Московский государственный технический университет им. Н.Э. Баумана

Контроллер шагового и асинхронного двигателей.

Текст Программы

РОФ.МГТУ.000001-01 11

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. *N* дубл. |  |
| Взам. Инв. *N* |  |
| Подп. и дата |  |
| Инв. *N* подп. |  |

Листов 16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Проверил | |  | Рафиков А.Г. | |
|  | (подпись, дата) | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Разработал | |  | Малютин Р.С. | |
|  | (подпись, дата) | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  | Храпов Н.А. | |
|  | (подпись, дата) | |  |

АННОТАЦИЯ

В данном программном документе приведен текст программы для контроллера шагового и асинхронного двигателей. Текст программы реализован в виде символической записи на исходном языке. Исходным языком данной разработки являются C, в качестве интерфейса разработана HTML страница. Среды разработки: MikroC PRO for PIC 7.2.0, Visual Studio Code.

Основными функциями аппаратной программы являются конфигурирование серверной части, управление направлением, скоростью и углом поворота шагового двигателя, управлением направлением, скоростью и углом поворота асинхронного двигателя.

Основной функцией интерфейса является взаимодействие пользователя с контроллером.

Оформление программного документа “Текст программы” произведено по требованиям ЕСПД (ГОСТ 19.101-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.410-78, ГОСТ 19.604-78).

1 ТЕКСТ АППАРАТНОЙ ПРОГРАММЫ НА ИСХОДНОМ ЯЗЫКЕ

main.c

1 // Объявление заголовков и статусов

2 const char HTTPheaderErr[] = "HTTP/1.1 400 Bad Request\nAccess-Control-Allow-Origin:\*\nContent-type:";

3 const char HTTPheader[] = "HTTP/1.1 200 OK\nAccess-Control-Allow-Origin:\*\nContent-type:";

4 const char HTTPMimeTypeHTML[] = "text/html\n\n";

5 const char OKStatus[] = "OK";

6 const char FormatError[] = "Wrong command format";

7 const char OptionError[] = "Wrong option";

8 const char SpeedError[] = "Failed to parse speed";

9 const char AngleError[] = "Failed to parse angle";

10

11 // Интерфейсы для работы с Ethernet

12 sfr sbit SPI\_Ethernet\_Rst at RC0\_bit;

13 sfr sbit SPI\_Ethernet\_CS at RC1\_bit;

14 sfr sbit SPI\_Ethernet\_Rst\_Direction at TRISC0\_bit;

15 sfr sbit SPI\_Ethernet\_CS\_Direction at TRISC1\_bit;

16

17 unsigned char MACAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19, 0x3f};

18 unsigned char IPAddr[4] = {10, 211, 55, 5};

19 unsigned char getRequest[20];

20

21 unsigned int async\_step = 0;

22 unsigned int motor\_speed, i, j;

23

24 // Структура для работы с Ethernet

25 typedef struct {

26 unsigned canCloseTCP : 1;

27 unsigned isBroadcast : 1;

28 } TEthPktFlags;

29

30 // Структура текущих конфигураций

31 typedef struct {

32 int motor\_type;

33 int running;

34 int right;

35 int delay;

36 int tick\_counter;

37 int angle;

38 int angle\_half;

39 int is\_half\_step;

40 int steps\_counter;

41 int port\_value;

42 } Config;

43

44 // Инициализация конфигурационной переменной

45 Config cfg = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

46

47 // Функция для очистки конфигурационной переменной

48 void emptyCfg() {

49 cfg.motor\_type = 0;

50 cfg.running = 0;

51 cfg.right = 0;

52 cfg.delay = 0;

53 cfg.tick\_counter = 0;

54 cfg.angle = 0;

55 cfg.angle\_half = 0;

56 cfg.is\_half\_step = 0;

57 cfg.steps\_counter = 0;

58 cfg.port\_value = 0;

59 }

60

61 // ----------------------------------------

62 // Асинхронный мотор

63 // ----------------------------------------

64

65 // Управления рабочим циклом ШИМа

66 void set\_pwm\_duty(unsigned int pwm\_duty)

