1 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 Header file for shooter control software

3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

4

5 #ifndef ShooterControl\_H

6 #define ShooterControl\_H

7

8 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

9

10 // General

11 void **InitShooterControl**( void );

12 void **InitReloaderRequestControl** ( void );

13

14 // Ball counting

15 void **IncrementBallCounter**( void );

16 void **DecrementBallCounter**( void );

17 unsigned char **GetNumBalls**( void );

18

19 // Magazine control

20 void **SendBallRequest**( void );

21 void **LoadOneBall**( void );

22 void **ShootOneBall**( void );

23 void **RotateMagazine**( void );

24 void **PanServo**( void );

25

26 void **RotateToFull**( void );

27 void **RotateToEmpty**( void );

28

29 // Fan power

30 void **TurnFanOn**( void );

31 void **TurnFanOff**( void );

32

33 /\* Ball counting \*/

34 #define MAX\_BALLS 5

35 #define NO\_BALLS 0

36

37

38 #endif /\* ShooterControl\_H \*/

1 // #define CALIBRATE

2 // #define PRINT

3 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4 Module

5 ShooterControl.c

6

7 Description

8 This service provides functions for all of hardware manipulation necessary

9 to run ball shooter magazine, fan, and reloader

10

11 Notes

12

13 History

14 When Who What/Why

15 -------------- --- --------

16 2/24/14 whg created for testing with empty methods

17 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

18 /\*----------------------------- Include Files -----------------------------\*/

19 #include "ES\_Configure.h"

20 #include "ES\_Framework.h"

21

22 #include "ShooterControl.h" // My header

23 #include "E128\_ServoLib.h" // For servo control

24 #include <s12evec.h> // For interrupt

25 #include "GameMasterHSM.h" // For post

26

27

28 /\*----------------------------- Module Defines ----------------------------\*/

29 // Ball defines

30 #define ONE\_BALL 1

31 #define EMPTY NO\_BALLS

32 #define FULL MAX\_BALLS

33 #define NUM\_MAG\_SLOTS 6

34 #define NUM\_SERVO\_POSITIONS (NUM\_MAG\_SLOTS + 1)

35

36

37 #define RELOAD\_EMPTY\_SLOT 800

38 #define RELOAD\_1 1090

39 #define RELOAD\_2 1410

40 #define RELOAD\_3 1710

41 #define RELOAD\_4 2000

42 #define RELOAD\_5 2300

43

44

45 #define SHOOT\_EMPTY\_SLOT 2270

46 #define SHOOT\_1 1970

47 #define SHOOT\_2 1700

48 #define SHOOT\_3 1360

49 #define SHOOT\_4 1060

50 #define SHOOT\_5 760

51

52 #define EMPTY\_END\_POSITION 500

53 #define BALL\_FIVE\_POSITION 820

54 #define BALL\_FOUR\_POSITION 1130

55 #define BALL\_THREE\_POSITION 1420

56 #define BALL\_TWO\_POSITION 1730

57 #define BALL\_ONE\_POSITION 2020

58 #define EMPTY\_START\_POSITION 2340

59

60

61 // Pan Defines

62 #define PAN\_STEP 10

63 #define TO\_EMPTY 1

64 #define TO\_BALL (-1\*TO\_EMPTY)

65

66 // For calibration

67 #define SLOT\_STEP 380

68

69 // For Ball request timer

70 #define \_IN\_MS\_ \* 375

71 #define THIRTY\_MS (30 \_IN\_MS\_)

72 #define TEN\_MS (10 \_IN\_MS\_)

73

74 #define MAX\_NUM\_COMPARES 20

75 /\*---------------------------- Module Functions ---------------------------\*/

76 static void **RequestBall**(void);

77 static void **PanShoot**(void);

78 static void **PanLoad** (void);

79

80 /\*---------------------------- Module Variables ---------------------------\*/

81 static unsigned char numBallsRemaining; // Number of balls current in magazine

82 static signed char PanStepCount; // Number of pan steps taken

83 static signed char PanDirection; // Direction to pan in [ 1 or -1 ]

84 static signed int PositionDelta; // Amount to move between positions

85

86 unsigned int ReloadPositions[NUM\_MAG\_SLOTS] =

87 {RELOAD\_EMPTY\_SLOT,

88 RELOAD\_1, RELOAD\_2, RELOAD\_3, RELOAD\_4, RELOAD\_5};

89

90 unsigned int ShootPositions[NUM\_MAG\_SLOTS] =

91 {SHOOT\_5, SHOOT\_4, SHOOT\_3, SHOOT\_2, SHOOT\_1,

92 SHOOT\_EMPTY\_SLOT};

93

94

95 static unsigned int curWidth = 100; // in microseconds

96

97 // module level variable to keep track of how many pulses we have given so far

