EDA223/DIT162 Real-Time Systems January 15, 2019

Programming Hints and Guidelines

Below you find some useful information for your work with the laboratory assignment.

Software design:

- Follow the software design guidelines advocated by the course material and the lab assistants. The laboratory assignment is challenging enough even when you do things “by the book”. By deviating from recommended programming guidelines you run a high risk (based on past experiences from the course) of not being able to succeed with the programming assignment.
- Before starting to write your program code, draw access graphs and timing diagrams to make sure that your software design works “on paper”.
- Variables and data structures that constitute the state of an object (i.e., keep their values between method calls) should be placed inside the object, and, if needed by concurrently executing tasks in your program, should be accessed only via synchronized methods. Exception: if you have big data structures, such as lists and tables, with non-changing contents you may instead define them as global variables and refer to them directly.

Debugging by print-outs:

- While you are testing your software it is convenient to print debug messages to the console. For example, if you change a variable by pressing a key on the workstation keyboard, it could be helpful to display the variable’s new value. Or if you receive a CAN message, it may be beneficial to display the contents of the message.
- Use the `atoi` function for converting string representations of integers to their numeric counterparts. Include `<stdlib.h>` to declare the function correctly.
- Use the `snprintf` function for composing strings of arbitrary format for output to the console. Include `<stdio.h>` to declare the function correctly. Make sure that the character array that you use with `snprintf` is large enough to store the entire composed string, including any expanded formatting characters (e.g., `‘%d’`) and the trailing `‘\0’`. Caution: We discourage the use of `sprintf` since it does not prevent unintended writes of data outside the limits of a character array (which can cause many hard-to-solve problems in your software!)
- Printing floating point numbers with `snprintf` is disabled in order to keep code size down. Instead, type cast your numbers to integers and print as such.
- If the console output gets messed up, i.e, not all intended text gets printed at the right time, it is an indicator that your program is overloading the system with too frequent text updates. Do not print debug messages in code sections that are executed hundred or thousand times per second.

- We do not recommend the use of dynamic memory allocation in TinyTimber for applications with hard real-time constraints as it may cause unpredictable delays in the execution of the program.

In case of a program crash:

- The TinyTimber kernel has enabled support for the detection of divide-by-zero and unaligned data transfer hardware faults. If any of these faults occur the program will abort and return to the MD407 debug monitor. An unaligned data transfer means that a 16- or 32-bit word is read from or written to an odd memory address.
- If you get the message "PANIC!!! Empty queue" on the console output, it means that the system is temporarily overloaded by interrupt request. Probable causes: a key on the workstation keyboard has been pressed for too long (activating auto repeat), or some computer board has transmitted CAN messages too frequently.
- If you get the message "PANIC!!! Empty pool" on the console output, it means that the system cannot keep up with scheduling requests. Probable causes: a method is repeatedly calling itself with AFTER using an offset of zero, or multiple copies of the tone generator task exist because old copies were not successfully terminated.
- If a program is terminated (due to a crash, or via the RESET button) and returns to the MD407 debug monitor the program code should be downloaded again in order to guarantee the consistency of initialized static variables.