Software Design

22 March 2015

2IO70

The purpose of this document is to present a Java program that realises the functions specified in the Software Specification document. This program is an intermediate step towards writing the PP2 code that controls the sorting machine.

Group 16

Rolf Verschuuren

Wigger Boelens

Stefan van den Berg

Dat Phung

Maarten Keet

Tudor Petrescu

Version 1

Contents

Coding Standards

The java pseudo code follows the Google Java Style.
Source to Google Java Style: https://google-styleguide.googlecode.com/svn/trunk/javaguide.html.

PHP code used in this project follows the Zend Framework Coding Standard for PHP.
Source: http://framework.zend.com/manual/1.12/en/coding-standard.html.

Translating to pseudo java:

The java program starts by declaring the output variables. The names of the output variables will keep their original name, without spaces, in a camelCase form. The variable type will be determined from the Output table.

The inputs follow the same pattern.

Every states is represented as a function, keeping their name in the camelCase fashion, they will be all void functions due to the fact that they do not return anything.

Every state function will run preconditions if any, then check for specific input values using ***if*** statements, if an ***if*** statement is satisfied, there will be changes to the output values to match the next states output values, also the display is set to output the next states number, and then the next state function is called according to the state transition diagram, if no ***if*** statement is satisfied the current function is recalled.

The program is always looping, consequence of no deadlocks in the state machine as proven by the uppaal model test.

Example: Initial -> Calibrate_Sensor

So in this example the function initial is currently running, there are no preconditions to be checked, if the inputs have the desired value, in our case we check if the push button is pressed by the sorter, if so we will have the sorter moved down by activating the sorter motor via having the Hbridge0 variable set to 1. After this we set the display to showcase the number 2 then call calibrateSensor function and if the ***if*** statement wasn't satisfied we recall initial entering a loop.

Translating from java to php:

The java code was written such that the conversion process to php is as easy as possible.

All variable in java will have the "$" sign added at the beginning of their name to comply with the php standards. The "$" sign has no influence in the java program variable naming, while in php it is mandatory.

Validation of java to transition table

Every state is represented by a function. The if statements in that function are the transitions which can occur from that state. The timer interrupt and the abort transitions are not represented as if statements, because interrupts go to a separate state(function). In those if statements the values that have to change are changed. The display will also be updated to the correct number of the state. The function timerManage is called in each state. Because with that function we make sure that the all outputs have the correct voltage.

We checked that all states are represented in the java code by a function. We also checked if they have all the transitions as if statements and that the correct values are changed.

Validation of timerManage

Loop invariant:

All elements before the current element of the array have been set on if they had to be on.

Initialize:

We start with the first element. Thus there are no elements before it and the loop invariant holds.

Step case:

If we're at element k, then according to the loop invariant all elements before k have been set on if they had to be on. Then if k has to be on (counter<value of k) it will be set on else it will stay off. So now the loop invariant holds for the element k+1

Termination:

The loop will terminate when k is greater or equal to 7. Then the engines will be shut down.

