**Appendix: Java Program**

1 */\*\**

2  *\* Sort of a simulation of the PP2 program*

3  *\* controlling the Fischer*

4  *\* Technik in order to sort black and white discs.*

5  *\**

6  *\* @author Maarten Keet*

7  *\* @author Stefan van den Berg*

8  *\* @author Rolf Verschuuren*

9  *\* @author Wigger Boelens*

10  *\* @team Group 16*

11  *\* @since 13/3/2015*

12  *\*/*

13

14

15 **class** **SoftwareDesign** {

16 *//\*\*@CODE\*\**

17 *//inputs*

18 int $push, $startStop, $abort, $position,

19 $colour;

20

21 *//variables*

22 int $state = 0;

23 int $sleep = 0;

24 int $temp = 0;

25 int $location;

26 int $counter = 0;

27 int $engines;

28

29

30 *//constants*

31 **final** int TIMEMOTORDOWN = 30;

32 **final** int BELTROUND = 2000;

33 **final** int BELT = 1200;

34 **final** int SORT = 850;

35 **final** int LENSLAMPPOSITION = 5,

36 LENSLAMPSORTER = 6,

37 HBRIDGE0 = 0,

38 HBRIDGE1 = 1,

39 CONVEYORBELT = 3,

40 FEEDERENGINE = 7,

41 DISPLAY = 8,

42 LEDSTATEINDICATOR = 9;

43

44 **public** **static** void main(String args[]) {

45 SoftwareDesign SoftwareDesign = **new**

46 SoftwareDesign();

47

48

49 *//values for the data segment*

50 SoftwareDesign.initVar("outputs", 12);

51 SoftwareDesign.initVar("stackpointer", 1);

52 SoftwareDesign.initVar("offset", 1);

53

54 *//store the offset of the programm,this*

55 *// is used in the interrupt*

56 SoftwareDesign.storeData(startofthecode,

57 "offset", 0);

58

59 *//store the vlue of the stackpointer,so*

60 *// we can clear the stack*

61 *// easily*

62 SoftwareDesign.storeData(SP,

63 "stackpointer",

64 0);

65

66 $counter = 0;

67

68

69 *//reset outputs*

70 SoftwareDesign.storeData(0, "outputs",

71 SoftwareDesign

72 .HBRIDGE1);

73 SoftwareDesign.storeData(0, "outputs",

74 SoftwareDesign

75 .LENSLAMPPOSITION);

76 SoftwareDesign.storeData(0, "outputs",

77 SoftwareDesign

78 .LENSLAMPSORTER);

79 SoftwareDesign.storeData(0, "outputs",

80 SoftwareDesign

81 .LEDSTATEINDICATOR);

82 SoftwareDesign.storeData(0, "outputs",

83 SoftwareDesign

84 .DISPLAY);

85 SoftwareDesign.storeData(0, "outputs",

86 SoftwareDesign

87 .CONVEYORBELT);

88 SoftwareDesign.storeData(0, "outputs",

89 SoftwareDesign

90 .FEEDERENGINE);

91

92 *//start moving the sorter up*

93 SoftwareDesign.storeData(9, "outputs",

94 SoftwareDesign

95 .HBRIDGE0);

96

97 *//go to the first state and set the*

98 *// value for the display*

99 SoftwareDesign.$state = 0;

100 SoftwareDesign.initial();

101 }

102

103 *//state0*

104 void initial() {

105 setStackPointer(

106 getData("stackpointer", 0));

107 timerManage();

108 *//check if the sorter push button is*

109 *// pressed*

110 $push = getButtonPressed(5);

111 **if** ($push == 1) {

112 *//move the sorter down*

113 storeData(0, "outputs", HBRIDGE0);

114 storeData(9, "outputs", HBRIDGE1);

115 *//update the state*

116 $state = 1;

117 *//reset sleep for the next function*

118 $sleep = 0;

119 calibrateSorter();

120

121 }

122 *//loop*

123 initial();

