Embedded Systems Coursework Specification

Coursework 2: Music Synthesiser

Core functional specifications

1. The synthesiser shall play the appropriate musical tone as a sawtooth wave when a key is pressed
2. There shall be no perceptible delay between pressing a key and the tone starting
3. There shall be a volume control with at least 8 increments, which shall be operated by turning a knob
4. The OLED display shall show the name of the note being played and the current volume level
5. Every 100ms the OLED display shall refresh and LED LD3 on the MCU module shall toggle
6. The synthesiser shall be configurable, during compilation or operation, to act as a sender module or receiver module.
7. If the synthesiser is configured as a sender, it shall send a message on the CAN bus whenever a key is pressed or released
8. If the synthesiser is configured as a receiver, it shall play a note or stop playing a note when it receives an appropriate message on the CAN bus
9. CAN bus messages for playing a note shall be an 8-byte sequence `{0x50,x,y,0,0,0,0,0}`, where `x` is the octave number 0–8 and `y` is the note number as the number of semitones above the note C
10. CAN bus messages for ending a note shall be an 8-byte sequence `{0x52,x,y,0,0,0,0,0}`, where `x` is the octave number 0–8 and `y` is the note number as the number of semitones above the note C. The values of `x` and `y` in a note end command shall be ignored by the receiver unless polyphony is implemented as an advanced feature

Non-functional specifications

11. The system shall be implemented using interrupts and threads to achieve concurrent execution of tasks
12. All data and other resources that are accessed by multiple tasks shall be protected against errors caused by simultaneous access
13. The code shall be well-structured and maintainable

Documentation specifications

14. The report shall be presented as documentation in the GitHub repository for your code, consisting of one or more markdown files linked to a table of contents

The report shall contain:

15. An identification of all the tasks that are performed by the system with their method of implementation, thread or interrupt
16. A characterisation of each task with its theoretical minimum initiation interval and measured maximum execution time
17. A critical instant analysis of the rate monotonic scheduler, showing that all deadlines are met under worst-case conditions

18. A quantification of total CPU utilisation
19. An identification of all the shared data structures and the methods used to guarantee safe access and synchronisation
20. An analysis of inter-task blocking dependencies that shows any possibility of deadlock

Advanced features

21. Advanced features shall not modify or replace core functional specifications unless the core functionality can be restored through the user interface or by resetting the system
22. Advanced features should enhance the functionality of the system for the purpose of generating music
23. Advanced features should have appropriate real time constraints
24. Advanced features should demonstrate good software engineering practice for working as a team
25. Advanced features should utilise hardware features of the microcontroller to maximise functionality, performance or quality.
26. Advanced features may be supported by a description of their functionality in the report or a demonstration video to ensure that their extent is fully appreciated during marking.