67 {

68 CCP1CON = ((pwm\_duty << 4) & 0x30) | 0x0C;

69 CCPR1L = pwm\_duty >> 2;

70 }

71

72 // Функция инициализации выходов для асинхронного двигателя

73 void initAsync() {

74 TRISD = 0;

75 PORTD = 0;

76 INTCON = 0xC0;

77 C1IF\_bit = 0;

78 CCP1CON = 0x0C;

79 CCPR1L = 0;

80 set\_pwm\_duty((unsigned int) cfg.delay);

81 }

82

83 // Обработка шагов асинхронного двигателя

84 void asyncMove()

85 {

86 switch(async\_step){

87 case 0:

88 CCP1CON = 0; // Выкл ШИМ

89 PORTD = 0x08;

90 PSTRCON = 0x08; // Выход ШИМ на RD7

91 CCP1CON = 0x0C; // Вкл ШИМ

92 CM1CON0 = 0xA2; // BEMF C

93 break;

94 case 1:

95 PORTD = 0x04;

96 CM1CON0 = 0xA1; // BEMF B

97 break;

98 case 2:

99 CCP1CON = 0; // Выкл ШИМ

100 PORTD = 0x04;

101 PSTRCON = 0x04; // Выход ШИМ на RD6

102 CCP1CON = 0x0C; // Вкл ШИМ

103 CM1CON0 = 0xA0; // BEMF A

104 break;

105 case 3:

106 PORTD = 0x10;

107 CM1CON0 = 0xA2; // BEMF C

108 break;

109 case 4:

110 CCP1CON = 0; // Выкл ШИМ

111 PORTD = 0x10;

112 PSTRCON = 0x02; // Выход ШИМ на RD5

113 CCP1CON = 0x0C; // Вкл ШИМ

114 CM1CON0 = 0xA1; // BEMF B

115 break;

116 case 5:

117 PORTD = 0x08;

118 CM1CON0 = 0xA0; // BEMF A

119 break;

120 }

121 async\_step++;

122 if(async\_step >= 6)

123 async\_step = 0;

124 }

125

126 // Обработчик движения асинхронным потором

127 void HandleAsyncMotor() {

128 int i = cfg.angle;

129 while(i > 0)

130 {

131 j = cfg.delay;

132 while(j--) ;

133 asyncMove();

134 i = i - 1;

135 }

136 }

137

138 // ----------------------------------------

139 // Шаговый мотор

140 // ----------------------------------------

141

142 // Обработчик движения шаговым потором

143 void HandleStepperMotor() {

144 int new\_port\_value = 0b0000;

145 if (cfg.running) {

146 new\_port\_value |= 0b0001;

147 }

148 new\_port\_value = 0b0001;

149 if (cfg.is\_half\_step == 1) {

150 new\_port\_value |= 0b0010;

151 }

152 switch (cfg.right) {

153 case 0:

154 new\_port\_value |= 0b0100;

155 break;

156 case 1:

157 new\_port\_value &= 0b1011;

158 break;

159 }

160 cfg.port\_value = new\_port\_value;

161 }

162

163 // Обработка команды на основе полученной

164 // конфигурации

165 void run() {

166 while (cfg.running) {

167 if (cfg.motor\_type == 0) {

168 HandleStepperMotor();

169 ++cfg.tick\_counter;

170 if (cfg.tick\_counter >= cfg.delay) {

171 cfg.port\_value ^= 0b1000;

172 cfg.tick\_counter = 0;

173 ++cfg.steps\_counter;

174 }

175 PORTB = cfg.port\_value;

176 if (cfg.is\_half\_step == 0 && cfg.steps\_counter == cfg.angle) {

177 if (cfg.angle\_half == 0) {

178 emptyCfg();

179 continue;

180 }

181 cfg.is\_half\_step = 1;

182 cfg.steps\_counter = 0;

183 } else if (cfg.is\_half\_step == 1 && cfg.steps\_counter == cfg.angle\_half) {

184 emptyCfg();

185 }

186 } else if (cfg.motor\_type == 1) {

187 initAsync();

188 HandleAsyncMotor();

189 emptyCfg();