124 }

125

126 *//state 1*

127 void calibrateSorter() {

128 timerManage();

129 *//the sorter is now moving down,*

130 *//and we're waitng for it to reach the*

131 *// bottom*

132 **if** ($sleep == TIMEMOTORDOWN \* 1000) {

133 *//stop the sorter*

134 storeData(0, "outputs", HBRIDGE1);

135 *//update the state*

136 $state = 2;

137 *//reset sleep*

138 $sleep = 0;

139 resting();

140 }

141 *//loop*

142 $sleep++;

143 calibrateSorter();

144 }

145

146 *//state 2*

147 void resting() {

148 timerManage();

149 *//the program waits for the user to*

150 *// press the start/stop*

151 $startStop = getButtonPressed(0);

152 **if** ($startStop == 1) {

153 *//sleep so we don't go to the pause*

154 *// immediatly*

155 sleep(2000);

156 *//power up the lights*

157 storeData(12, "outputs",

158 LENSLAMPPOSITION);

159 storeData(12, "outputs",

160 LENSLAMPSORTER);

161 *//start up the belt and the feeder*

162 storeData(9, "outputs", CONVEYORBELT);

163 storeData(5, "outputs", FEEDERENGINE);

164 *//set and start the countdown*

165 setCountdown(BELTROUND + BELT);

166 startCountdown();

167 *//update the state*

168 $state = 3;

169 running();

170 }

171 *//loop*

172 resting();

173 }

174

175 *//state 3*

176 void running() {

177 timerManage();

178 *//check if we need to pause*

179 $startStop = getButtonPressed(0);

180 **if** ($startStop == 1) {

181 *//stop the feeder engine*

182 storeData(0, "outputs", FEEDERENGINE);

183 *//set the timer*

184 setCountdown(BELT);

185 *//update the state*

186 $state = 9;

187 runningTimer();

188 }

189 *//check if a disk is at the position*

190 *// detector*

191 $position = getButtonPressed(7);

192 **if** ($position == 1) {

193 *//reset the countdown,because a*

194 *// disk was detected*

195 setCountdown(BELTROUND + BELT);

196 *//update the state*

197 $state = 4;

198 runningWait();

199 }

200 *//loop*

201 running();

202 }

203

204 void runningWait() {

205 timerManage();

206 *//check if we need to pause*

207 $startStop = getButtonPressed(0);

208 **if** ($startStop == 1) {

209 *//stop the feeder engine*

210 storeData(0, "outputs", FEEDERENGINE);

211 *//set the timer*

212 setCountdown(BELT);

213 *//update the state*

214 $state = 9;

215 runningTimer();

216 }

217 *//check if a disk is at the positiond*

218 *// detector*

219 $position = getButtonPressed(7);

220 **if** ($position == 1) {

221 *//reset the countdown,because a*

222 *// disk was detected*

223 setCountdown(BELTROUND + BELT);

224 *//update the state*

225 $state = 5;

226 runningTimerReset();

227 }

228 *//check if a white disk is at the color*

229 *// detector*

230 $colour = getButtonPressed(6);

231 **if** ($colour == 1) {

232 *//move the sorter up*

233 storeData(9, "outputs", HBRIDGE0);

234 *//update the state*

235 $state = 6;

236 motorUp();

237 }

238 *//loop*

239 runningWait();

240 }

241

242 *//state 5*

243 void runningTimerReset() {

244 timerManage();

245 *//update the state*

246 $state = 5;

247 runningWait();

248 }

249

250 *//state 6*

251 void motorUp() {

252 timerManage();

253 *//check if we need to pause*

254 $startStop = getButtonPressed(0);

255 **if** ($startStop == 1) {

256 *//stop the feeder engine*

257 storeData(0, "outputs", FEEDERENGINE);

258 *//set the timer*

259 setCountdown(BELT);

260 motorUpTimer();

261 }

262 *//check if the sorter push button is*

263 *// pressed*

264 $push = getButtonPressed(5);

