Timer challenge

Semester 3 Embedded Systems

1 Introduction

1.1 Note for Bachelor / Associate degree students

This challenge is for both bachelor (BA) as associate degree (AD) students. BA students in general will start with research which will eventually result in a complete design. AD students do not have to carry out / document the research steps which would normally lead to a global design. Instead AD students will be given a global design. AD students will still have to fill in the detailed design of the separate components (UML).
In this document the phrases "**BA only**" or "**AD only**" will be used to distinguish between the two.

1.2 Background

Embedded systems are often connected to actuators and sensors or signals with specific timing requirements or properties. Think of a PMW signal or a signal from which the frequency must be measured. Therefore most microcontrollers used on embedded boards contain timers which can be use in a wide variety of applications. In this challenge you will define your own application and explore the usage of timers. Look at the front page of this assignment for inspiration! Consult your teacher to check if the learning goals can be covered with your application.

1.3 Requirements

In the remainder of the challenge we will use the acronym MCP to address the Manual Control Panel.
The letters in front of the requirements refer to the MoSCoW method.

1.3.1 Functional

1. (M) Your application must be operable using an MCP (Manual Control Panel) containing buttons and LEDs or similar.
2. (M) Your application must be operable using a Serial Monitor.
3. (M) Your application must use a sensor or input signal with specific timing properties. Think of a rotary encoder, a pulse signal from a bicycle computer etc.
4. (M) Your application must use an actuator or output signal with specific timing properties. Think of a servo motor, an adjustable RGB LED light etc.
5. (M) Your application should use at least one timer in a meaningful way.
6. (S) The user must be able to adjust important parameters on the fly.

1.3.2 Non-functional

1. **BA Only:** (M) The design must make optimal use of timers. This means for example that counting must not be done using interrupts when a timer can be used.
2. (M) The timed signals must be measured or generated with high accuracy, typically 1 µs. Therefore do not make use of the System tick (SysTick) timer or HAL_GetTick(). This timer is only used to provide a system time tick of typically 1 ms and will not provide the required accuracy.
3. (M) Your application must always be responsive to user actions and button presses.
4. (M) The code must be free of memory leaks and written according to the coding standard.
5. (S) The user interface (MCP or serial communication) must be separated from the application logic.
6. (S) If the application does not respond to user input, it must reboot the system.

2 Challenge

Guideline: ± 6 days of work.

2.1 Challenge details

- **BA only**: Start the challenge by carrying out research. Investigate what is needed to implement the requirements of the challenge. Motivate the choices you make. This includes algorithms, needed internal MCU components, configurations and so on. Carry out the research by using the DOT framework strategies and methods, see http://ictresearchmethods.nl/Main_Page.
- **AD Only**: In the appendix A below a global design is given which you can follow.
- Design all the flowcharts (C) or UML (C++) diagrams needed for the operation of the application.
- Conduct POC's (Proof of concepts) which handle the configuration of the needed internal components of the MCU. POC's are experiments which you carry out to test critical parts of your design. Please document these POC's and use pictures / diagrams to clarify your setup and measurements. Note that a POC can be seen as a research method.
- Write and integrate all necessary code such that all requirements are implemented. Assess, using tests or testcases, that all requirements are implemented properly.
- Write a report, see details below.

2.2 Additional information

- Make use of CMSIS for the implementation.

2.3 Hints

- For separating the user interface (MCP or serial communication) from the application logic you can run the MCP and the serial communication in separate threads using FreeRTOS. Use the proper communication / synchronization techniques available for communication between threads.
- You can use the Analog Discovery 2 wave generator for generating a test signal.
  Important Note:
  The Analog Discovery 2 uses differential signals. This means a generated waveform will typically be symmetrical around the 0 V. On the other hand the STM32 boards have 3.3 V I/O pins, accepting only voltages between 0 and 3.3 V.
  Please make sure that the generated waveform is between 0 and 3.3 V. For the Analog Discovery 2 you can accomplish this by setting both wave generator amplitude and offset to 1.5 V.
  Neglecting this can ruin the board!