Appendix 1: Java code

```
class SoftwareDesign {
  //**@DATA**
  //outputs
int[] $outputs=new int[8];

  //**@CODE**
  //inputs
  Boolean $startStop,$abort,$push,$position,$colour;
  int $timer;
  //variables
  int $state=0;
  int $sleep=0;
  int $location;
  int $counter;
  int $engines;

  //constants
  final int $timeMotorDown=-1;
  final int $belt=-1;
  final int $sort=-1;
  final int $timerSort=-1;
  final int $lensLampPosition=0,
            $lensLampSorter=1,
            $hbridge1=2,
            $hbridge0=3,
            $conveyorBelt=4,
            $feederEngine=5,
            $display=6,
            $ledStateIndicator=7;



  void initial() {
    timerManage($outputs);
    $push=buttonPressed(5);
    if($push==true){
        $outputs[$hbridge0]=0;
        $outputs[$hbridge1]=9;
        $state = 1;
        display($state,"leds2","");
        calibrateSorter();

    }
    initial();
  }
  void calibrateSorter(){
      timerManage($outputs);
      if($sleep==$timeMotorDown*1000){
         $outputs[$hbridge1]=9;
         $state=2;
         display($state,"leds","");
         resting();
         $sleep=0;
      }
      $sleep++;
      calibrateSorter();
  }

  void resting(){
      timerManage($outputs);


  void resting() {$startStop=buttonPressed(0);
      if($startStop==true){
          $outputs[$lensLampPosition]=12;
          $outputs[$lensLampSorter]=12;
          $outputs[$conveyorBelt]=9;
          $outputs[$feederEngine]=5;
          setTimer(2+$belt);


          $state=3;
          display($state,"leds2","");
          running();
      }
      resting();
  }

  void running(){
      timerManage($outputs);
      $position=buttonPressed(7);
      $startStop=buttonPressed(0);
      if($startStop=true){
          $outputs[$feederEngine]=0;
          setTimer($belt);
          runningTimer();
      }
      if($position=true){
          setTimer(2+$belt);

          $state=4;
          display($state,"leds2","");
          runningWait();
      }
      running();
  }

  void runningWait(){
      timerManage($outputs);
      $position=buttonPressed(7);
      $colour=buttonPressed(6);
      $startStop=buttonPressed(0);
      if($startStop=true){
          $outputs[$feederEngine]=0;
          setTimer($belt);
          runningTimer();
      }
      if($position){
           setTimer(2+$belt);

          $state=5;
          display($state,"leds2","");
          runningTimerReset();
      }
      if($colour){
          $outputs[$hbridge0]=9;

          setTimer($sort);

          $state=6;
          display($state,"leds2","");
          motorUp();
      }
      runningWait();
  }

  void runningTimerReset(){
```

```
131         timerManage($outputs);
132         runningWait();
133     }
134
135     void motorUp(){
136         timerManage($outputs);
137         $push=buttonPressed(7);
138         $startStop=buttonPressed(0);
139         if($startStop=true){
140             $outputs[$feederEngine]=0;
141             setTimer($belt);
142             motorUpTimer();
143         }
144         if($push=true){
145             $outputs[$hbridge0]=0;
146             $state=7;
147             display($state,"leds2","");
148             whiteWait();
149         }
150     }
151
152     void whiteWait(){
153         timerManage($outputs);
154         if($sleep==$timerSort*1000){
155         $outputs[$hbridge1]=9;
156         $state=8;
157         display($state,"leds2","");
158         motorDown();
159         $sleep=0;
160         }
161         $startStop=buttonPressed(0);
162         if($startStop=true){
163             $outputs[$feederEngine]=0;
164             setTimer($belt);
165             whiteWaitTimer();
166         }
167         $sleep++;
168         whiteWait();
169     }
170
171     void motorDown(){
172         timerManage($outputs);
173         if($sleep==$timeMotorDown*1000){
174             $outputs[$hbridge1]=0;
175             $state=9;
176             $sleep=0;
177             display($state,"leds2","");
178             runningWait();
179         }
180         $startStop=buttonPressed(0);
181         if($startStop=true){
182             $outputs[$feederEngine]=0;
183             setTimer($belt);
184             motorDownTimer();
185         }
186         $sleep++;
187         motorDown();
188
189     }
190
191     void runningTimer(){
192         timerManage($outputs);
193         runningStop();
194     }
195
196     void motorUpTimer(){
197         timerManage($outputs);
198         motorUpStop();
199     }
200
201     void whiteWaitTimer(){
202         timerManage($outputs);
203         whiteWaitStop();
204     }
205
206     void motorDownTimer(){
207         timerManage($outputs);
208         motorDownStop();
209     }
210
211
212     void runningStop(){
213         timerManage($outputs);
214         $colour=buttonPressed(6);
215         if($colour==true){
216             $outputs[$hbridge0]=9;
217             $state=10;
218             display($state,"leds2","");
219             motorUpStop();
220         }
221         runningStop();
222     }
223
224     void motorUpStop(){
225         timerManage($outputs);
226         $push=buttonPressed(5);
227         if($push==true){
228             $outputs[$hbridge0]=0;
229             $state=11;
230             display($state,"leds2","");
231         }
232         motorUpStop();
233     }
234
235     void whiteWaitStop(){
236         timerManage($outputs);
237         if($sleep==$timerSort*1000){
238         $outputs[$hbridge1]=9;
239         $state=12;
240         display($state,"leds2","");
241         motorDown();
242         $sleep=0;
243         }
244
245         $sleep++;
246         whiteWait();
247     }
248
249     void motorDownStop(){
250         timerManage($outputs);
251         if($sleep==$timeMotorDown*1000){
252             $outputs[$hbridge1]=0;
253             $state=9;
254             $sleep=0;
255             display($state,"leds2","");
256             runningWait();
257         }
258         $sleep++;
259         motorDown();
260     }
261
262     void timerInterrupt(){
263         timerManage($outputs);
264         $outputs[$hbridge0]=1;
265         $outputs[$hbridge1]=0;
266         $outputs[$lensLampPosition]=0;
267         $outputs[$lensLampSorter]=0;
268         $outputs[$ledStateIndicator]=0;
```

```
269         $outputs[$display]=0;
270         $outputs[$conveyorBelt]=0;
271         $outputs[$feederEngine]=0;
272         initial();
273 
274     }
275 
276     void abort(){
277         timerManage($outputs);
278         $outputs[$hbridge0]=0;
279         $outputs[$hbridge1]=0;
280         $outputs[$lensLampPosition]=0;
281         $outputs[$lensLampSorter]=0;
282         $outputs[$ledStateIndicator]=0;
283         $outputs[$display]=0;
284         $outputs[$conveyorBelt]=0;
285         $outputs[$feederEngine]=0;
286         aborted();
287 
288     }
289 
290     void aborted(){
291         timerManage($outputs);
292         $startStop=buttonPressed(0);
293         if($startStop=true){
294             $outputs[$hbridge0]=1;
295             $state=0;
296             display($state,"leds2","");
297             initial();
298         }
299         aborted();
300 
301     }
302 
303     void timerManage(int[] $outputs){
304      $location = $location % 7;
305      $counter = $counter % 12;
306 
307      if($counter < $outputs[$location]){
308       $engines = $engines + pow(2, $location);
309      }
310 
311      if($location >= 7){
312       display($engines, "leds","");
313       $engines = 0;
314       return;
315      }
316 
317      $location++;
318      $counter++;
319      timerManage($outputs);
320      return;
321     }
322 
323 
324 
325      public static void main( String args[] ) {
326        new SoftwareDesign().initial();
327      }
328   }
```