

```
1 void steamGenPID() {
2
3     if (!modbusTCPClient6.connected()) {
4         modbusTCPClient6.begin(serverIOEX6, 502);
5     }
6     // PT304_TWV308_INPUT_PRESSURE_RAW = modbusTCPClient2.holdingRegisterRead(3023); //ref:43017 read Channel 7 PT304
7     PT304_TWV308_INPUT_PRESSURE = map(modbusTCPClient6.inputRegisterRead(6), 0, 65535, 0, 250); // read Channel 6 PT304
8     if (PT304_TWV308_INPUT_PRESSURE >= 170) {
9         BLOWER_SPEED_OFFSET = STEAM_GEN_PID.calculate(170, PT304_TWV308_INPUT_PRESSURE);
10    }
11    if (PT304_TWV308_INPUT_PRESSURE < 170) {
12        BLOWER_SPEED_OFFSET = STEAM_GEN_PID_soft.calculate(170, PT304_TWV308_INPUT_PRESSURE);
13    }
14
15    if (!modbusTCPClient1.connected()) {
16        modbusTCPClient1.begin(serverIOEX1, 502);
17    }
18    modbusTCPClient1.holdingRegisterWrite(40, BLOWER_SPEED_OFFSET);
19
20}
21 void superheatTest() {
22
23     if (!modbusTCPClient3.connected()) {
24         modbusTCPClient3.begin(serverIOEX3, 502);
25     }
26     TT303_HX504_STEAM_OUT = (modbusTCPClient3.holdingRegisterRead(42)) * (0.095); //tt303
27     TT306_EJECTOR_STEAM_IN = (modbusTCPClient3.holdingRegisterRead(43)) * (0.095); //tt306
28
29     if (!modbusTCPClient6.connected()) {
30         modbusTCPClient6.begin(serverIOEX6, 502);
31     }
32     PT318_HX406_OUTPUT_PRESSURE = map(modbusTCPClient6.inputRegisterRead(3), 0, 65535, 0, 250); //read Channel 3
PressureTransducer_318.. 1psi/80counts maximux
33     PT213_RO_PRESSURE = map(modbusTCPClient6.inputRegisterRead(4), 0, 65535, 0, 250); // read Channel 4 PT213
34
35     if ((TT303_HX504_STEAM_OUT - TT306_EJECTOR_STEAM_IN >= 1) && TT303_HX504_STEAM_OUT > 50
36         || TT306_EJECTOR_STEAM_IN > 70 || TT306_EJECTOR_STEAM_IN > 200 || TT303_HX504_STEAM_OUT > 200 ) {
37
38         RO_SPEED_OFFSET = SUPER_HEAT_TT303.calc_reverse(170, TT303_HX504_STEAM_OUT);
39         FCV205_OFFSET = FCV205_PID.calc_reverse(180, TT303_HX504_STEAM_OUT);
40
41         if (!modbusTCPClient1.connected()) {
42             modbusTCPClient1.begin(serverIOEX1, 502);
43         }
44         // modbusTCPClient1.holdingRegisterWrite(41, RO_SPEED_OFFSET); //write channel 1 (WP_Speed)
45         // modbusTCPClient1.holdingRegisterWrite(43, FCV205_OFFSET); //fcv205
46         if (!modbusTCPClient7.connected()) {
47             modbusTCPClient7.begin(serverIOEX7, 502);
48         }
49         modbusTCPClient7.coilWrite(10, ON); //WP_EN..Control is opposite
50     }
51 }
52
53 void rampWaterPump() {
54
55     if (CURRENT_MILLIS-PREVIOUS_MILLIS_8>=10000 && RO_PUMP_FEEDBACK < RO_PUMP_AT_10_GRAMS_PER_SEC && FSM_STATE >= 1 &&
RO_PUMP_COUNT<=1) {
56         RO_PUMP_COUNT += 0.1;
57         PREVIOUS_MILLIS_8=millis();
58         modbusTCPClient1.holdingRegisterWrite(41, RO_PUMP_AT_10_GRAMS_PER_SEC * RO_PUMP_COUNT); //write channel 1
(WP_Speed) made steam at 1*2000//last 3*2000
59     }
60 }
61
62 void connect_IO_Expanders() {
63
64     //CONNECT TO ACROMAGS
65     while (!modbusTCPClient1.connected()) {
66         modbusTCPClient1.begin(serverIOEX1, 502);
67         delay(1);
68     }
69     Serial.write('_'); Serial.write('!'); Serial.println(F("ioex1 "));
70     while (!modbusTCPClient2.connected()) {
71         modbusTCPClient2.begin(serverIOEX2, 502);
```

```
72     delay(1);
73 }
74 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex2  "));
75 while (!modbusTCPClient3.connected()) {
76     modbusTCPClient3.begin(serverIOEX3, 502);
77     delay(1);
78 }
79 Serial.write('_'); Serial.write('!'); Serial.println("ioex3  ");
80 while (!modbusTCPClient4.connected()) {
81     modbusTCPClient4.begin(serverIOEX4, 502);
82     delay(1);
83 }
84 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex4  "));
85 while (!modbusTCPClient5.connected()) {
86     modbusTCPClient5.begin(serverIOEX5, 502);
87     delay(1);
88 }
89 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex5  "));
90 while (!modbusTCPClient6.connected()) {
91     modbusTCPClient6.begin(serverIOEX6, 502);
92     delay(1);
93 }
94 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex6  "));
95 while (!modbusTCPClient7.connected()) {
96     modbusTCPClient7.begin(serverIOEX7, 502);
97     delay(1);
98 }
99 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex7  "));
100 while (!modbusTCPClient8.connected()) {
101     modbusTCPClient8.begin(serverIOEX8, 502);
102     delay(1);
103 }
104 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex8  "));
105 }
106 void readPTs() {
107
108     if (!modbusTCPClient6.connected()) {
109         modbusTCPClient6.begin(serverIOEX6, 502);
110     }
111     //checking PTs
112     modbusTCPClient6.begin(serverIOEX6, 502);
113     PT318_HX406_OUTPUT_PRESSURE = map(modbusTCPClient6.inputRegisterRead(3), 0, 65535, 0, 250); //read Channel 3
PressureTransducer_318.. 1psi/80counts maximux
114     PT213_RO_PRESSURE = map(modbusTCPClient6.inputRegisterRead(4), 0, 65535, 0, 250); // read Channel 4 PT213
115     PT420_STEAM_EJECTOR_PRESSURE = map( modbusTCPClient6.inputRegisterRead(5), 0, 65535, 0, 250); // read Channel 5
PT420
116     PT304_TWV308_INPUT_PRESSURE = map(modbusTCPClient6.inputRegisterRead(6), 0, 65535, 0, 250); // read Channel 6 PT304
117
118     txGUI[28] = PT318_HX406_OUTPUT_PRESSURE;
119     txGUI[29] = PT213_RO_PRESSURE;
120     txGUI[30] = PT420_STEAM_EJECTOR_PRESSURE;
121     txGUI[31] = PT304_TWV308_INPUT_PRESSURE ;
122
123     if (PT318_HX406_OUTPUT_PRESSURE >= 240) {
124         FSM_STATE = INITIALIZE;
125         ERROR = 3;
126     }
127     else {
128         ERROR = 0;
129     }
130     if ( PT213_RO_PRESSURE >= 240) {
131         FSM_STATE = INITIALIZE;
132         ERROR = 4;
133     }
134     else {
135         ERROR = 0;
136     }
137     if (PT420_STEAM_EJECTOR_PRESSURE >= 240) {
138         FSM_STATE = INITIALIZE;
139         ERROR = 5;
140     }
141     else {
142         ERROR = 0;
143     }
```

```
144 if (PT304_TWV308_INPUT_PRESSURE >= 240) {  
145     FSM_STATE = INITIALIZE;  
146     ERROR = 6;  
147 }  
148 else {  
149     ERROR = 0;  
150 }  
151  
152 }  
153 void readTCs() {  
154  
155     if (!modbusTCPClient3.connected()) {  