3 Hand in to Canvas

Important note:
The grading for this challenge will mainly be based on your documentation. The reader should be able to verify the design process and the design choices based on objective arguments and measurements. Use pictures and diagrams to clarify your setup / measurements and design. Fully functional code without the required documentation has not much value.

Please hand in:

1. All code

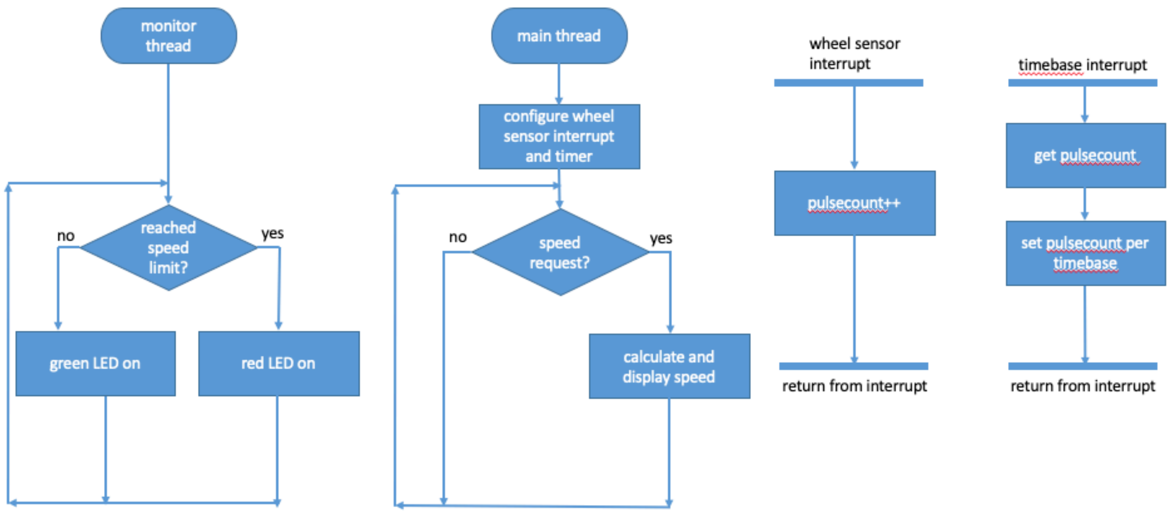
2. Proper documentation consisting of:
    a. Title page with date and name
    b. Short introduction
    c. **BA only:** Your research: Show all relevant research steps (DOT framework) and elaborate on the choices made
    d. Your design: all diagrams with accompanying texts and research
    e. How you tested your implementation including proof (how you validated the results and so on)
    f. Reflection on what you have learned
    g. Bibliography (sources)

Appendix A: Design proposal for AD

Below a global design proposal is given for a bicycle speedometer. It uses a wheel sensor generating one pulse each bicycle wheel rotation. Also one timer is used in timebase mode. This timer records every timebase period the number of pulses generated by the wheel sensor which is a measure for the speed. This speed can be used in the main thread to display the speed. In addition a monitor thread is created which turns on a red LED when the speed limit is reached.

Notes

- The global design is at a high abstraction level and the implementation may vary.
- The wheel sensor can be simulated by a button pressed once for every rotation of the bicycle wheel.
- FreeRTOS can be used to create the separate monitor thread. That way it is guaranteed that the monitoring function will react fast regardless of what the main thread is doing.
- If you want you can deviate from the design proposal, but please document the arguments.


monitor thread
reached speed limit?
no
yes
green LED on
red LED on
main thread
configure wheel sensor interrupt and timer
speed request?
no
yes
calculate and display speed
wheel sensor interrupt
pulsecount++
return from interrupt
timebase interrupt
get pulsecount
set pulsecount per timebase
return from interrupt