provided task. You must also submit the assignment files `Galil.h`, `Galil.cpp`, and `Main.cpp` to Moodle in the 'Assignment 1 - Practical Assessment' submission box directly after completion of your assessment time (you will be given five minutes for this).

Completion and submission of this component must be during your assigned week 5 lab time slot. Early submission for one bonus mark must be during your assigned week 4 lab time and early submission for two bonus marks must be during your assigned week 3 lab time.

***Note**: if you choose to submit early, both components must be submitted early. Submitting **both** parts one week early will grant a total of one bonus mark. Similarly submitting **both** parts two weeks early will grant a total of two bonus marks.*

Note: bonus marks are added to the assignment, which is capped at 20 marks (you are note able to get more than 20 marks for the assignment).

8 Penalties

1. If you are found to have plagiarised you will get zero marks for the assignment and you will be reported to School and University 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 5% of the total marks for the assignment. Please refer to the late penalty details in the course outline. Any submission made later than five calendar days will get zero marks.
4. If you do not attend the assessment lab session you must apply for special considerations through myUNSW.