156         modbusTCPClient3.begin(serverIOEX3, 502);  
157     }  
158     if (!modbusTCPClient4.connected()) {  
159         modbusTCPClient4.begin(serverIOEX4, 502);  
160     }  
161     if (!modbusTCPClient5.connected()) {  
162         modbusTCPClient5.begin(serverIOEX5, 502);  
163     }  
164  
165     //read acromag 4  
166     TT142_SR_FUEL = modbusTCPClient3.holdingRegisterRead(40) * (0.095);  
167     TT301_HX406_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(41) * (0.095);  
168     TT303_HX504_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(42) * (0.095);  
169     TT306_EJECTOR_STEAM_IN = modbusTCPClient3.holdingRegisterRead(43) * (0.095);  
170     TT313_HX402_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(44) * (0.095);  
171     TT319_HX402_STEAM_SYSTEM = modbusTCPClient3.holdingRegisterRead(45) * (0.095);  
172     TT407_STEAM_REFORMER_OUT_LREF = modbusTCPClient3.holdingRegisterRead(46) * (0.095);  
173     TT513_HX504_IN = modbusTCPClient3.holdingRegisterRead(47) * (0.095);  
174  
175     TT410HTS_OUT_LREF = modbusTCPClient4.holdingRegisterRead(40) * (0.095);  
176     TT441_SMR_TUBE1_OUT = modbusTCPClient4.holdingRegisterRead(41) * (0.095);  
177     TT442_SMR_TUBE2_OUT = modbusTCPClient4.holdingRegisterRead(42) * (0.095);  
178  
179     TT512_SILICON_CARBIDE_OUT = modbusTCPClient4.holdingRegisterRead(44) * (0.095);  
180     TT408HTS_IN_LREF = modbusTCPClient4.holdingRegisterRead(45) * (0.095);  
181     TT514_HX504_OUT = modbusTCPClient4.holdingRegisterRead(46) * (0.095);  
182     TT411_FPZ_OUT_LREF = modbusTCPClient4.holdingRegisterRead(47) * (0.095);  
183  
184     TT430_SMR_TUBES_INLET = modbusTCPClient5.holdingRegisterRead(40) * (0.095);  
185     TT511_SILICON_CARBIDE_OUT = modbusTCPClient5.holdingRegisterRead(41) * (0.095);  
186     TT444_SMR_TUBE4_OUT = modbusTCPClient5.holdingRegisterRead(42) * (0.095);  
187     TT445_SMR_TUBE5_OUT = modbusTCPClient5.holdingRegisterRead(43) * (0.095);  
188     TT446_SMR_TUBE6_OUT = modbusTCPClient5.holdingRegisterRead(44) * (0.095);  
189     TT447_SMR_TUBE7_OUT = modbusTCPClient5.holdingRegisterRead(45) * (0.095);  
190     TT448_SMR_TUBE8_OUT = modbusTCPClient5.holdingRegisterRead(46) * (0.095);  
191     TT449_SMR_TUBE9_OUT = modbusTCPClient5.holdingRegisterRead(47) * (0.095);  
192  
193     txGUI[0] = TT142_SR_FUEL;  
194     txGUI[1] = TT301_HX406_STEAM_OUT;  
195     txGUI[2] = TT303_HX504_STEAM_OUT;  
196     txGUI[3] = TT306_EJECTOR_STEAM_IN;  
197     txGUI[4] = TT313_HX402_STEAM_OUT;  
198     txGUI[5] = TT319_HX402_STEAM_SYSTEM;  
199     txGUI[6] = TT407_STEAM_REFORMER_OUT_LREF;  
200     txGUI[7] = TT408HTS_IN_LREF;  
201     txGUI[8] = TT410HTS_OUT_LREF;  
202     txGUI[9] = TT411_FPZ_OUT_LREF;  
203     txGUI[10] = TT430_SMR_TUBES_INLET;  
204     txGUI[11] = TT511_SILICON_CARBIDE_OUT;  
205     txGUI[12] = TT512_SILICON_CARBIDE_OUT;  
206     txGUI[13] = TT513_HX504_IN;  
207     txGUI[14] = TT514_HX504_OUT;  
208     txGUI[15] = TT441_SMR_TUBE1_OUT;  
209     txGUI[16] = TT442_SMR_TUBE2_OUT;  
210     txGUI[17] = TT443_SMR_TUBE3_OUT;  
211     txGUI[18] = TT444_SMR_TUBE4_OUT;  
212     txGUI[19] = TT445_SMR_TUBE5_OUT;  
213     txGUI[20] = TT446_SMR_TUBE6_OUT;  
214     txGUI[21] = TT447_SMR_TUBE7_OUT;  
215     txGUI[22] = TT448_SMR_TUBE8_OUT;  
216     txGUI[23] = TT449_SMR_TUBE9_OUT;  
217     txGUI[24] = BLOWER_SPEED_FEEDBACK;
```

```
218 txGUI[25] = RO_PUMP_FEEDBACK;
219 txGUI[27] = FT132_NG_FEED_FLOW;
220 txGUI[34] = FCV134_BURNER_FUEL_FLOW_FB;
221
222 /* switch (TC_CHECK_COUNTER) {
223     case 1:
224         if (CURRENT_MILLIS - PREVIOUS_MILLIS_3 >= 1000) {
225
226             TT142_SR_FUEL = modbusTCPClient3.holdingRegisterRead(40) * (0.095);
227             TT301_HX406_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(41) * (0.095);
228             TT303_HX504_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(42) * (0.095);
229             TT306_EJECTOR_STEAM_IN = modbusTCPClient3.holdingRegisterRead(43) * (0.095);
230             TT313_HX402_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(44) * (0.095);
231             TT319_HX402_STEAM_SYSTEM = modbusTCPClient3.holdingRegisterRead(45) * (0.095);
232             TT407_STEAM_REFORMER_OUT_LREF = modbusTCPClient3.holdingRegisterRead(46) * (0.095);
233             TT408HTS_IN_LREF = modbusTCPClient3.holdingRegisterRead(47) * (0.095);
234
235             txGUI[0] = TT142_SR_FUEL;
236             txGUI[1] = TT301_HX406_STEAM_OUT;
237             txGUI[2] = TT303_HX504_STEAM_OUT;
238             txGUI[3] = TT306_EJECTOR_STEAM_IN;
239             txGUI[4] = TT313_HX402_STEAM_OUT;
240             txGUI[5] = TT319_HX402_STEAM_SYSTEM;
241             txGUI[6] = TT407_STEAM_REFORMER_OUT_LREF;
242             txGUI[7] = TT408HTS_IN_LREF;
243
244         if (TT301_HX406_STEAM_OUT > 1000) {
245             //Serial.println("TT301_HX406_STEAM_OUT > 1000C");
246             FSM_STATE = INITIALIZE;
247             ERROR = 13;
248         }
249         if (TT303_HX504_STEAM_OUT > 1000) {
250             // Serial.println("TT303_HX504_STEAM_OUT > 1000C");
251             FSM_STATE = INITIALIZE;
252             ERROR = 14;
253         }
254         if (TT306_EJECTOR_STEAM_IN > 1000) {
255             // Serial.println("TT306_EJECTOR_STEAM_IN > 1000C");
256             FSM_STATE = INITIALIZE;
257             ERROR = 15;
258         }
259         if (TT313_HX402_STEAM_OUT > 1000) {
260             // Serial.println("TT313_HX402_STEAM_OUT > 1000C");
261             FSM_STATE = INITIALIZE;
262             ERROR = 16;
263         }
264         if (TT319_HX402_STEAM_SYSTEM > 1000) {
265             // Serial.println("TT319_HX402_STEAM_SYSTEM > 1000C");
266             FSM_STATE = INITIALIZE;
267             ERROR = 17;
268         }
269         if (TT407_STEAM_REFORMER_OUT_LREF > 1000) {
270             // Serial.println("TT407_STEAM_REFORMER_OUT_LREF > 1000C");
271             // FSM_STATE = INITIALIZE;
272             // ERROR = 18;
273         }
274         if (TT408HTS_IN_LREF > 1000) {
275             // Serial.println("TT408HTS_IN_LREF > 1000C");
