SSFT 2022 **Interrupt-Driven Program**

Exercises Day 1: UCLID5

(Modeling / Model Checking)

In this assignment you will complete/correct a model of the interrupt-driven program covered in the lecture on Modeling for Verification (Feb. 28). Recall that the program contained a `main` function and an interrupt service routine `ISR`. In class, we discussed how neither purely synchronous composition nor purely asynchronous composition works for creating a model that faithfully represents timing semantics.

A unit of concurrency in this model is the `module`. Modules can be synchronously composed (default in UCLID5) or asynchronously composed (you will need to encode this manually). As discussed in class, this program reqires a mix of asynchronous and synchronous composition. In this question, you are tasked with coming up with the right kind of composition and encode it in your UCLID5 model.

Download the skeleton code from the Homeworks/HW3 folder on bCourses: the file you need is `IntSW.ucl`. This file contains a partial and slightly buggy model of the system that you will need to modify only in a few prescribed places indicated by comments.

Complete the following steps:

(i) Read the file `IntSW.ucl`. Read the properties listed in the Sys module and describe each of them in English. Note that the composition of main and ISR within the module Sys is incorectly done (you will need to fix it in subsequent questions).

(ii) Run the file as

```
uclid IntSW.ucl -m Sys
```

Interpret the results from the verifier. If you obtain counterexamples to certain properties, explain why those counterexamples arise.

(iii) Fix the model to correctly compose main and ISR in module Sys and eliminate the above counterexamples. You will only need to change the procedure `update_mode`.

(iv) Now consider the top-level main module. The composition of Sys and Env must be asynchronous composition with interleaving semantics. Ascertain whether this is the

case. If not, correct the model to correctly compose Sys and Env. You will only need to change the `init` and `next` blocks by changing the way the variable `turn` is updated.

The property `consec_main_pc_values` should hold on the skeleton code we have provided but may not hold on your corrected asynchronous composition. In your write-up, explain what this property is checking. If it does not hold on the asynchronous composition, explain why.

Note: you need to use the command `uclid IntSW.ucl` (without the "`-m Sys`" for this part of the question since you are analyzing the main module.