Assignments week 6 t/m 9 – TI-RTOS

This assignment will test learning goal 5 (leerdoel 5) as found in the course guide. This goal will account for 35 % of the final grade for ROS01.

Contents

1 Final assignment LD5 ROS01

Modern cars, see Figure 1, provide an enormous number of functionalities. In the late 1970s the Engine Control Unit was introduced. It used pre-computed values in a lookup table to find engine specific values. Nowadays, a car's internals is an explosion of electronics. Just a few examples of electronically controlled systems: air-bag, antilock brakes, automatic transmission, engine control, power steering, and cruise control. Basically, a car is an enormous high-performance real-time system divided into small sub systems. The electronics and software account for 40 % in the production costs of a car. To indicate the complexity of the software: 100 million lines of code, billed at 10 $ per line.

Figure 1: An example of a modern car.

2 Assignment

In this assignment we consider a small part of a car: the airbag system and a black box system that logs information. Your task is to model the tasks and execute them on the LaunchPad. Your working platform is TI-RTOS. The tasks are delivered to your LaunchPad in random order by means of another microcontroller (second LaunchPad). Your instructors will supply the code for the testing platform. The test platform indicates whether or not your tasks meet their deadlines. Figure 2 shows what the final set-up will look like.

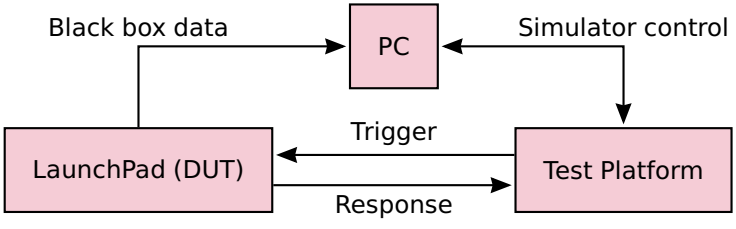

Figure 2: Simulation set-up.

The complete set of tasks you have to model is shown in Table 1. More specific information about the tasks is given in Section 2.3.

Table 1: RTOS tasks.

Task	Sensor	Actuator
1	Shock sensor (Accelerometer)	Airbag
2	Compass	Black box logger
3	GPS	Black box logger

2.1 Testing Platform

We will use another CC3220S LaunchPad as the testing platform. In Table 2 the pins used by the testing platform are listed.

Table 2: Pin mapping.

GPIO #	PIN #	Description	In/out for DUT
6	61	Compass event	IN
7	62	Airbag event	IN
8	63	GPS event	IN
9	64	GPS response	OUT
10	1	Airbag response	OUT
11	2	Compass response	OUT

Each event will be triggered by a positive edge on the corresponding input pin. The duration of this pulse is not specified, so you should use an edge triggered input. Each response must be reported by setting the corresponding output pin high, and then low again. The duration of this pulse must be at least 10 µs.

2.1.1 Loading testing software

If both the testing launchpad and the DUT launchpad are connected to the same computer, it's important to distinguish between the two LaunchPads. This can be done by changing the serial number of the testing LaunchPad.

- Open a console window at `C:\ti\ccs\ccs_base\common\uscif\xds110`.
- Use the xdsdfu tool to see if the LaunchPad is recognized `xdsdfu.exe -e` and put the LaunchPad in DFU mode `xdsdfu.exe -m`.
- Use the xdsdfu tool to change the serial number to, for example, E0081020 using `xdsdfu.exe -n 1020 -r`.

To run the testing software, you'll have to:

- choose Project » New CCS Projec... in the menu and select the project settings as shown in Figure 3;

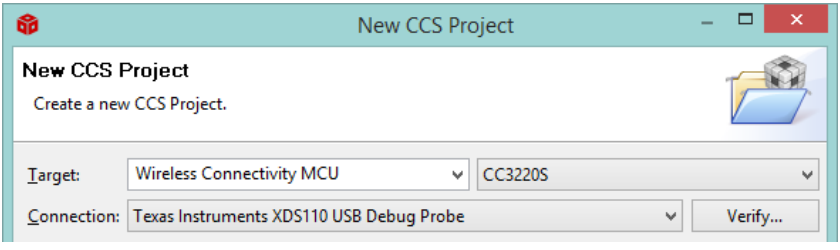

Figure 3: Settings for the new CCS project.