276             FSM_STATE = INITIALIZE;
277             ERROR = 19;
278         }
279
280
281         Serial.print("TT142_SR_FUEL: ");Serial.println(TT142_SR_FUEL);
282         Serial.print("TT301_HX406_STEAM_OUT : ");Serial.println(TT301_HX406_STEAM_OUT);
283         Serial.print("TT303_HX504_STEAM_OUT : ");Serial.println(TT303_HX504_STEAM_OUT);
284         Serial.print("TT306_EJECTOR_STEAM_IN : ");Serial.println(TT306_EJECTOR_STEAM_IN);
285         Serial.print("TT313_HX402_STEAM_OUT : ");Serial.println(TT313_HX402_STEAM_OUT);
286         Serial.print("TT319_HX402_STEAM_SYSTEM : ");Serial.println(TT319_HX402_STEAM_SYSTEM);
287         Serial.print("TT407_STEAM_REFORMER_OUT_LREF : ");Serial.println(TT407_STEAM_REFORMER_OUT_LREF);
288         Serial.print("TT408HTS_IN_LREF : ");Serial.println(TT408HTS_IN_LREF);
289
290
291         PREVIOUS_MILLIS_3 = millis();
```

```
292     TC_CHECK_COUNTER = 2;
293     break;
294 }
295
296 case 2:
297 //read acromag5
298 if (CURRENT_MILLIS - PREVIOUS_MILLIS_3 >= 1000) {
299
300     TT410HTS_OUT_LREF = modbusTCPClient4.holdingRegisterRead(40) * (0.095);
301     TT441_SMR_TUBE1_OUT = modbusTCPClient4.holdingRegisterRead(41) * (0.095);
302     TT442_SMR_TUBE2_OUT = modbusTCPClient4.holdingRegisterRead(42) * (0.095);
303     TT511_SILICON_CARBIDE_OUT = modbusTCPClient4.holdingRegisterRead(43) * (0.095);
304     TT512_SILICON_CARBIDE_OUT = modbusTCPClient4.holdingRegisterRead(44) * (0.095);
305     TT513_HX504_IN = modbusTCPClient4.holdingRegisterRead(45) * (0.095);
306     TT514_HX504_OUT = modbusTCPClient4.holdingRegisterRead(46) * (0.095);
307     TT411_FPZ_OUT_LREF = modbusTCPClient4.holdingRegisterRead(47) * (0.095);
308
309     txGUI[8] = TT410HTS_OUT_LREF;
310     txGUI[9] = TT411_FPZ_OUT_LREF;
311     txGUI[10] = TT430_SMR_TUBES_INLET;
312     txGUI[11] = TT511_SILICON_CARBIDE_OUT;
313     txGUI[12] = TT512_SILICON_CARBIDE_OUT;
314     txGUI[13] = TT513_HX504_IN;
315     txGUI[14] = TT514_HX504_OUT;
316     txGUI[15] = TT441_SMR_TUBE1_OUT;
317
318     //    if (TT410HTS_OUT_LREF > 1000) {
319     //        Serial.println("TT410HTS_OUT_LREF > 1000C");
320     //        FSM_STATE = INITIALIZE;
321     //        ERROR = 20;
322     //}
323     if (TT411_FPZ_OUT_LREF > 1000) {
324         //Serial.println("TT411_FPZ_OUT_LREF > 1000C");
325         FSM_STATE = INITIALIZE;
326         ERROR = 21;
327     }
328     if (TT430_SMR_TUBES_INLET > 1000) {
329         // Serial.println("TT430_SMR_TUBES_INLET > 1000C");
330         FSM_STATE = INITIALIZE;
331         ERROR = 22;
332     }
333     if (TT511_SILICON_CARBIDE_OUT > 1000) {
334         // Serial.println("TT511_SILICON_CARBIDE_OUT > 1000C");
335         FSM_STATE = INITIALIZE;
336         ERROR = 23;
337     }
338     //    if (TT512_SILICON_CARBIDE_OUT > 1000) {
339     //        Serial.println("TT512_SILICON_CARBIDE_OUT > 1000C");
340     //        FSM_STATE = INITIALIZE;
341     //        ERROR = 24;
342     //}
343     if (TT513_HX504_IN > 1000) {
344         // Serial.println("TT513_HX504_IN > 1000C");
345         FSM_STATE = INITIALIZE;
346         ERROR = 25;
347     }
348     if (TT514_HX504_OUT > 1000) {
349         // Serial.println("TT514_HX504_OUT > 1000C");
350         FSM_STATE = INITIALIZE;
351         ERROR = 26;
352     }
353     if (TT441_SMR_TUBE1_OUT > 800) {
354         // Serial.println("TT441_SMR_TUBE1_OUT > 1000C");
355         FSM_STATE = INITIALIZE;
356         ERROR = 27;
357     }
358
359
360     Serial.print("TT410HTS_OUT_LREF : ");Serial.println(TT410HTS_OUT_LREF);
361     Serial.print("TT411_FPZ_OUT_LREF : ");Serial.println(TT411_FPZ_OUT_LREF);
362     Serial.print("TT430_SMR_TUBES_INLET : ");Serial.println(TT430_SMR_TUBES_INLET);
363     Serial.print("TT511_SILICON_CARBIDE_OUT : ");Serial.println(TT511_SILICON_CARBIDE_OUT);
364     Serial.print("TT512_SILICON_CARBIDE_OUT : ");Serial.println(TT512_SILICON_CARBIDE_OUT);
365     Serial.print("TT513_HX504_IN : ");Serial.println(TT513_HX504_IN);
```

```
366     Serial.print("TT514_HX504_OUT : ");Serial.println(TT514_HX504_OUT);
367     Serial.print("TT441_SMR_TUBE1_OUT : ");Serial.println(TT441_SMR_TUBE1_OUT);
368
369
370     TC_CHECK_COUNTER = 3;
371     PREVIOUS_MILLIS_3 = millis();
372     break;
373 }
374
375 case 3:
376     if (CURRENT_MILLIS - PREVIOUS_MILLIS_3 >= 1000) {
377
378         TT430_SMR_TUBES_INLET = modbusTCPClient5.holdingRegisterRead(40) * (0.095);
379         TT443_SMR_TUBE3_OUT = modbusTCPClient5.holdingRegisterRead(41) * (0.095);
380         TT444_SMR_TUBE4_OUT = modbusTCPClient5.holdingRegisterRead(42) * (0.095);
381         TT445_SMR_TUBE5_OUT = modbusTCPClient5.holdingRegisterRead(43) * (0.095);
382         TT446_SMR_TUBE6_OUT = modbusTCPClient5.holdingRegisterRead(44) * (0.095);
383         TT447_SMR_TUBE7_OUT = modbusTCPClient5.holdingRegisterRead(45) * (0.095);
384         TT448_SMR_TUBE8_OUT = modbusTCPClient5.holdingRegisterRead(46) * (0.095);
385         TT449_SMR_TUBE9_OUT = modbusTCPClient5.holdingRegisterRead(47) * (0.095);
386
387         txGUI[16] = TT442_SMR_TUBE2_OUT;
388
389         txGUI[17] = TT443_SMR_TUBE3_OUT;
390         txGUI[18] = TT444_SMR_TUBE4_OUT;
391         txGUI[19] = TT445_SMR_TUBE5_OUT;
392         txGUI[20] = TT446_SMR_TUBE6_OUT;
393         txGUI[21] = TT447_SMR_TUBE7_OUT;
394         txGUI[22] = TT448_SMR_TUBE8_OUT;
395         txGUI[23] = TT449_SMR_TUBE9_OUT;
396
397         if (TT442_SMR_TUBE2_OUT > 800) {
398             // Serial.println("TT442_SMR_TUBE2_OUT > 1000C");
399             FSM_STATE = INITIALIZE;
400             ERROR = 28;
401         }
402         if (TT443_SMR_TUBE3_OUT > 800) {
403             // Serial.println("TT443_SMR_TUBE3_OUT > 1000C");
404             FSM_STATE = INITIALIZE;
405             ERROR = 29;
406         }
407         if (TT444_SMR_TUBE4_OUT > 800) {
408             // Serial.println("TT444_SMR_TUBE4_OUT > 700C");
409             FSM_STATE = INITIALIZE;
410             ERROR = 30;
411         }
412         if (TT445_SMR_TUBE5_OUT > 800) {
413             // Serial.println("TT445_SMR_TUBE5_OUT > 700C");
