CS452 – SPRING 2014
ASSIGNMENT 0

BILL COWAN
UNIVERSITY OF WATERLOO

A. INTRODUCTION

In this assignment you will familiarize yourselves with the basic functionality of the TS-7200 board, its ARM CPU and the train controller while learning techniques for handling asynchronous events by polling.

We recommend that you first write, compile, and execute several simple programs for the TS-7200, and only then start working on the assignment.

B. DESCRIPTION

Write a program that runs on the TS-7200 system. Your program interacts with the user via the monitor while controlling the following independent real-time activities:

1. a digital clock showing minutes, seconds, and tenths of seconds, which measures time intervals accurately, in the sense that it does not slow down or lose ticks,
2. a command line interface to the track that sets train speeds and switches turn-outs, and
3. a real-time display on the terminal showing the state of the track.

The monitor should use cursor addressing to display on its screen at least the following:

1. the current time,
2. a table of switch positions,
3. a list of the most recently triggered sensors, and
4. a prompt at which the user can type commands.

The command line interface should support, at a minimum, the following commands.

1. `tr  train_number  train_speed` – Set any train in motion at the desired speed (0 for stop).
2. `rv  train_number` – The train should reverse direction. To reverse direction set the train speed to zero, wait until the train stops, send the reverse direction command, then send a speed command to re-accelerate the train.
3. `sw switch_number switch_direction` – Throw the given switch to straight (S) or curved (C). *Do not under any circumstances activate a switch over and over again.* Doing so will burn out the solenoid, thereby

pissing off all of the following: me, the TAs, your classmates and Fraser.

4. q – halt the system and return to RedBoot.

Use the built-in 32-bit timer to implement the clock. Do not use interrupts.

c. Program Structure

Your program should be written as a polling loop, using output port COM1 to talk to the train set, and output port COM2 to talk to the terminal. (This is the default cabling configuration and you should always leave it so.)

The clock should not lose time.

d. Possibly Helpful Comments

On the main web page is a link called Printed Notes, which gives you a page with a whole lot of technical documentation including notes I have written for the course. You will definitely want to look at the documentation to which this page points.

You can use the headlight and the horn on the trains as easy methods for telling if your commands to the trains are successful.

The terminals have three serial (com) ports. You can open two windows, and run a gtk-term in each, with one set to the characteristics of the train interface. In that way you can separate a hard problem, 'Is the problem in the UART's provision of correct output or are my train commands incorrect?' into two simple problems.

If you are using functions from bwio for anything except debugging you are probably doing the assignment incorrectly.

There is a deliberately introduced bug in one of the bwio functions.

e. Hand in

Hand in the following, nicely formatted and printed.

1. A description of how to operate your program, including the full pathname of your executable filewhich we will download for testing.
2. A description of the structure of your program. We will judge your program primarily on the basis of this description. Describe which algorithms and data structures you used and why you chose them. This description should include a list of unimplemented aspects of the assignments, if any. Also, if you know of bugs in your implementation describe them explicitly.
3. The location of all source code submitted for the assignment and a set of MD5 hashes of each file. The code must remain unmodified after submission until the assignments are returned.
4. A listing of all files submitted.
5. Answers to the following questions:

i. How do you know that your clock does not miss updates or lose time?
ii. How long does the train hardware take to reply to a sensor query?

(Note: To answer these questions, you need to do some timings of the performance of your polling loop.)