98 // to the reloader

99 static unsigned char CurrentNumCompares = 0;

100

101

102 /\*------------------------------ Module Code ------------------------------\*/

103 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

104 Function

105 InitShooterControl

106

107 Parameters

108 --

109

110 Returns

111 bool, false if error in initialization, true otherwise

112

113 Description

114 Inits pins/hardware necessary to run ball shooter

115 Notes

116

117 Author

118 J. Edward Carryer, 01/16/12, 10:00

119 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

120 void **InitShooterControl** ( void )

121 {

122 // Init necessary pins

123 // E128\_Servo12\_Init("xSxx"); - done in general init

124

125 // Set shooter to initially full

126 PanDirection = TO\_BALL;

127 numBallsRemaining = FULL;

128 **E128\_Servo12\_SetPulseWidth**(BALL\_SERVO\_CHANNEL, EMPTY\_START\_POSITION);

129

130 #ifdef CALIBRATE

131 curWidth = EMPTY\_START\_POSITION;

132 **E128\_Servo12\_SetPulseWidth**(BALL\_SERVO\_CHANNEL, curWidth);

133 #endif

134

135 // set fan motor pin to output and then set it low

136 FAN\_MOTOR\_REG |= FAN\_MOTOR\_PIN;

137 FAN\_MOTOR\_PORT &= ~FAN\_MOTOR\_PIN;

138 }

139

140 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

141 Function

142 InitReloaderRequestControl

143

144 Parameters

145 --

146

147 Returns

148 bool, false if error in initialization, true otherwise

149

150 Description

151 Inits pins/hardware necessary to request a ball from the reloader.

152 The reloader LEDs use an output compare on pin T6.

153

154 Notes

155

156 Author

157 J. Edward Carryer, 01/16/12, 10:00

158 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

159 void **InitReloaderRequestControl** ( void )

160 {

161

162 // set Timer 1 cap/comp 4 (Pin T4) to output compare.

163 TIM1\_TIOS |= \_S12\_IOS4;

164

165 // disconnect from pin - will toggle manually

166 TIM1\_TCTL1 &= ~(\_S12\_OL4 | \_S12\_OM4);

167

168 /\*

169 // set pin to toggle on output compare

170 TIM1\_TCTL1 |= \_S12\_OL4;

171 TIM1\_TCTL1 &= ~\_S12\_OM4;

172 \*/

173 // clear the OC4 flag

174 TIM1\_TFLG1 = \_S12\_C4F;

175 // disable OC4 interrupts

176 TIM1\_TIE &= ~\_S12\_C4I;

177

178 // Enable all interrupts

179 // EnableInterrupts; - handled in general init

180

181 // set pin T4 to be an output

182 RELOADER\_LED\_REG |= RELOADER\_LED\_PIN;

183 // set the pin low

184 RELOADER\_LED\_PORT &= ~RELOADER\_LED\_PIN;

185 }

186

187 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

188 Function

189 IncrementBallCounter

190

191 Parameters

192 --

193

194 Returns

195 --

196

197 Description

198 Adds 1 to number of balls in magazine - separate for debugging purposes

199

200 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

201 void **IncrementBallCounter**( void )

202 {

203 numBallsRemaining += ONE\_BALL;

204

205 #ifdef CALIBRATE

206 curWidth += PAN\_STEP; // increment small step

207 #endif

208 }

209

210 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

211 Function

212 DecrementBallCounter

213

214 Parameters

215 --

216

217 Returns

218 --

219

220 Description

221 Subtracts 1 to number of balls in magazine - separate for debugging purposes

222

223 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

224 void **DecrementBallCounter**( void )

225 {

226 numBallsRemaining -= ONE\_BALL;

227

228 #ifdef CALIBRATE

229 curWidth += SLOT\_STEP; // increment large step

230 #endif

231 }

232

233 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

234 Function

235 GetNumBalls

236

237 Parameters

238 --

239

240 Returns

241 Number of balls currently in magazine

242

243 Description

244 Subtracts 1 to number of balls in magazine

245

246 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

247 unsigned char **GetNumBalls**( void )

248 {

249

250 return numBallsRemaining;