265 **if** ($push == 1) {

266 *//stop the engine,because it is in*

267 *// the right position*

268 storeData(0, "outputs", HBRIDGE0);

269 *//update the state*

270 $state = 7;

271 whiteWait();

272 }

273 *//loop*

274 motorUp();

275 }

276

277 *//state 7*

278 void whiteWait() {

279 timerManage();

280 *//we are waiting for the white disk to*

281 *// be sorted*

282 **if** ($sleep == SORT \* 1000) {

283 *//start moving the sorter down*

284 storeData(9, "outputs", HBRIDGE1);

285 *//update the state*

286 $state = 8;

287 *//reset sleep for the next function*

288 $sleep = 0;

289 motorDown();

290

291 }

292 *//check if we need to pause*

293 $startStop = getButtonPressed(0);

294 **if** ($startStop == 1) {

295 *//stop the feeder engine*

296 storeData(0, "outputs", FEEDERENGINE);

297 *//set the timer*

298 setCountdown(BELT);

299 *//update the state*

300 $state = 11;

301 whiteWaitTimer();

302 }

303 *//loop*

304 $sleep++;

305 whiteWait();

306 }

307

308 *//state 8*

309 void motorDown() {

310 timerManage();

311 *//the sorter is moving down*

312 **if** ($sleep == TIMEMOTORDOWN \* 1000) {

313 *//stop the sorter*

314 storeData(0, "outputs", HBRIDGE1);

315 *//update the state*

316 $state = 9;

317 *//reset sleep for the next function*

318 $sleep = 0;

319 runningWait();

320 }

321 *//check if we need to pause*

322 $startStop = getButtonPressed(0);

323 **if** ($startStop == 1) {

324 *//stop the feeder engine*

325 storeData(0, "outputs", FEEDERENGINE);

326 *//set the timer*

327 setCountdown(BELT);

328 motorDownTimer();

329 }

330 *//loop*

331 $sleep++;

332 motorDown();

333

334 }

335

336 *//state 9*

337 void runningTimer() {

338 timerManage();

339 *//update state*

340 $state = 13;

341 runningStop();

342 }

343

344 *//state 10*

345 void motorUpTimer() {

346 timerManage();

347 *//update state*

348 $state = 14;

349 motorUpStop();

350 }

351

352 *//state 11*

353 void whiteWaitTimer() {

354 timerManage();

355 *//update state*

356 $state = 15;

357 whiteWaitStop();

358 }

359

360 *//state 12*

361 void motorDownTimer() {

362 timerManage();

363 *//update state*

364 $state = 16;

365 motorDownStop();

366 }

367

368 *//state 13*

369 void runningStop() {

370 timerManage();

371 *//check if a white disk is at the*

372 *// colour detector*

373 $colour = getButtonPressed(6);

374 **if** ($colour == 1) {

375 *//move the sorter engine up*

376 storeData(9, "outputs", HBRIDGE0);

377 *//update the state*

378 $state = 10;

379 motorUpStop();

380 }

381 *//loop*

382 runningStop();

383 }

384

385 *//state 14*

386 void motorUpStop() {

387 timerManage();

388 *//check if the sorter push button is*

389 *// pressed*

390 $push = getButtonPressed(5);

391 **if** ($push == 1) {

392 *//stop the engien for the sorter*

393 storeData(0, "outputs", HBRIDGE0);

394 *//update the state*

395 $state = 11;

396 whiteWaitStop();

397 }

398 motorUpStop();

399 }

400

401 *//state 15*

402 void whiteWaitStop() {

403 timerManage();

404 *//check if the white disk has been sorted*

405 **if** ($sleep == SORT \* 1000) {

406 *//start moving the sorter down*

407 storeData(9, "outputs", HBRIDGE1);

408 *//update the state*

409 $state = 12;

410 *//reset the sleep for the next*

411 *// function*

412 $sleep = 0;

413 motorDown();

414 }

415 *//loop*

416 $sleep++;