190 }

191 }

192 }

193

194 // Обработчик Ethernet TCP запросов

195 unsigned int SPI\_Ethernet\_UserTCP(unsigned char \*remoteHost,

196 unsigned int remotePort,

197 unsigned int localPort,

198 unsigned int reqLength, TEthPktFlags \*flags) {

199 unsigned int length;

200 char\* freqEnd;

201 int angleStart;

202

203 for (length = 0; length < 20; ++length) {

204 getRequest[length] = SPI\_Ethernet\_getByte();

205 }

206 getRequest[length] = 0;

207

208 if (memcmp(getRequest, "GET /", 5)) {

209 return (0);

210 }

211 if (localPort != 80) {

212 return (0);

213 }

214

215 if (!memcmp(getRequest + 5, "OFF", 3)) {

216 emptyCfg();

217 length = SPI\_Ethernet\_putConstString(HTTPheader);

218 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

219 length += SPI\_Ethernet\_putConstString(OKStatus);

220 return length;

221 }

222 /\*

223 ST - Шаговый

224 AC - Асинхронный

225 \*/

226 if (!memcmp(getRequest + 5, "ST", 2)) {

227 cfg.motor\_type = 0;

228 } else if (!memcmp(getRequest + 5, "AC", 2)) {

229 cfg.motor\_type = 1;

230 } else {

231 length = SPI\_Ethernet\_putConstString(HTTPheaderErr);

232 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

233 length += SPI\_Ethernet\_putConstString(OptionError);

234 return length;

235 }

236

237 // "," символ

238 if (memcmp(getRequest + 7, ",", 1)) {

239 length = SPI\_Ethernet\_putConstString(HTTPheaderErr);

240 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

241 length += SPI\_Ethernet\_putConstString(FormatError);

242 return length;

243 }

244

245 /\*

246 R - Вправо

247 L - Влево

248 \*/

249 if (!memcmp(getRequest + 8, "R", 1)) {

250 cfg.right = 1;

251 } else if (!memcmp(getRequest + 8, "L", 1)) {

252 cfg.right = 0;

253 } else {

254 length = SPI\_Ethernet\_putConstString(HTTPheaderErr);

255 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

256 length += SPI\_Ethernet\_putConstString(OptionError);

257 return length;

258 }

259

260 // "," символ

261 if (memcmp(getRequest + 9, ",", 1)) {

262 length = SPI\_Ethernet\_putConstString(HTTPheaderErr);

263 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

264 length += SPI\_Ethernet\_putConstString(FormatError);

265 return length;

266 }

267

268 // Достает число из запроса, которое отвечает за скорость

269 cfg.delay = atoi(getRequest + 10);

270 if (cfg.delay == 0) {

271 length = SPI\_Ethernet\_putConstString(HTTPheaderErr);

272 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

273 length += SPI\_Ethernet\_putConstString(SpeedError);

274 return length;

275 }

276 if (!cfg.motor\_type) {

277 cfg.delay = 18512 / cfg.delay;

278 }

279

280 // Достает число из запроса, которое отвечает за угол поворота

281 freqEnd = strchr(getRequest + 10, ',');

282 angleStart = (int)(freqEnd - getRequest) + 1;

283 cfg.angle = atoi(getRequest + angleStart);

284 if (cfg.angle == 0) {

285 length = SPI\_Ethernet\_putConstString(HTTPheaderErr);

286 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

287 length += SPI\_Ethernet\_putConstString(AngleError);

288 return length;

289 }

290

291 switch (cfg.motor\_type)

292 {

293 case 0:

294 ++cfg.angle;

295 cfg.angle\_half = (int)((float) cfg.angle / 1.8);

296 if (((float) cfg.angle / 1.8) - (float)cfg.angle\_half > 0.0) {

297 cfg.angle\_half = 1;

298 } else {

299 cfg.angle\_half = 0;

300 }

301 cfg.angle = (int)((float) cfg.angle / 1.8);

302 break;

303 case 1:

304 cfg.angle \*= 10;

305 break;

306 }

307

308 cfg.running = 1;

309 run();