251 }

252

253

254

255

256

257 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

258 Function

259 SendBallRequest

260

261 Parameters

262 --

263

264 Returns

265 --

266

267 Description

268 Runs through ball request

269

270 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

271 void **SendBallRequest**( void )

272 {

273 **RequestBall**();

274 }

275

276 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

277 Function

278 LoadOneBall

279

280 Parameters

281 --

282

283 Returns

284 --

285

286 Description

287 Rotates magazine servo to a new empty spot

288

289 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

290 void **LoadOneBall**( void )

291 {

292 // Rotate magazine to an empty spot

293 PositionDelta = ReloadPositions[numBallsRemaining] - ReloadPositions[numBallsRemaining - 1];

294 PanStepCount = 0; // clear step count

295 // Move magazine slowly for easy loading

296 **PanServo**(); // Pan servo across position

297 }

298

299 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

300 Function

301 ShootOneBall

302

303 Parameters

304 --

305

306 Returns

307 --

308

309 Description

310 Rotates magazine servo to a spot containing a ball

311

312 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

313 void **ShootOneBall**( void )

314 {

315 // Rotate magazine to a spot with a ball

316 **E128\_Servo12\_SetPulseWidth**(BALL\_SERVO\_CHANNEL, ShootPositions[numBallsRemaining]);

317 // Move magazine slowly for easy shooting

318 // No panning

319 }

320

321 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

322 Function

323 Rotate mag to position

324

325 Parameters

326 --

327

328 Returns

329 --

330

331 Description

332 Rotates magazine servo to ball position - separate to allow manually moving magazine

333

334 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

335 void **RotateMagazine**( void )

336 {

337 // Rotate magazine to a spot depending on number of balls

338 // E128\_Servo12\_SetPulseWidth(BALL\_SERVO\_CHANNEL, ServoPositions[MAX\_BALLS - numBallsRemaining]);

339

340 #ifdef CALIBRATE

341 **E128\_Servo12\_SetPulseWidth**(BALL\_SERVO\_CHANNEL, curWidth);

342 #endif

343 }

344

345 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

346 Function

347 RotateToFull

348

349 Parameters

350 --

351

352 Returns

353 --

354

355 Description

356 Rotates magazine servo to full position (5 balls, empty in barrel)

357

358 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

359 void **RotateToFull**( void )

360 {

361 // Rotate magazine to a spot depending on number of balls

362 **E128\_Servo12\_SetPulseWidth**(BALL\_SERVO\_CHANNEL, EMPTY\_START\_POSITION);

363 }

364

365 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

366 Function

367 RotateToEmpty

368

369 Parameters

370 --

371

372 Returns

373 --

374

375 Description

376 Rotates magazine servo to Empty position (5 balls, empty in barrel)

377

378 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

379 void **RotateToEmpty**( void )

380 {

381 // Rotate magazine to a spot depending on number of balls

382 **E128\_Servo12\_SetPulseWidth**(BALL\_SERVO\_CHANNEL, EMPTY\_END\_POSITION);

383 }

384

385

386

387

388

389 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

390 Function

391 TurnFanOn

392

393 Parameters

394 --

395

396 Returns

397 --

398

399 Description

400 Turns ball launcher fan on

401

402 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

403 void **TurnFanOn**( void )

404 {

405 FAN\_MOTOR\_PORT |= FAN\_MOTOR\_PIN;

406 }

407

408 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

409 Function

410 TurnFanOff

411

412 Parameters

413 --

414

415 Returns

416 --

417

418 Description

419 Turns ball launcher fan off

420

421 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

422 void **TurnFanOff**( void )

423 {

424 FAN\_MOTOR\_PORT &= ~FAN\_MOTOR\_PIN;

425 }

426

427 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

428 private functions

429 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

430

431 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

432 Function

433 PanServo

434

435 Parameters

436 --

437

438 Returns

439 --

440

441 Description

442 Slowly moves servo through center position

443

444 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

445 void **PanServo**( void )

446 {

447 static unsigned int PositionOffset;

448 signed char PanOffset;

449 signed int NumSteps = (PositionDelta / PAN\_STEP);

450

451 // Increase step count

452 PanStepCount++;

453 // Calculate new offset based on current ball position

454 PanOffset = (NumSteps - PanStepCount);

455 PositionOffset = PanOffset \* PAN\_STEP;

456 // Move Servo

457 **E128\_Servo12\_SetPulseWidth**(BALL\_SERVO\_CHANNEL, ReloadPositions[numBallsRemaining] - PositionOffset);

458

459 // Restart timer if more steps are to be taken

460 if (PanStepCount < NumSteps) **ES\_Timer\_InitTimer**(PAN\_TIMER, PAN\_DURATION);

461 }

462

463 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

464 Module

465 ReloaderRequestControl.c

466

467 Description

468 This service provides functions for all of hardware manipulation necessary

469 to request a ball from the reloader using an output compare toggle

470 interrupt response routine.

