ASSIGNMENT I

MTRN3500 Computing Applications in Mechatronic Systems - 2014

1 General Instructions

1. You must get this assignment marked by a demonstrator on a date between 15 and 19 September 2014. If you get this assignment marked before this period you will get **2 bonus marks**. If you get the assignment marked after this period, there will be a **penalty of 1 mark** per each delayed day.

2. Make an appointment with the demonstrator allocated to you, before 12 September 2014. Refer to a list on Moodle to find out the demonstrator allocated to you. If you fail to make an appointment by this date, you will **lose ONE mark**.

3. This assignment is a take home assignment. Have a complete package developed as described in this assignment and bring its source code as described (without the `main()` function) along with you on a USB drive on the day of the assessment. Only a **maximum of 20 minutes** will be available for you to complete your task, followed by 10 minutes of assessing by a demonstrator.

4. On the day of the assessment the demonstrator will ask you to write a `main()` function to carry out a specified task. Your work must be carried out on a PC104 based system in the Mechatronics Teaching Laboratory in the Block House of UNSW.

5. Marks will be awarded according to the following scheme:

   (a) Making the program fully operational on the day of the assessment (4 Marks)

   (b) Answers to the 5 questions asked by the demonstrator about your program (5 Marks)

   (c) Modularity: (i). Logical breakdown into separate files, (ii). Well reasoned selection of data and function members of the classes and non-member functions (if any). (2 marks)

   (d) Structure: (i). Order of programming statements, (ii). Indentation, (iii). Use of braces and parenthesis, (iv). Consistent and well reasoned choice of constant/variable/function names (4 marks)

   (e) Program constructs: (i). Well reasoned choice of data types, (ii). Proper choice of iterative loops, (iii). Orderly use of other constructs such as `switch`, `break`, `continue`, etc (iv). Logical selection of constructors (v). Achieving best program logic with least number of lines. (5 marks)

6. Your mark out of 20, will be scaled to a mark out of 15 for the purpose of forming the final mark.

2 Assignment Problem

This assignment involves developing an object oriented software package for the three hardware units listed below. The description of these hardware units is available on Moodle under Hardware Documentation.

1. PCM-3718 multi-function interface card
2. PC104-DAC06 analog output card
3. MSI-P404 encoder input card

A software module consisting of a `.h` file and a `.cpp` file must be created for each of the hardware items mentioned above. In total, you must bring in 6 files (or more if you wish). Of the six files, three will be header files and the other three will be `.cpp` files. The following functionalities must be provided.

1. In all interface card modules, provide a means of specifying the base address of the card.
2. For the PCM-3718 card, in addition to all other member functions and member data you wish to incorporate, write member functions for
   (a) Digital inputs on a specified byte
   (b) Digital inputs on a specified bit
   (c) Digital outputs on a specified byte
   (d) Digital outputs on a specified bit
   (e) Analog Inputs on a specified channel
   (f) Overloading << operator so that a statement of the following form prints analog to digital converted values of all channels on the screen with a tab separation. `PCM3718` is an instantiated object. Mention all default values you use.
   `cout << PCM3718 << endl;`
3. For the PC104-DAC06 card, in addition to all other member functions and member data you wish to incorporate, write member functions for
   (a) Analog output on a specified channel
4. For the MSI-P404 card, in addition to all other member functions, write member functions for
   (a) Encoder reset of a specified channel
   (b) Encoder read of a specified channel
   (c) Overload the operator ! to reset all encoders. For an object `MyEncoder`, a statement of the form,
   `!MyEncoder;`
   must reset all encoder channels.