310 length = SPI\_Ethernet\_putConstString(HTTPheader);

311 length += SPI\_Ethernet\_putConstString(HTTPMimeTypeHTML);

312 length += SPI\_Ethernet\_putConstString(OKStatus);

313 return length;

314 }

315

316 // Обработчик Ethernet UDP запросов; Нужен для корректной работы

317 unsigned int SPI\_Ethernet\_UserUDP(unsigned char \*remoteHost,

318 unsigned int remotePort,

319 unsigned int destPort, unsigned int reqLength,

320 TEthPktFlags \*flags) {

321 return (0);

322 }

323

324 // Основная функция

325 void main() {

326 ANSELA = 0x10;

327 OSCCON = 0b0111000;

328 TRISB = 0x00;

329 PORTB = 0b0000;

330 ANSELC = 0;

331

332 SPI1\_Init(); // Инициализация SPI модуля

333 SPI\_Ethernet\_Init(MACAddr, IPAddr, 0x01); // Инициализация Ethernet модуля

334 while (1) {

335 SPI\_Ethernet\_doPacket(); // Обработка следующего пакета

336 }

2 ТЕКСТ ИНТЕРФЕЙСА НА ИСХОДНОМ ЯЗЫКЕ

index.html

1 <html>

2 <style>

3 input {

4 border: 0;

5 cursor: pointer;

6 }

7 button {

8 width: 100%;

9 background: 0;

10 cursor: pointer;

11 }

12 button[id="STOP"], button[id="CLEAR"], button[id="START"] {

13 margin: 2px 2px 2px 2px;

14 width: 25%;

15 }

16 input {

17 background: 0;

18 border-bottom: 1px solid black;

19 margin-bottom: 8px;

20 }

21 input:focus {

22 outline: none;

23 }

24 input[id=speed] {

25 width: 60px;

26 margin-bottom: 2px;

27 text-align: center;

28 }

29 input[id=angle] {

30 width: 60px;

31 margin-bottom: 2px;

32 text-align: center;

33 }

34 table {

35 background-color: #acffff;

36 border-radius: 5px;

37 width: 500;

38 }

39 .active-td {

40 background-color: #52efef;

41 }

42 .running-task {

43 background-color: #f9e772;

44 }

45 .completed-task {

46 background-color: #72f984;

47 }

48 .failed-task {

49 background-color: #f97672;

50 }

51 .remove-button {

52 right: -90px;

53 top: 1px;

54 position: relative;

55 width: 70px;

56 height: 19px;

57 background-color: #f97672;

58 }

59 .run-button {

60 right: -90px;

61 top: 1px;

62 position: relative;

63 width: 70px;

64 height: 19px;

65 background-color: #72f97f;

66 }

67 </style>

68 <script type="text/javascript">

69 var config = {

70 "id": "",

71 "type": "",

72 "direction": "",

73 "speed": "",

74 "angle": ""

75 };

76

77 // Update config field

78 function updateConfig(key, value) {

79 config[key] = value;

80 }

81 // Change task row color to green

82 function completeTask(id) {

83 let intId = parseInt(id)

84 let el = document.querySelectorAll("table[id='tasks'] tbody")[intId].querySelector("tr")

85 el.className = "completed-task"

86 }

87 // Change task row color to red

88 function failTask(id) {

89 let intId = parseInt(id)

90 let el = document.querySelectorAll("table[id='tasks'] tbody")[intId].querySelector("tr")

91 el.className = "failed-task"

92 }

93 // Remove task from table

94 function removeTask(id) {

95 let intId = parseInt(id)

96 document.querySelectorAll("table[id='tasks'] tbody")[intId].remove()

97 let updateEls = document.querySelectorAll("table[id='tasks'] tr")

98 if (updateEls.length == 3) {

99 return;

100 }

101 for (let i = intId + 2; i < document.querySelectorAll("table[id='tasks'] tr").length; i++) {

102 let el = updateEls[i].querySelector("td")