471

472 HOW THIS \*SHOULD\* WORK. (Different than the pseudo code on the drive.)

473 -----------------------

474 You call RequestBall. That initially sets the LED output pin high,

475 schedules the first output compare for the long pulse time (30ms), and sets

476 the whole process in motion.

477 When this first output compare happens, the line should toggle low.

478 The next output compare will then be set for the low pulse time (10ms).

479 On the next output compare (no. 2), the line should toggle high, and the

480 next output compare will then be set for the long pulse time.

481 This cycle should continue until the 10th high time (18 output compares),

482 on which the line will toggle high.

483 On the 19th output compare, the line will toggle low, and the interrupt

484 response routine will NOT schedule a future compare. Rather, it will just

485 reset the variable CurrentNumCompares to 0. This way, we let the line stay

486 low, thereby ending the whole series of 10 hi-lo pulses. And there was much rejoicing.

487 Now the robot is waiting for a ball. When we detect the ball using

488 the IR disruption sensor, you call RequestBall again. This whole cycle

489 repeats until you have asked for 5 balls.

490

491 If this OC system winds up not working after debugging, there are two other possible

492 ways to do this is. The first is to simply disconnect the timer from the pin and to

493 control the pin toggling manually.

494 The second is to use a single output compare (DISCONNECTED FROM THE PIN)

495 in conjunction with a 75% duty cycle PWM on the same pin with a period of 40 ms.

496 You set the output compare for 375 ms as you start the PWM. When the OC

497 happens, you turn off the PWM. 375 ms cuts it off in the middle of the last

498 10ms low time, thereby ending the whole process.

499

500

501 Notes

502

503 History

504 When Who What/Why

505 -------------- --- --------

506 2/24/14 whg created for testing with empty methods

507 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

508

509 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

510 Function

511 RequestBall starts ball request process by setting timer and enabling interrupt

512 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

513 static void **RequestBall**(void)

514 {

515 // Make sure pin is low

516 RELOADER\_LED\_PORT &= ~RELOADER\_LED\_PIN;

517 // enable OC4 interrupts

518 TIM1\_TIE |= \_S12\_C4I;

519 // schedule the next compare

520 TIM1\_TC4 = TIM1\_TCNT + TEN\_MS;

521 // clear the OC4 flag

522 TIM1\_TFLG1 = \_S12\_C4F;

523 }

524

525 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

526 Function

527 PulseControl pulses LED pin according to a timer output compare

528 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

529 void interrupt \_Vec\_tim1ch4 **PulseControl** (void)

530 {

531 CurrentNumCompares++;

532 // toggle the line

533 RELOADER\_LED\_PORT ^= RELOADER\_LED\_PIN;

534

535 // schedule another compare (either long or short pulse) as long as we are

536 // at fewer than MAX number of compares

537 if (CurrentNumCompares < MAX\_NUM\_COMPARES) {

538 // on odd compares (ie compare 1) - set short time until next fall

539 if (CurrentNumCompares % 2) {

540 TIM1\_TC4 = TIM1\_TCNT + TEN\_MS;

541 } else { // on even count - set long time until next rise

542 TIM1\_TC4 = TIM1\_TCNT + THIRTY\_MS;

543 }

544 }

545 // once we have reached the MAX number of compares, we want to stay low,

546 // so we let the OC toggle the pin low and will not schedule another compare

547 else {

548 // Post request complete event

549 ES\_Event NewEvent = {EV\_REQUEST\_SENT, 0};

550 **PostGameMasterHSMS**(NewEvent);

551 // Reset compare count

552 CurrentNumCompares = 0;

553 // Make sure pin is low

554 RELOADER\_LED\_PORT &= ~RELOADER\_LED\_PIN;

555 // disable OC4 interrupts

556 TIM1\_TIE &= ~\_S12\_C4I;

557 }

558

559 // clear the OC4 flag

560 TIM1\_TFLG1 = \_S12\_C4F;

561 }

562

563

564 /\*------------------------------- Footnotes -------------------------------\*/

565 /\*------------------------------ End of file ------------------------------\*/