Safety-Critical Systems 4:
Engineering of Embedded Software Systems

AG Betriebssysteme, FB 3
Jan Bredereke

WiSe 2002/03

issued: Oct. 29, 2002
due: Nov. 14, 2002

Assignment 3

Bicycle Odometer – Software Requirements Specification

We continue with our simple bicycle odometer from the last assignments, again. We decide that the odometer shall contain a embedded controller with some integrated input/output hardware and an intergrated timer. For simplicity, we assume that a basic real-time operating system exists for the controller. It provides us with an interface to the hardware.

The operating system calls the user routine `rtmain()` exactly every 1 ms. This is controlled by a hardware timer. We assume that processing of the user routine is fast enough to fit into these time slots easily. After power-up, and before `rtmain()` is called for the first time, the user routine `rtinit()` is called once. When power fails, the controller just stops working and the display goes blank. (The behaviour on power failure need not be specified formally.)

At some points of time while the user routine `rtmain()` is not running, the operating system performs output and input. The variable `inByte` has 8 bits with the following meaning: the least significant bit (at position 0) reflects the rotation sensor. It is 1 if the resistance was $< 10\ \Omega$ at measurement time, and 0 otherwise. The next bit (at position 1) reflects the multi-function button position. It is 1 if the button was down at measurement time, and 0 otherwise. The other six bits are always 0.

The variable `modeOut` has 8 bits with the following meaning: the bits 0, 1, and 2 reflect the indicators for "km/h", "km total", and "km trip", respectively. If a bit is 1, the corresponding indicator is visible, if the bit is 0, the indicator is not visible. When at least one of the bits 1 and 2 is set, the decimal point of the number display is visible, otherwise it is invisible. The display is updated at some point of time after the `rtmain()` routine has terminated, and before it is called again. The remaining 5 bits must always be 0.

The variable `numberOut[]` is an array of five 8-bit variables. The array ranges from index 0 to index 4. Each of the variables determines the visible value of one digit on the odometer's display. `numberOut[0]` is the most significant digit and `numberOut[4]` is the least significant digit. Indices between these behave accordingly. A value of 0 displays the digit "0", a value of 9 displays the digit "9". Values between 0 and 9 behave accordingly. A value of 15 leaves the digit blank. The behaviour for other values is not defined. Again, the display is updated at some point of time after the `rtmain()` routine has terminated, and before it is called again.

- Specify the input and output quantities of the software rigorously, according to the above informal description.
- Write rigorous requirements specifications for the relations IN and OUT, according to the above informal description.
- Write a rigorous software requirements specification SOF that is *acceptable* with respect to the system requirements specification. (No formal proof required.)

As usual, add information missing in the informal description, and document this where the decision is not obvious.

In order to make the results of the solutions comparable, you should use as a base the system requirements specification on which we agree on Thursday Oct. 31 in the seminar. (It will also be published on the lecture's Web site.)

Submit your solution both on paper and by email to `brederek@tzi.de` (as a PostScript or Pdf file).