103 el.id = parseInt(el.id) - 1

104 el.querySelector("font").innerText = el.id

105 updateEls[i].querySelectorAll("button")[0].id = el.id

106 updateEls[i].querySelectorAll("button")[1].id = el.id

107 }

108 }

109 // Start tasks

110 function startTasks(){

111 let els = document.querySelectorAll("table[id='tasks'] tbody")

112 runTask("1", els.length)

113 }

114 // Send task to PIC

115 function runTask(id, lastTask = -1) {

116 let intId = parseInt(id)

117 let el = document.querySelectorAll("table[id='tasks'] tbody")[intId].querySelector("tr")

118 el.className = "running-task"

119 let motor = el.querySelectorAll("td")[1].innerText === "Stepper" ? "ST" : "AC";

120 let direction = el.querySelectorAll("td")[2].innerText === "Right" ? "R" : "L";

121 let speed = el.querySelectorAll("td")[3].innerText;

122 let angle = el.querySelectorAll("td")[4].querySelector("font").innerText;

123 let ip = document.querySelector("input[id=ip]").value

124

125 var xhr = new XMLHttpRequest();

126 xhr.open("GET", `http://${ip}:80/${motor},${direction},${speed},${angle}`, true);

127 xhr.onload = function() {

128 console.log(xhr.status, xhr.statusText)

129 if (xhr.status === 200) {

130 completeTask(id)

131 if (lastTask !== -1 && intId < lastTask) {

132 runTask((intId + 1).toString(), lastTask)

133 }

134 } else {

135 failTask(id)

136 }

137 }

138 xhr.send();

139 }

140 // Stop all tasks

141 function stopTasks() {

142 let ip = document.querySelector("input[id=ip]").value

143 fetch(`http://${ip}:80/OFF`, {

144 method: "GET",

145 mode: 'no-cors'

146 });

147 let els = document.querySelectorAll("table[id='tasks'] tbody")

148 for (let i = 1; i < els.length; i++) {

149 els[i].querySelector("tr").className = ""

150 }

151 }

152 // Clear tasks table

153 function clearTasks(){

154 let els = document.querySelectorAll("table[id='tasks'] tbody")

155 for (let i = 1; i < els.length; i++) {

156 els[i].remove()

157 }

158 }

159 // Add new task button

160 function addButtonClick(){

161 config.id = document.querySelectorAll("table[id='tasks'] tr").length - 2

162 config.speed = document.querySelector("input[id=speed]").value

163 config.angle = document.querySelector("input[id=angle]").value

164 document.querySelector("table[id='tasks']").innerHTML += `<tr>

165 <td align=center width="8%" id="${config.id}">

166 <font size=2 color=Black face="verdana">${config.id}</font>

167 </td>

168 <td align=center width="23%">

169 <font size=2 color=Black face="verdana">${config.type}</font>

170 </td>

171 <td align=center width="23%">

172 <font size=2 color=Black face="verdana">${config.direction}</font>

173 </td>

174 <td align=center width="23%">

175 <font size=2 color=Black face="verdana">${config.speed}</font>

176 </td>

177 <td align=center width="23%">

178 <font size=2 color=Black face="verdana" style="position:relative; left: 30%;">${config.angle}</font>

179 <button class="remove-button" type="button" id="${config.id}" onclick="removeTask(this.id)">Remove</button>

180 <button class="run-button" type="button" id="${config.id}" onclick="runTask(this.id)">Run</button>

181 </td>

182 </tr>`

183 }

184 // Handle motor type button click

185 function typeButtonClick(element\_id) {

186 let tButtons = document.querySelectorAll("td.motor-type");

187 for (tb of tButtons) {

188 tb.className = "motor-type";

189 }

190 let naming = document.querySelector("form > table:nth-child(1) > tbody > tr:nth-child(6) > td > font > b")

191 let naming\_sec = document.querySelectorAll("form > table:nth-child(1) > tbody > tr:nth-child(6) > td > font")[1].querySelector("b")

192 if (element\_id === "Async") {

193 naming.innerText = "Speed:"

194 naming\_sec.innerText = ""

195 } else {

196 naming.innerText = "Frequency:"

197 naming\_sec.innerText = "Hz"