414             FSM_STATE = INITIALIZE;
415             ERROR = 31;
416         }
417         if (TT446_SMR_TUBE6_OUT > 800) {
418             // Serial.println("TT446_SMR_TUBE6_OUT > 700C");
419             FSM_STATE = INITIALIZE;
420             ERROR = 32;
421         }
422         if (TT447_SMR_TUBE7_OUT > 800) {
423             // Serial.println("TT447_SMR_TUBE7_OUT > 700C");
424             FSM_STATE = INITIALIZE;
425             ERROR = 33;
426         }
427         if (TT448_SMR_TUBE8_OUT > 800) {
428             // Serial.println("TT448_SMR_TUBE8_OUT > 700C");
429             FSM_STATE = INITIALIZE;
430             ERROR = 34;
431         }
432         if (TT449_SMR_TUBE9_OUT > 800) {
433             // Serial.println("TT449_SMR_TUBE9_OUT > 700C");
434             FSM_STATE = INITIALIZE;
435             ERROR = 35;
436         }
437
438
439     Serial.print("TT442_SMR_TUBE2_OUT : ");Serial.println(TT442_SMR_TUBE2_OUT);
```

```
440     Serial.print("TT443_SMR_TUBE3_OUT : ");Serial.println(TT443_SMR_TUBE3_OUT);
441     Serial.print("TT444_SMR_TUBE4_OUT : ");Serial.println(TT444_SMR_TUBE4_OUT);
442     Serial.print("TT445_SMR_TUBE5_OUT : ");Serial.println(TT445_SMR_TUBE5_OUT);
443     Serial.print("TT446_SMR_TUBE6_OUT : ");Serial.println(TT446_SMR_TUBE6_OUT);
444     Serial.print("TT447_SMR_TUBE7_OUT : ");Serial.println(TT447_SMR_TUBE7_OUT);
445     Serial.print("TT448_SMR_TUBE8_OUT : ");Serial.println(TT448_SMR_TUBE8_OUT);
446     Serial.print("TT449_SMR_TUBE9_OUT : ");Serial.println(TT449_SMR_TUBE9_OUT);
447
448
449     TC_CHECK_COUNTER = 1;
450     PREVIOUS_MILLIS_3 = millis();
451     DB_TX();
452     break;
453 }
454
455 */
456 }
457 void readOut() {
458     lcd.setBacklight(HIGH);
459     switch (READOUT_COUNTER) {
460
461     case 1:
462         if (CURRENT_MILLIS - PREVIOUS_MILLIS_4 >= 3000) {
463             lcd.clear();
464             lcd.setCursor(0, 0);
465             lcd.print("P1.");
466             lcd.print((int)PT304_TWV308_INPUT_PRESSURE);//nine tubes
467             lcd.setCursor(7, 0);
468             lcd.print("P2.");
469             lcd.print((int)PT318_HX406_OUTPUT_PRESSURE);//nine tubes
470             lcd.setCursor(14, 0);
471             lcd.print(FSM_STATE);
472             lcd.setCursor(0, 1);
473             lcd.print("P3.");
474             lcd.print((int)PT420_STEAM_EJECTOR_PRESSURE);//nine tubes
475             lcd.setCursor(7, 1);
476             lcd.print("P4.");
477             lcd.print((int)PT213_R0_PRESSURE);//nine tubes
478             READOUT_COUNTER = 2;
479             PREVIOUS_MILLIS_4 = millis();
480             break;
481         }
482     case 2:
483         if (CURRENT_MILLIS - PREVIOUS_MILLIS_4 >= 3000) {
484             lcd.clear();
485             lcd.setCursor(0, 0);
486             lcd.print("S9.");
487             lcd.print((int)TT449_SMR_TUBE9_OUT);//nine tubes
488             lcd.setCursor(7, 0);
489             lcd.print("H1.");
490             lcd.print((int)TT511_SILICON_CARBIDE_OUT);//nine tubes
491             lcd.setCursor(14, 0);
492             lcd.print(FSM_STATE);
493             lcd.setCursor(0, 1);
494             lcd.print("H2.");
495             lcd.print((int)TT513_HX504_IN);//nine tubes
496             lcd.setCursor(7, 1);
497             lcd.print("SH.");
498             lcd.print((int)TT301_HX406_STEAM_OUT);//nine tubes
499             READOUT_COUNTER = 3;
500             PREVIOUS_MILLIS_4 = millis();
501             break;
502         }
503     case 3:
504         if (CURRENT_MILLIS - PREVIOUS_MILLIS_4 >= 3000) {
505             lcd.begin(16, 2);
506             lcd.clear();
507             lcd.setCursor(0, 0);
508             lcd.print("S1.");
509             lcd.print((int)TT441_SMR_TUBE1_OUT);//nine tubes
510             lcd.setCursor(7, 0);
511             lcd.print("S2.");
512             lcd.print((int)TT442_SMR_TUBE2_OUT);//nine tubes
513             lcd.setCursor(14, 0);
```

```
514     lcd.print(FSM_STATE);
515     lcd.setCursor(0, 1);
516     lcd.print("S3.");
517     lcd.print((int)TT443_SMR_TUBE3_OUT);//nine tubes
518     lcd.setCursor(7, 1);
519     lcd.print("S4.");
520     lcd.print((int)TT444_SMR_TUBE4_OUT);//nine tubes
521     READOUT_COUNTER = 4;
522     PREVIOUS_MILLIS_4 = millis();
523     break;
524 }
525 case 4:
526 if (CURRENT_MILLIS - PREVIOUS_MILLIS_4 >= 3000) {
527     lcd.clear();
528     lcd.setCursor(0, 0);
529     lcd.print("S5.");
530     lcd.print((int)TT445_SMR_TUBE5_OUT);//nine tubes
531     lcd.setCursor(7, 0);
532     lcd.print("S6.");
533     lcd.print((int)TT446_SMR_TUBE6_OUT);//nine tubes
534     lcd.setCursor(14, 0);
535     lcd.print(FSM_STATE);
536     lcd.setCursor(0, 1);
537     lcd.print("S7.");
538     lcd.print((int)TT447_SMR_TUBE7_OUT);//nine tubes
539     lcd.setCursor(7, 1);
540     lcd.print("S8.");
541     lcd.print((int)TT448_SMR_TUBE8_OUT);//nine tubes
542     READOUT_COUNTER = 1;
543     PREVIOUS_MILLIS_4 = millis();
544     break;
545 }
546 }
547 }
548 }
549 void integrityCheck() {
550
551
552 if (!SENSOR_INTEGRITY_CHECK) {
553
554
555 while (!modbusTCPClient1.connected()) {
556     modbusTCPClient1.begin(serverIOEX1, 502);
557     // Serial.println("CONNECTING IOEX1.");
558 }
559 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex1~ "));
560 if (!modbusTCPClient2.connected()) {
561     modbusTCPClient2.begin(serverIOEX2, 502);
562     // Serial.println("CONNECTING IOEX2.");
563 }
564 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex2~ "));
565 if (!modbusTCPClient3.connected()) {
566     modbusTCPClient3.begin(serverIOEX3, 502);
567     // Serial.println("CONNECTING IOEX3.");
568 }
569 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex3~ "));
570 if (!modbusTCPClient4.connected()) {
571     modbusTCPClient4.begin(serverIOEX4, 502);
572     // Serial.println("CONNECTING IOEX4.");
573 }
574 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex4~ "));
575 if (!modbusTCPClient5.connected()) {
576     modbusTCPClient5.begin(serverIOEX5, 502);
577     // Serial.println("CONNECTING IOEX5.");
578 }
579 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex5~ "));
