MTRX2700 - Mechatronics 2
C Lab - C coding for microcontrollers

This lab is to be completed during the labs in weeks 6 and 7. The outcomes are to be presented during your allocated lab session in week 8.

Objectives

In this lab you will get a practical understanding of the theoretical concepts from this course. We will be using the C language to create programs that can:

- Understand how the microcontroller handles memory (pointers, iterators)
- Demonstrate the use of interrupts for communicating over serial
- Demonstrate the use of timers to perform time critical functions
- Perform an integration of software modules.

Assessment process

This assessment is to be completed in groups of 3 or 4 students. To help facilitate the group work, this problem should be approached by having one or more students responsible for each module (area/topic) - though it is important that each group member understands how all modules work and can describe the details to a tutor.

A good approach to achieve this is for each student to focus on a task, do some research on the topic and present their findings to the group. The group can provide feedback, and learn more about the topic themselves in the process. Before completing the task, the software plan should be completed where you write an outline of how the task/module will be completed (at this point, you should write function templates), including documentation for the inputs and outputs of each function.

Testing is an important part of software engineering. Ideally, the testing plan should come before writing any code. What are the valid range of inputs/outputs that should be handled by each function. What are the constraints and limitations? You should get a different person to be involved in the construction of a testing plan for each software module.

Required documentation

The documentation required for this project should be incorporated into the group git repository. This will allow the tutors to access the code/documentation, **and provide a history of the project including who contributed to each component**.

The git repository should include the following parts which are used for the assessment process:

- **The code** - the folder structure of the repo should be logical and should be able to be described to the tutor.

- **The documentation should be presented as the README for the group repo**. This is the only documentation that will be considered during the assessment process. The readme should be viewed through the github web interface for the demonstration. An example README is shown in the image below. The README document should include the following details:

    - The details about the project (group members, roles, responsibilities)
    - High level information about the code - the function, how the program is broken down into modules.
    - Instructions for the user
    - Details about testing procedures

- **The minutes and agendas from the group meetings.** Stored in a folder in the repo called “minutes”

The day of the assessment

During the lab in week 8, each group will be required to present their solution. There will be some bonus for presenting early in the session, and some penalty for presenting late in the session. The details for this are still to be determined. The tutor will be looking through the git repository which must be completed and ready to go before the demonstration. Each member of the group will be called on to describe various aspects of each module with a focus on the assessment discussion points listed at the end of each exercise.

Before the demonstration, your group must tag the repo for release. Click on “tags” as shown in the image below, and create a new release called “assessment demonstration”.

Assessment day checklist

- ❏ Git repo documentation complete and ready for assessment
- ❏ Create a release version tag called “assessment demonstration”
- ❏ Group members prepared to describe the software modules and testing plan
- ❏ Plan how you will be live demonstrating the functions of each software module to the tutor.

Exercise 1 Serial with interrupts

Objective

This task involves creating an interface to the serial port using interrupts (for both reading and writing). In the ASM lab, serial was implemented using the polling approach. In this lab we will be using interrupts to allow the serial receiving and transmitting to occur in the background, as to not disrupt other tasks.The design of the software module is critical, it should be generic enough such that it could be easily extended to handle any number of serial ports. This task should be demonstrated in either hardware or simulation.

Core functionality

The software should be able to take a pre-filled buffer of 8 bit values and send them to a serial port. It should also be able to read from a serial port and store the incoming 8 bit values in an input buffer. A special character used to determine the end of the input or output should be set as a variable.

Hints

- The simulator can use both serial ports SCI0 and SCI1 without any problems. On the dragon board, SCI0 is generally not available (without setting the switches to “run” instead of “load”) as it is used for debugging.

Discussion points for assessment

- How can you buffer the data (for receive and transmit)
- How long (roughly) does the interrupt handler take to service the interrupt.

Exercise 2 Serial command parser

Objective

The serial port can be a useful interface for the microcontroller to connect to the outside world. The objective of this exercise is to design a method for receiving serial commands and using them to trigger actions in the software. In addition, the serial port can be used to send out information, or error messages when required.

Core functionality

Implement a parser that takes serial data and interprets commands given from an external serial terminal program. The commands should include (as a minimum) a unique identifier for the requested action, and the ability to include some parameters for that action. There should be a mechanism to detect commands that are not formed correctly.

In addition, this exercise requires the microcontroller to be able to send out generic information or error messages when required.

This is primarily a design task, the organisation of the software, and the ability to easily extend to include additional commands or information/error messages will be assessed.

Hints

- You may want to use a predefined set of characters to indicate when a new command is being requested.
- Come up with some example commands to use the peripherals - i.e. turn on LEDs, or similar
- An example of an information or error message could include sending out an error message when the input serial buffer is full.

Discussion points for assessment

- How easy would it be to extend this to different types of messages?
- What ways could the parser fail?

Exercise 3 Music module

Objective

The objective of this exercise is to make the dragon board able to play a tune. This tune should be defined by an 8-bit array (such as characters) representing the pitch and length of each note, and include the possibility of using rests (periods with no sound).

Core functionality

The tune is to be defined by a string of 8-bit values. There should be a flag of some kind to indicate whether the music module is currently playing, or whether it is ready to receive a new tune. This should be connected to exercise 2, and a new tune can be passed through from the serial interface.

The information about how long a newly received song will take to play should be sent to the serial port as an information message.

Hints

- Look at the example code from the week 4 live lecture.

Discussion points for assessment

- What happens if a new tune is requested before the previous one finishes?
- What happens if there are unexpected characters or parsing errors from the tune string.

Marking Scheme

Marks will be allocated based on the following criteria

- Does the core functionality, and any extensions perform as expected
- Quality of the solution
    - Code documentation
    - Modularity of the code
    - Code/algorithm structure
    - Plan for testing
- Presentation and discussion

This project is different from the previous lab. **The solution you present should be contained within a single project**, with some mechanism used to select what components you want to display/demonstrate. During the assessment process, we will require the demonstration of each component, the order in which you do this is up to you.

There will be approximately 30% allocated to each of the 3 exercises, with some marks allocated to administrative tasks including the management of the group, distribution of tasks, completion of meeting minutes and agendas (which must be included in the repository in a folder called “minutes”) and clarity/layout/quality of the project README documentation.

Please note - the tutors will be taking into account the challenges of working with remote group members. We understand that this can add another layer of complexity to the learning process. This is a very important challenge that requires good management and communication, well structured planning and strong leadership.