198 }

199 document.querySelector(`td.motor-type[id=${element\_id}]`).className = "motor-type active-td";

200 updateConfig("type", element\_id)

201 }

202 // Handle motor direction button click

203 function dirButtonClick(element\_id) {

204 let tButtons = document.querySelectorAll("td.direction");

205 for (tb of tButtons) {

206 tb.className = "direction";

207 }

208 document.querySelector(`td.direction[id=${element\_id}]`).className = "direction active-td";

209 updateConfig("direction", element\_id)

210 }

211 </script>

212 <body>

213 <form name="input" method="get">

214 <table align=center width=500 border=4>

215 <tr>

216 <td align=center colspan=2>

217 <font size=7 color=Black face="verdana"><b>MOTOR CONTROL</b></font>

218 <label for="ip">IP:</label>

219 <input type="text" id="ip" name="ip" required placeholder="10.211.55.5"/>

220 </td>

221 </tr>

222 <tr>

223 <td align=center colspan=2>

224 <font size=2 color=Black face="verdana"><b>Motor Type</b></font>

225 </td>

226 </tr>

227 <tr>

228 <td align=center width="50%" class="motor-type" id="Stepper">

229 <button type="button" id="Stepper" onclick="typeButtonClick(this.id)">Stepper</button>

230 </td>

231 <td align=center width="50%" class="motor-type" id="Async">

232 <button type="button" id="Async" onclick="typeButtonClick(this.id)">Async</button>

233 </td>

234 </tr>

235 <tr>

236 <td align=center colspan=2>

237 <font size=2 color=Black face="verdana"><b>Direction</b></font>

238 </td>

239 </tr>

240 <tr>

241 <td align=center width="50%" class="direction" id="Left">

242 <button type="button" id="Left" onclick="dirButtonClick(this.id)">Left</button>

243 </td>

244 <td align=center width="50%" class="direction" id="Right">

245 <button type="button" id="Right" onclick="dirButtonClick(this.id)">Right</button>

246 </td>

247 </tr>

248 <tr>

249 <td align=center colspan=2>

250 <font size=2 color=Black face="verdana"><b>Frequency:</b></font>

251 <input type="number" id="speed" name="speed" min=10 step=10 required placeholder="10"/>

252 <font size=2 color=Black face="verdana"><b>Hz</b></font>

253 </td>

254 </tr>

255 <tr>

256 <td align=center colspan=2>

257 <font size=2 color=Black face="verdana"><b>Angle:</b></font>

258 <input type="number" id="angle" name="angle" min=0 required placeholder="1"/>

259 <font size=2 color=Black face="verdana"><b>°</b></font>

260 </td>

261 </tr>

262 <tr>

263 <td align=center width="50%" colspan="2">

264 <button type="button" id="ADD" onclick="addButtonClick()">Add</button>

265 </td>

266 </tr>

267 </table>

268 <br>

269 <table align=center width=500 border=4 id="tasks" text-align="center">

270 <tr>

271 <td align=center colspan=5>

272 <font size=5 color=Black face="verdana"><b>Tasks</b></font>

273 </td>

274 </tr>

275 <tr>

276 <td align=center colspan=5>

277 <button type="button" id="START" onclick="startTasks()">Start</button>

278 <button type="button" id="STOP" onclick="stopTasks()">Stop</button>

279 <button type="button" id="CLEAR" onclick="clearTasks()">Clear</button>

280 </td>

281 </tr>

282 <tr>

283 <td align=center width="8%">

284 <font size=2 color=Black face="verdana"><b>ID</b></font>

285 </td>

286 <td align=center width="23%">

287 <font size=2 color=Black face="verdana"><b>Motor</b></font>

288 </td>

289 <td align=center width="23%">

290 <font size=2 color=Black face="verdana"><b>Direction</b></font>

291 </td>

292 <td align=center width="23%">

293 <font size=2 color=Black face="verdana"><b>Speed</b></font>

294 </td>

295 <td align=center width="23%">

296 <font size=2 color=Black face="verdana"><b>Angle</b></font>

297 </td>

298 </tr>

299 </table>

300 </form>

301 </body>

302 </html>