580 if (!modbusTCPClient7.connected()) {
581     modbusTCPClient7.begin(serverIOEX7, 502);
582     // Serial.println("CONNECTING IOEX7.");
583 }
584 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex7~ "));
585 if (!modbusTCPClient8.connected()) {
586     modbusTCPClient8.begin(serverIOEX8, 502);
587     // Serial.println("CONNECTING IOEX8.");
```

```
588 }
589 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex8~ "));
590 if (!modbusTCPClient6.connected()) {
591     modbusTCPClient6.begin(serverIOEX6, 502);
592     // Serial.println("CONNECTING IOEX6.");
593 }
594 Serial.write('_'); Serial.write('!'); Serial.println(F("ioex6~ "));
595
596 PT318_HX406_OUTPUT_PRESSURE = map(modbusTCPClient6.inputRegisterRead(3), 0, 65535, 0, 250); // read Channel 3
PressureTransducer_318.. 1psi/80counts maximum
597 PT213_RO_PRESSURE = map( modbusTCPClient6.inputRegisterRead(4), 0, 65535, 0, 250 ); // read Channel 4 PT213
598 PT420_STEAM_EJECTOR_PRESSURE = map(modbusTCPClient6.inputRegisterRead(5), 0, 65535, 0, 250 ); //read Channel 5
PT420
599 PT304_TWV308_INPUT_PRESSURE = map(modbusTCPClient6.inputRegisterRead(6), 0, 65535, 0, 250); // read Channel 6
PT304
600
601 if (PT318_HX406_OUTPUT_PRESSURE >= 240) {
602     FSM_STATE = INITIALIZE;
603     ERROR = 3;
604 }
605 else {
606     ERROR = 0;
607 }
608 if (PT213_RO_PRESSURE >= 240) {
609     FSM_STATE = INITIALIZE;
610     ERROR = 4;
611 }
612 else {
613     ERROR = 0;
614 }
615 if (PT420_STEAM_EJECTOR_PRESSURE >= 240) {
616     FSM_STATE = INITIALIZE;
617     ERROR = 5;
618 }
619 else {
620     ERROR = 0;
621 }
622 if (PT304_TWV308_INPUT_PRESSURE >= 240) {
623     FSM_STATE = INITIALIZE;
624     ERROR = 6;
625 }
626 else {
627     ERROR = 0;
628 }
629
630 TT142_SR_FUEL = modbusTCPClient3.holdingRegisterRead(40) * (0.095);
631 TT301_HX406_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(41) * (0.095);
632 TT303_HX504_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(42) * (0.095);
633 TT306_EJECTOR_STEAM_IN = modbusTCPClient3.holdingRegisterRead(43) * (0.095);
634 TT313_HX402_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(44) * (0.095);
635 TT319_HX402_STEAM_SYSTEM = modbusTCPClient3.holdingRegisterRead(45) * (0.095);
636 TT407_STEAM_REFORMER_OUT_LREF = modbusTCPClient3.holdingRegisterRead(46) * (0.095);
637 TT513_HX504_IN = modbusTCPClient3.holdingRegisterRead(47) * (0.095);
638
639 TT410HTS_OUT_LREF = modbusTCPClient4.holdingRegisterRead(40) * (0.095);
640 TT411FPZ_OUT_LREF = modbusTCPClient4.holdingRegisterRead(41) * (0.095);
641 TT430_SMR_TUBES_INLET = modbusTCPClient4.holdingRegisterRead(42) * (0.095);
642 TT443_SMR_TUBE3_OUT = modbusTCPClient4.holdingRegisterRead(43) * (0.095);
643 TT512_SILICON_CARBIDE_OUT = modbusTCPClient4.holdingRegisterRead(44) * (0.095);
644 TT408HTS_IN_LREF = modbusTCPClient4.holdingRegisterRead(45) * (0.095);
645 TT514_HX504_OUT = modbusTCPClient4.holdingRegisterRead(46) * (0.095);
646 TT441_SMR_TUBE1_OUT = modbusTCPClient4.holdingRegisterRead(47) * (0.095);
647
648 TT442_SMR_TUBE2_OUT = modbusTCPClient5.holdingRegisterRead(40) * (0.095);
649 TT511_SILICON_CARBIDE_OUT = modbusTCPClient5.holdingRegisterRead(41) * (0.095);
650 TT444_SMR_TUBE4_OUT = modbusTCPClient5.holdingRegisterRead(42) * (0.095);
651 TT445_SMR_TUBE5_OUT = modbusTCPClient5.holdingRegisterRead(43) * (0.095);
652 TT446_SMR_TUBE6_OUT = modbusTCPClient5.holdingRegisterRead(44) * (0.095);
653 TT447_SMR_TUBE7_OUT = modbusTCPClient5.holdingRegisterRead(45) * (0.095);
654 TT448_SMR_TUBE8_OUT = modbusTCPClient5.holdingRegisterRead(46) * (0.095);
655 TT449_SMR_TUBE9_OUT = modbusTCPClient5.holdingRegisterRead(47) * (0.095);
656
657
658
```

```
659 txGUI[0] = TT142_SR_FUEL;
660 txGUI[1] = TT301_HX406_STEAM_OUT;
661 txGUI[2] = TT303_HX504_STEAM_OUT;
662 txGUI[3] = TT306_EJECTOR_STEAM_IN;
663 txGUI[4] = TT313_HX402_STEAM_OUT;
664 txGUI[5] = TT319_HX402_STEAM_SYSTEM;
665 txGUI[6] = TT407_STEAM_REFORMER_OUT_LREF;
666 txGUI[7] = TT408HTS_IN_LREF;
667 txGUI[8] = TT410HTS_OUT_LREF;
668 txGUI[9] = TT411FPZ_OUT_LREF;
669 txGUI[10] = TT430_SMR_TUBES_INLET;
670 txGUI[11] = TT511_SILICON_CARBIDE_OUT;
671 txGUI[12] = TT512_SILICON_CARBIDE_OUT;
672 txGUI[13] = TT513_HX504_IN;
673 txGUI[14] = TT514_HX504_OUT;
674 txGUI[15] = TT441_SMR_TUBE1_OUT;
675 txGUI[16] = TT442_SMR_TUBE2_OUT;
676 txGUI[17] = TT443_SMR_TUBE3_OUT;
677 txGUI[18] = TT444_SMR_TUBE4_OUT;
678 txGUI[19] = TT445_SMR_TUBE5_OUT;
679 txGUI[20] = TT446_SMR_TUBE6_OUT;
680 txGUI[21] = TT447_SMR_TUBE7_OUT;
681 txGUI[22] = TT448_SMR_TUBE8_OUT;
682 txGUI[23] = TT449_SMR_TUBE9_OUT;
683 txGUI[24] = BLOWER_SPEED_FEEDBACK;
684 txGUI[25] = RO_PUMP_FEEDBACK;
685 txGUI[27] = FT132_NG_FEED_FLOW;
686 txGUI[34] = FCV134_BURNER_FUEL_FLOW_FB;
687
688 //analog outputs
689 modbusTCPClient1.holdingRegisterWrite(40, OFF); //write channel 0 (BLWRSpeed)
690 modbusTCPClient1.holdingRegisterWrite(41, OFF); //write channel 1 (WP_Speed)
691 modbusTCPClient1.holdingRegisterWrite(42, OFF); //write channel 2 (FCV134)
692 modbusTCPClient1.holdingRegisterWrite(43, OFF); //write channel 3 (FCV205)
693 modbusTCPClient2.holdingRegisterWrite(40, OFF); //write channel 0 (FCV141)
694
695 //digital outputs
696 modbusTCPClient7.coilWrite(5, OFF); //XV801
697 modbusTCPClient7.coilWrite(9, 1); //BLWR_EN ON bc opposite
698 modbusTCPClient7.coilWrite(10, 1); //WP_EN ON bc opposite
699 modbusTCPClient7.coilWrite(6, OFF); //TWV308
700 modbusTCPClient7.coilWrite(4, OFF); //XV1100
701 modbusTCPClient7.coilWrite(3, OFF); //XV501
702 modbusTCPClient7.coilWrite(8, OFF); //BMM_CR2
703 modbusTCPClient7.coilWrite(7, OFF); //TWV901
704 modbusTCPClient7.coilWrite(2, OFF); //xv909
705
706 FCV134_BURNER_FUEL_FLOW_FB = map(modbusTCPClient6.inputRegisterRead(7), 100, 65535, 0, 100); // read Channel 7
707 FCV134
708 if (!(FCV134_BURNER_FUEL_FLOW_FB < 1)) {
709     ERROR = 2;
710     FSM_STATE = INITIALIZE;
711     SENSOR_INTEGRITY_CHECK = false;
712 }
713 else {
714     SENSOR_INTEGRITY_CHECK = true;
715 }
716 modbusTCPClient1.holdingRegisterWrite(40, 0); //write channel 0 (BLWRSpeed) off
717
718 //get that dang dynamic pressure switch off!!