- enter a project name and select *Empty Project*;
- open the project's folder in the Project Explorer, open the TargetConfigs subfolder and open *CC3220S.ccxml*;
- go to the Advanced tab, click on *Texas Instruments XDS110 USB debug Probe* and change the *Debug Probe Selection* setting to *Select by serial number*;
- enter the serial number you configured in earlier steps;
- go to Run » Load » Select Program to Load and select `ROS01_testprogramma-.out`;

- open a terminal to the USER serial port on 9600 baud and default settings. Type `help` in the console to see the available commands.

2.2 Black box

The black box, that will log all the data, is simulated by the computer connected to the LaunchPad. Use the serial port of the DUT LaunchPad as communication link. Any data logged in your simulation thus shows up on the serial console of the computer it is connected to. Note that this is the connection to the PC shown in the upper left part of Figure 2.

You may only communicate the data using 9600 baud 8N1 on UART0.

2.3 Task definitions

Study this section carefully. Please keep in mind that it is important that tasks do not finish too early or too late. This is especially important for safety critical systems such as air-bags. After an event is triggered the appropriate task should respond between the best-case deadline and the worst-case deadline. See Figure 4 for a depiction. As you will see it will be difficult to meet all deadlines when multiple input events arrive in a short period of time. After a task finishes it should respond to the testing platform by triggering a response.

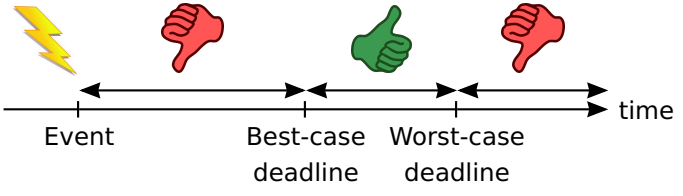

Figure 4: Best-case and worst-case deadline.

2.3.1 Task 1: Air-bag system

Description:	For the airbag system to work, and save lives, it is crucial that it doesn't open too early nor too late. If this task responses too early or too late, missing the deadline becomes indeed mortal. Violation of this deadline is not allowed within this assignment. When the task is finished report this to the testing platform.
Arrival time:	3000 ms ± 1500 ms
Best-case deadline:	30 ms
Worst-case deadline:	32 ms

2.3.2 Task 2: Compass

Description:	This is a simple logger task. The compass sensor will accurately determine the angle of the car relative to the magnetic poles. This data will have to be logged around 5 times a second to the black box. The data always consists of 32 (ASCII) characters. The actual contents of this message is for you to decide. Include a counter in your message, so you can check that all messages are logged. For legal purposes it is very important that no messages are lost. Violation of this requirement is not allowed within this assignment. When the task is finished communicating this must be reported to the testing platform.
Arrival time:	200 ms ± 20 ms
Best-case deadline:	5 ms
Worst-case deadline:	35 ms

2.3.3 Task 3: GPS

Description:	Besides the directional data of the dedicated compass sensor, it is also important to log any positional data. The data always consists of 64 (ASCII) characters. The actual contents of this message is for you to decide. Include a counter in your message, so you can check that all messages are logged. For legal purposes it is very important that no messages are lost. Violation of this requirement is not allowed within this assignment. When the task is finished communicating this must be reported to the testing platform.
Arrival time:	1300 ms ± 20 ms
Best-case deadline:	5 ms
Worst-case deadline:	70 ms

3 Deliverables

You have to hand in:

- source code;
- debugable binary of your project (.out file with all debug information);
- single page report.

Your report should discuss the following:

- short introduction (motivation, problem description and summary);
- overview of your implementation and results;
- conclusion.

Give a clear overview of the tasks in your system by describing their relation with TI-RTOS. Indicate timing constraints and priorities. Determine for some of your functions the expected execution time. Be short and concise. Display your results clear and precise (i.e. use tables and graphics) add screenshots of the testing platform.

Compress your source code, binary and report into one zip file and submit this file to the assignment in Microsoft Teams. The file name has the following convention:
`ROS01_LD5_studentnumber_surname_studentnumber_surname.pdf`
For example:
`ROS01_LD5_0812345_Kowalski_0854321_Klaassen.pdf`
Please carefully comment your code and watch your grammar in all your deliverables. Always mention used sources.

In general, be precise and exact. Good luck!