INSTRUCTIONS TO ENGINEERS

Read this document fully before starting.
Read the first two pages for how the test will be administered, then the problems themselves (page 3 onwards) - then follow the instructions in first 2 pages.

1. Understand the problem - do some basic study, think through all dimensions of **Problem** and compile a **list of written questions and assumptions** based on what questions you think are necessary for an effective design and solution. Understand the requirement and related items. Send your queries to the trainer. This is **SUBMISSION A**. Imagine that you are sending this list in an email directly to a customer. You must submit this list at most 4 hours from start of assignment. If you finish **SUBMISSION A** faster, you can start working on any aspects of problem that you think are not too influenced by the answers to those questions. **SUBMISSION A** will be evaluated based on how completely your questions **spec** out the problem so that the final output coincides with the customer's expectations.

2. The trainer will email you the answers to your questions in written form ~8 hours after the start of assignment (irrespective of when you send the questions within the first 4 hrs). Your next submission will be the Design for your module. You can take around 4 hours for your design. Get your design validated from the trainer. At this point, you will be required to submit your existing code and/or a text file with your thought process so far (need not be run-able code or highly formatted thoughts but will be evaluated for your ability to be productive when the overall problem is still a bit unclear). This is **SUBMISSION B**.

3. On receipt of answers, you can start creating a solution for problem. Submit the basic functional code (as without any added frills) as soon as you finish the testing and you are satisfied that it does what it is supposed to do. The complete module development will be for 12 hours. Split into 2 different releases and submit in 6-6 hours duration (**SUBMISSION C and SUBMISSION D)** Taking longer than this for a more complete or efficient solution is also acceptable. Talk to your trainer for any help in splitting into 2 submissions.

4. Submit your final code at the end of the week **(SUBMISSION E)**. Make incremental backups of code after each running version of interest. Any non-runnable (broken arrow) code that is submitted will not be accepted.

POINTS TO NOTE

Please include any non-obvious instructions for running the code as an email with each submission. Also, any other assumptions or things that the examiner must know to appreciate the value of your submission. Making unjustified assumptions will count against you but not mentioning those assumptions will count even more against you so it is better to mention those assumptions.

You are allowed to use the internet or help documents at any time.

You can also email your questions at any subsequent time after the 1st two hours but the answers may not be provided at guaranteed timings and the questions will **NOT** count towards your evaluation unless they are totally unjustified questions.

Extra features not explicitly specified are welcome but you will be primarily graded based on the features specified. The extra features will definitely count in your favor but **ONLY** if the basic features are of reasonably good quality and met satisfactorily.

At any point, if your question is due to minor misunderstandings in English language, the question **WILL NOT** count against you and will be answered as soon as possible
Note that the test will be paused at 6PM each day and resumed at 9AM the next day with the code being checked in at 6PM every day. Between days, you are **NOT** supposed to discuss the problem, questions or solutions with other engineers. Since the examiners themselves have never solved this particular set of problems before, the evaluation is going to be on a comparative basis and discussing it would indirectly lower your own evaluation versus others. This test is intended for your **INDIVIDUAL** engineering abilities.

PROBLEM STATEMENT

1. Software Model of a multi-channel ADC

Objective:

We want to make a software model for a multi- channel ADC. First, we will build this model, then we will use this model (basically a vi) to build a GUI to plot and stream-to-disk this data. Once the model has been built and tested, and some time while we are building or have built the GUI, the customer will have completed a LabVIEW driver to actually read the data. Then we will replace our model with this driver, so the GUI starts plotting and buffering real data from the device. The GUI may need to work with different ADCs, so it is important that the model and the GUI handles the following changeable aspects of the ADC:

- Returns single dimensional array of U8 data
- The input ADC we are reading from may be putting out data with different number of bits (<=16), number of channels (<=8), arrangement of channels (with some repeating pattern: example - channel1 byte1, channel1 byte2, channel2 byte1, channel2 byte2), sampling rate.
- All we know is that when you call the driver, the driver will return a 1-D array.
- The GUI will be designed to call the driver/model every 250ms and based on the sample rate of the ADC, the model/driver will return a certain number of bytes in every call. The model will be designed to produce a number of bytes that depends on the time since the last call to the model.
- The design of the model/GUI must be done in such a way that when customer tries to adapt to a new ADC, then no change in LabVIEW source code is necessary. (Just a change to some INI file.)

Finally, include proper testing code to prove that your software model for Problem 1 works correctly for the communicated specs and for how it will be used in Problem 2. Include the test code as part of the solution that will be turned in for Problem 1.

2. Multichannel ADC Data Viewer

Design and implement a simple GUI that can do the following using the model built as in Problem 1.

- Allow customer to select a certain ADC.
- Allow user to start/stop data capture.
- Allow user to select which channel to view.
- Allow to select a data storage path.

Show an example of good, scalable and functional LabVIEW code.