719 BLOWER_SPEED_FEEDBACK = map(modbusTCPClient6.holdingRegisterRead(0), 0, 65535, 0, 60); //10volts/10000counts
720 while (BLOWER_SPEED_FEEDBACK > 1) {
721     modbusTCPClient1.holdingRegisterWrite(40, 0); //write channel 0 (BLWRSpeed) off
722     BLOWER_SPEED_FEEDBACK = map(modbusTCPClient6.holdingRegisterRead(0), 0, 65535, 0, 60); //10volts/20000counts
723     delay(1);
724 }
725 }
726
727 void readOCI() {
728     // Serial.println("Reading OCI417.");
729     OCI_RESULT = NODE.readInputRegisters(OCI_INPUT_STATUS_REGISTER, 2); //address,qty
730     //do something with data if read is successful
731     if (OCI_RESULT == NODE.ku8MBSuccess)
```

```
732 {
733     OCI_INPUT_STATUS_WORD = NODE.getResponseBuffer(0);
734     // Serial.print("OCI INPUT WORD: ");Serial.println(OCI_INPUT_STATUS_WORD,BIN);
735
736     COMBUSTION_PRESSURE_SWITCH = bitRead(OCI_INPUT_STATUS_WORD, 7); //COMBUSTION AIR SW
737
738     OCI_OUTPUT_STATUS_WORD = NODE.getResponseBuffer(1);
739     //Serial.print("OCI OUTPUT WORD: ");Serial.println(OCI_OUTPUT_STATUS_WORD,BIN);
740     BMM_PROOF_OF_FLAME = bitRead(OCI_OUTPUT_STATUS_WORD, 0);
741     if (FSM_STATE > BURNER_RAMP && BMM_PROOF_OF_FLAME == false) {
742         FSM_STATE = INITIALIZE;
743         ERROR = 10;
744     }
745     BMM_ALARM_STATUS = bitRead(OCI_OUTPUT_STATUS_WORD, 1);
746     if (FSM_STATE > BMM_IGNITION && BMM_ALARM_STATUS == true) {
747         FSM_STATE = INITIALIZE;
748         ERROR = 10;
749     }
750     //OCI_TO_BMM_COM=bitRead(OCI_OUTPUT_STATUS_WORD,2);
751     // if(FSM_STATE>6 && OCI_TO_BMM_COM==true){FSM_STATE=0;ERROR=10;}
752 }
753 }
754 void readBtn() {
755     //read if there has been a change for the inputs
756     //function for reading buttons and signaling indicators
757
758     //check status of inputs xor last di status woth current di status
759     //if (!modbusTCPClient8.connected()) {
760     // modbusTCPClient8.begin(serverIOEX8, 502);
761     // }
762     Serial.write('_'); Serial.write('!'); Serial.println(F("read buttons "));
763     CURRENT_DI_STATUS_WORD = modbusTCPClient8.inputRegisterRead(48);
764     Serial.write('_'); Serial.write('!'); Serial.println(CURRENT_DI_STATUS_WORD );
765     DUN_PSL = bitRead(CURRENT_DI_STATUS_WORD, 4);
766     txGUI[33] = DUN_PSL;
767     // if (DUN_PSL) {
768     //     ERROR = 36;
769     // }
770     DUN_PSH = bitRead(CURRENT_DI_STATUS_WORD, 3);
771     txGUI[32] = DUN_PSH;
772     // if (DUN_PSH) {
773     //     ERROR = 37;
774     // }
775     DUN_ZSL = bitRead(CURRENT_DI_STATUS_WORD, 5);
776     txGUI[35] = DUN_ZSL;
777     // if (!DUN_ZSL && FSM_STATE > BMM_IGNITION) {
778     //     ERROR = 38;
779     // }
780
781     DI_STATUS_CHANGE = CURRENT_DI_STATUS_WORD ^ LAST_DI_STATUS_WORD; //xor to check for a change in inputs
782     //Serial.print("CURRENT_DI_STATUS_WORD: "); Serial.println(CURRENT_DI_STATUS_WORD);
783     //Serial.print("DI_STATUS_CHANGE: "); Serial.println(DI_STATUS_CHANGE);
784     LAST_DI_STATUS_WORD = CURRENT_DI_STATUS_WORD;
785
786
787     if (DI_STATUS_CHANGE ) { //check for a change in inputs
788
789         //estop
790         if (bitRead(DI_STATUS_CHANGE, 0)) {
791             if (bitRead(CURRENT_DI_STATUS_WORD, 0)) { //estop
792                 ESTOP_FLAG = true; GRN_BTN_FLAG = false; AMB_BTN_FLAG = false; ERROR = 1;
793                 modbusTCPClient7.coilWrite(0, OFF); //green pilot OFF
794                 modbusTCPClient7.coilWrite(0, OFF); //AMB pilot OFF
795             }
796             else {
797                 ESTOP_FLAG == false;
798             }
799         }
800         if (ESTOP_FLAG == false) {
801
802             //amb button
803             if (bitRead(DI_STATUS_CHANGE, 2)) {
804                 if (bitRead(CURRENT_DI_STATUS_WORD, 2) && FSM_STATE == STABILIZE_MODE) {
805                     AMB_BTN_FLAG = true;
```

```
806     //modbusTCPClient7.coilWrite(0,0); //grn pilot light
807     modbusTCPClient7.coilWrite(7, 1); //twv901 switch reformat to PSA
808     modbusTCPClient7.coilWrite(1, 1); //amb pilot light ON
809 }
810 else {}
811 }

812 //grn btn
813 if (bitRead(DI_STATUS_CHANGE, 1)) {
814     if (bitRead(CURRENT_DI_STATUS_WORD, 1)) {
815         GRN_BTN_FLAG = true;
816         modbusTCPClient7.coilWrite(0, 1); //grn pilot
817         //modbusTCPClient7.coilWrite(2,0); //amb pilot
818     }
819 }
820 }
821 }
822 }
823 }

824 void error_Checker() {
825     //may be redundant but for safety
826     //shut gas down if an issue
827
828     if (ERROR) {
829         // Serial.println("ERROR CHECKER TRIGGERED!");
830
831         //GRN_BTN_FLAG = false;
832         AMB_BTN_FLAG = false; ESTOP_FLAG = true;
833
834         if (!modbusTCPClient1.connected()) {
835             modbusTCPClient1.begin(serverIOEX1, 502);
836         }
837         modbusTCPClient1.holdingRegisterWrite(42, OFF); //write channel 2 (FCV134)
838         if (!modbusTCPClient2.connected()) {
839             modbusTCPClient2.begin(serverIOEX2, 502);
840         }
841         modbusTCPClient1.holdingRegisterWrite(40, OFF); //write channel 4 (FCV141)
842