417 whiteWaitStop();

418 }

419

420 *//state 16*

421 void motorDownStop() {

422 timerManage();

423 *//check if the sorter has moved down*

424 **if** ($sleep == TIMEMOTORDOWN) {

425 *//stop the engine of the sorter*

426 storeData(0, "outputs", HBRIDGE1);

427 *//update the state*

428 $state = 9;

429 *//reset sleep for the next function*

430 $sleep = 0;

431 runningWait();

432 }

433 *//loop*

434 $sleep++;

435 motorDownStop();

436 }

437

438 *//not a state*

439 void timerInterrupt() {

440 *//show that we have timer interrupt*

441 $state = 18;

442 *//make the sorter move up*

443 storeData(9, "outputs", HBRIDGE0);

444 *//stop all other outputs*

445 storeData(0, "outputs", HBRIDGE1);

446 storeData(0, "outputs", LENSLAMPPOSITION);

447 storeData(0, "outputs", LENSLAMPSORTER);

448 storeData(0, "outputs",

449 LEDSTATEINDICATOR);

450 storeData(0, "outputs", DISPLAY);

451 storeData(0, "outputs", CONVEYORBELT);

452 storeData(0, "outputs", FEEDERENGINE);

453 *//make sure that the outputs get set*

454 *// immediatly*

455 timerManage();

456 *//set the display to the state of initial*

457 $state = 0;

458

459 initial();

460

461 }

462

463 void abort() {

464 *//stop all outputs*

465 storeData(0, "outputs", HBRIDGE0);

466 storeData(0, "outputs", HBRIDGE1);

467 storeData(0, "outputs", LENSLAMPPOSITION);

468 storeData(0, "outputs", LENSLAMPSORTER);

469 storeData(0, "outputs",

470 LEDSTATEINDICATOR);

471 storeData(0, "outputs", DISPLAY);

472 storeData(0, "outputs", CONVEYORBELT);

473 storeData(0, "outputs", FEEDERENGINE);

474 *//make sure the outputs stop immediatly*

475 timerManage();

476 *//update the state to be correct in*

477 *// aborted*

478 $state = 17;

479 aborted();

480

481 }

482

483 *//state 17*

484 void aborted() {

485 timerManage();

486 *//check if we can start again*

487 $startStop = getButtonPressed(0);

488 **if** ($startStop == 1) {

489 *//start moving the sorter up for*

490 *// calibration*

491 storeData(1, "outputs", HBRIDGE0);

492 *//update the state*

493 $state = 0;

494 initial();

495 }

496 *//loop*

497 aborted();

498

499 }

500

501 void timerManage() {

502

503

504 *//make sure that when counter can not*

505 *// be higher than 12*

506 mod(13, $counter);

507 *//get the voltage of output $location*

508 int $voltage = getData("outputs",

509 $location);

510 *//power up the output when it needs to*

511 **if** ($voltage > $counter) {

512 $engines += pow(2, $voltage);

513 }

514 *//check if we are in a new itteration*

515 **if** ($counter == 0) {

516 *//set the first part of the display*

517 $temp = getData("state", 0);

518 mod(10, $temp);

519 display($temp, "display", "1");

520

521

522 }

523 *//check if we are at the end of the*

524 *// itteration*

525 **if** ($counter == 12) {

526 *//set the second part of the display;*

527 $temp = getData("state", 0);

528 $temp = $temp / 10;

529 mod(10, $temp);

530 display($temp, "display", "01");

531

532 }

533 *//check if we did all outputs*

534 **if** ($location > 7) {

535 display($engines, "leds", "");

536 *//set the variables for the next run*

537 $engines = 0;

538 $location = 0;

539 $counter++;

540

541 *//check if abort is pressed*

542 $abort = getButtonPressed(1);

543 **if** ($abort == 1) {

544 abort();*//stop the machine*

545 }

546 **return**;

547 }

548

549

550 $location++;

551 timerManage();

552 }

553 }