843         if (!modbusTCPClient7.connected()) {
844             modbusTCPClient7.begin(serverIOEX7, 502);
845         }
846         modbusTCPClient7.coilWrite(6, OFF); //TWV308
847         modbusTCPClient7.coilWrite(4, OFF); //XV1100
848         modbusTCPClient7.coilWrite(3, OFF); //XV501
849         modbusTCPClient7.coilWrite(8, OFF); //BMM_CR2
850         modbusTCPClient7.coilWrite(7, OFF); //TWV901
851         modbusTCPClient7.coilWrite(2, ON); //CH4 digital output xv909
852
853     }
854 }
855 else {
856     ESTOP_FLAG = false;
857 }
858
859
860 }
861 void steamPressureLow() {
862 }
863
864 void monitor_SR_Tube_Temps() {
865
866 }
867
868 void blinkGRN() {
869
870     if (CURRENT_MILLIS - PREVIOUS_MILLIS_5 >= 500 && GRN_BTN_FLAG == false && ESTOP_FLAG == false) {
871         PREVIOUS_MILLIS_5 = millis();
872         if (GRN_PLT_STATE == false) {
873             GRN_PLT_STATE = true;
874         }
875         else {
876             GRN_PLT_STATE = false;
877         }
878         modbusTCPClient7.coilWrite(0, GRN_PLT_STATE); //grn pilot
879     }
}
```

```
880 }
881 void blinkAMB() {
882     if (CURRENT_MILLIS - PREVIOUS_MILLIS_6 >= 500 && AMB_BTN_FLAG == false && ESTOP_FLAG == false) {
883         PREVIOUS_MILLIS_6 = millis();
884         if (AMB_PLT_STATE == false) {
885             AMB_PLT_STATE = true;
886         }
887         else {
888             AMB_PLT_STATE = false;
889         }
890         modbusTCPClient7.coilWrite(1, AMB_PLT_STATE); //grn pilot
891     }
892 }
893 }
894 void DB_RX() {
895     /*regRX = telGetValue(TEL_ADDR, REGRX);
896     if (regRX) {
897         telWriteValue(TEL_ADDR, REGRX, 0x00);
898
899     regRX = telGetValue(TEL_ADDR, REGRX);
900     SUPERHEAT_TIMER = telGetValue(TEL_ADDR, SHTMR);
901     BMM_OFF_TIMER = telGetValue(TEL_ADDR, BOTMR);
902     BMM_START_TIMER = telGetValue(TEL_ADDR, BSTMR);
903     BMM_PURGE_TIMER = telGetValue(TEL_ADDR, BPTMR);
904     BMM_IGNITION_TIMER = telGetValue(TEL_ADDR, BCTMR);
905     BURNER_RAMP_TIMER = telGetValue(TEL_ADDR, BRTMR);
906     BURNER_REACH_END_TIMER = telGetValue(TEL_ADDR, BETMR);
907     STEAM_GENERATION_TIMER = telGetValue(TEL_ADDR, SGTMR);
908     STEAM_AT_170PSI_TIMER = telGetValue(TEL_ADDR, SPTMR);
909     OPEN_SR_FUEL_TIMER = telGetValue(TEL_ADDR, SRTMR);
910     BLOWER_PURGE_SPEED = telGetValue(TEL_ADDR, BLPSD);
911     BLOWER_IGNITION_SPEED = telGetValue(TEL_ADDR, BLISD);
912     BLOWER_RAMP_END = telGetValue(TEL_ADDR, BLEND);
913     BLOWER_TOP_SPEED = telGetValue(TEL_ADDR, BLTSD);
914     RO_PUMP_AT_10_GRAMS_PER_SEC = telGetValue(TEL_ADDR, WP10G);
915     RO_PUMP_TOP_SPEED = telGetValue(TEL_ADDR, WPTSD);
916     FCV205_AT_35_PERCENT = telGetValue(TEL_ADDR, F205S);
917     FCV205_AT_50_PERCENT = telGetValue(TEL_ADDR, F205E);
918     FCV134_BURNER_FUEL_FLOW_IGNITION = telGetValue(TEL_ADDR, BFIGN);
919     FCV134_BURNER_FUEL_FLOW_RAMP_END = telGetValue(TEL_ADDR, BFRED);
920     FCV141_SR_FUEL_START_PERCENT = telGetValue(TEL_ADDR, SRFST);
921     FT132_PIPE_DIA_CONV = telGetValue(TEL_ADDR, FTDIA);
922     FT132_COUNTS_TO_G_PER_SEC = telGetValue(TEL_ADDR, FTGPS);
923     FT132_4MA_OFFSET = telGetValue(TEL_ADDR, FT4MA);
924     BURNER_TEMP_RAMP_END = telGetValue(TEL_ADDR, BTRED);
925     BURNER_TEMP_CROSSOVER = telGetValue(TEL_ADDR, BTCOV);
926     SR_FUEL_CUT = telGetValue(TEL_ADDR, SRFCT);
927
928 }*/
929 }
930 }
931 void DB_TX() {
932 /*
933     //motor fb,flow fb,fcv position fb
934     telWriteValue(TEL_ADDR, BL_FB, BLOWER_SPEED_FEEDBACK);
935     telWriteValue(TEL_ADDR, WP_FB, RO_PUMP_FEEDBACK);
936     telWriteValue(TEL_ADDR, BF_FB, FCV134_BURNER_FUEL_FLOW_FB);
937     telWriteValue(TEL_ADDR, SR_FB, FT132_ADJUSTED_MEASURE);
938     //pressure transducers
939     telWriteValue(TEL_ADDR, PT213, PT213_RO_PRESSURE);
940     telWriteValue(TEL_ADDR, PT318, PT318_HX406_OUTPUT_PRESSURE);
941     telWriteValue(TEL_ADDR, PT420, PT420_STEAM_EJECTOR_PRESSURE);
942     telWriteValue(TEL_ADDR, PT304, PT304_TWV308_INPUT_PRESSURE);
943     //thermocouples
944     telWriteValue(TEL_ADDR, TT142, TT142_SR_FUEL);
945     telWriteValue(TEL_ADDR, TT301, TT301_HX406_STEAM_OUT);
946     telWriteValue(TEL_ADDR, TT303, TT303_HX504_STEAM_OUT);
947     telWriteValue(TEL_ADDR, TT306, TT306_EJECTOR_STEAM_IN);
948     telWriteValue(TEL_ADDR, TT313, TT313_HX402_STEAM_OUT);
949     telWriteValue(TEL_ADDR, TT319, TT319_HX402_STEAM_SYSTEM);
950     telWriteValue(TEL_ADDR, TT407, TT407_STEAM_REFORMER_OUT_LREF);
951     telWriteValue(TEL_ADDR, TT408, TT408HTS_IN_LREF);
952     telWriteValue(TEL_ADDR, TT410, TT410HTS_OUT_LREF);
953     telWriteValue(TEL_ADDR, TT411, TT411FPZ_OUT_LREF);
```

```
954     telWriteValue(TEL_ADDR, TT430, TT430_SMR_TUBES_INLET);  
955     telWriteValue(TEL_ADDR, TT511, TT511_SILICON_CARBIDE_OUT);  
956     telWriteValue(TEL_ADDR, TT512, TT512_SILICON_CARBIDE_OUT);  
957     telWriteValue(TEL_ADDR, TT513, TT513_HX504_IN);  
958     telWriteValue(TEL_ADDR, TT514, TT514_HX504_OUT);  
959     telWriteValue(TEL_ADDR, TT441, TT441_SMR_TUBE1_OUT);  
960     telWriteValue(TEL_ADDR, TT442, TT442_SMR_TUBE2_OUT);  
961     telWriteValue(TEL_ADDR, TT443, TT443_SMR_TUBE3_OUT);  
962     telWriteValue(TEL_ADDR, TT444, TT444_SMR_TUBE4_OUT);  
963     telWriteValue(TEL_ADDR, TT445, TT445_SMR_TUBE5_OUT);  
964     telWriteValue(TEL_ADDR, TT446, TT446_SMR_TUBE6_OUT);  
965     telWriteValue(TEL_ADDR, TT447, TT447_SMR_TUBE7_OUT);  
966     telWriteValue(TEL_ADDR, TT448, TT448_SMR_TUBE8_OUT);  
967     telWriteValue(TEL_ADDR, TT449, TT449_SMR_TUBE9_OUT);  
968     //oci to bmm interface  
969     telWriteValue(TEL_ADDR, OCIIN, OCI_INPUT_STATUS_WORD);  
970     telWriteValue(TEL_ADDR, OCIOT, OCI_OUTPUT_STATUS_WORD);*/  
971 }  
972  
973 void DB_INIT() {  
974     /*  
975     telWriteValue(TEL_ADDR, REGRX, regRX);  
976     telWriteValue(TEL_ADDR, SHTMR, SUPERHEAT_TIMER);  
977     telWriteValue(TEL_ADDR, BOTMR, BMM_OFF_TIMER);  
978     telWriteValue(TEL_ADDR, BSTMR, BMM_START_TIMER);  
979     telWriteValue(TEL_ADDR, BPTMR, BMM_PURGE_TIMER);  
980     telWriteValue(TEL_ADDR, BCTMR, BMM_IGNITION_TIMER);  
981     telWriteValue(TEL_ADDR, BRTMR, BURNER_RAMP_TIMER);  
982     telWriteValue(TEL_ADDR, BETMR, BURNER_REACH_END_TIMER);  
983     telWriteValue(TEL_ADDR, SGTMR, STEAM_GENERATION_TIMER);  
984     telWriteValue(TEL_ADDR, SPTMR, STEAM_AT_170PSI_TIMER);  
985     telWriteValue(TEL_ADDR, SRTMR, OPEN_SR_FUEL_TIMER);  
986     telWriteValue(TEL_ADDR, BLPSD, BLOWER_PURGE_SPEED);  
987     telWriteValue(TEL_ADDR, BLISD, BLOWER_IGNITION_SPEED);  
988     telWriteValue(TEL_ADDR, BLEND, BLOWER_RAMP_END);  
989     telWriteValue(TEL_ADDR, BLTSD, BLOWER_TOP_SPEED);  
990     telWriteValue(TEL_ADDR, WP10G, RO_PUMP_AT_10_GRAMS_PER_SEC);  
991     telWriteValue(TEL_ADDR, WPTSD, RO_PUMP_TOP_SPEED);  
992     telWriteValue(TEL_ADDR, F205S, FCV205_AT_35_PERCENT);  
993     telWriteValue(TEL_ADDR, F205E, FCV205_AT_50_PERCENT);  
994     telWriteValue(TEL_ADDR, BFIGN, FCV134_BURNER_FUEL_FLOW_IGNITION);  
995     telWriteValue(TEL_ADDR, BFRED, FCV134_BURNER_FUEL_FLOW_RAMP_END);  
996     telWriteValue(TEL_ADDR, SRFST, FCV141_SR_FUEL_START_PERCENT);  
997     telWriteValue(TEL_ADDR, FTDIA, FT132_PIPE_DIA_CONV);  
998     telWriteValue(TEL_ADDR, FTGPS, FT132_COUNTS_TO_G_PER_SEC);  
999     telWriteValue(TEL_ADDR, FT4MA, FT132_4MA_OFFSET);  
1000    telWriteValue(TEL_ADDR, BTRED, BURNER_TEMP_RAMP_END);  
1001    telWriteValue(TEL_ADDR, BTCOV, BURNER_TEMP_CROSSOVER);  
1002    telWriteValue(TEL_ADDR, SRFCT, SR_FUEL_CUT);  
1003 */  
1004 }  
1005 void readAI() {  
1006     //function for reading  
1007     //motor speed  
1008     //fcv134fb ft132fb  
1009  
1010     BLOWER_SPEED_FEEDBACK = map(modbusTCPClient6.inputRegisterRead(0), 0, 65535, 0, 60); //10volts/10000counts  
1011     FCV134_BURNER_FUEL_FLOW_FB = map(modbusTCPClient6.inputRegisterRead(7), 100, 65535, 0, 100);  
1012     RO_PUMP_FEEDBACK = map(modbusTCPClient6.inputRegisterRead(1), 0, 65535, 0, 60);  
1013     FT132_NG_FEED_FLOW = modbusTCPClient6.inputRegisterRead(2);  
1014  
1015     txGUI[24] = BLOWER_SPEED_FEEDBACK;  
1016     txGUI[25] = RO_PUMP_FEEDBACK;  
1017     txGUI[27] = FT132_NG_FEED_FLOW;  
1018     txGUI[34] = FCV134_BURNER_FUEL_FLOW_FB;  
1019 }  
1020  
1021 void GUI() {  
1022  
1023     txGUI[0] = TT142_SR_FUEL;  
1024     txGUI[1] = TT301_HX406_STEAM_OUT;  
1025     txGUI[2] = TT303_HX504_STEAM_OUT;  
1026     txGUI[3] = TT306_EJECTOR_STEAM_IN;  
1027     txGUI[4] = TT313_HX402_STEAM_OUT;
```

```
1028 txGUI[5] = TT319_HX402_STEAM_SYSTEM;
1029 txGUI[6] = TT407_STEAM_REFORMER_OUT_LREF;
1030 txGUI[7] = TT408HTS_IN_LREF;
1031 txGUI[8] = TT410HTS_OUT_LREF;
1032 txGUI[9] = TT411FPZ_OUT_LREF;
1033 txGUI[10] = TT430_SMR_TUBES_INLET;
1034 txGUI[11] = TT511_SILICON_CARBIDE_OUT;
1035 txGUI[12] = TT512_SILICON_CARBIDE_OUT;
1036 txGUI[13] = TT513_HX504_IN;
1037 txGUI[14] = TT514_HX504_OUT;
1038 txGUI[15] = TT441_SMR_TUBE1_OUT;
1039 txGUI[16] = TT442_SMR_TUBE2_OUT;
1040 txGUI[17] = TT443_SMR_TUBE3_OUT;
1041 txGUI[18] = TT444_SMR_TUBE4_OUT;
1042 txGUI[19] = TT445_SMR_TUBE5_OUT;
1043 txGUI[20] = TT446_SMR_TUBE6_OUT;
1044 txGUI[21] = TT447_SMR_TUBE7_OUT;
1045 txGUI[22] = TT448_SMR_TUBE8_OUT;
1046 txGUI[23] = TT449_SMR_TUBE9_OUT;
1047 txGUI[24] = BLOWER_SPEED_FEEDBACK;
1048 txGUI[25] = RO_PUMP_FEEDBACK;
1049 txGUI[26] = ERROR;
1050 txGUI[27] = FT132_NG_FEED_FLOW;
1051 txGUI[28] = PT318_HX406_OUTPUT_PRESSURE;
1052 txGUI[29] = PT213_RO_PRESSURE;
1053 txGUI[30] = PT420_STEAM_EJECTOR_PRESSURE;
1054 txGUI[31] = PT304_TWV308_INPUT_PRESSURE;
1055 txGUI[32] = DUN_PSH;
1056 txGUI[33] = DUN_PSL;
1057 txGUI[34] = FCV134_BURNER_FUEL_FLOW_FB;
1058 txGUI[35] = DUN_ZSL;
1059
1060 for (int i = 0; i <= 35; i++) {
1061     // if (txGUI[i] != oldtxGUI[i] ) {
1062     //print new value to dashboard
1063     Serial.write('_');
1064     Serial.write(myChars[i]);
1065     Serial.println(txGUI[i]);
1066     oldtxGUI[i] = txGUI[i];
1067     // }
1068
1069 }
1070 }
1071 }
1072 void preTransmission() {
1073     digitalWrite(MAX485_RE_NEG, 1);
1074     digitalWrite(MAX485_DE, 1);
1075 }
1076 void postTransmission() {
1077     digitalWrite(MAX485_RE_NEG, 0);
1078     digitalWrite(MAX485_DE, 0);
1079 }
1080 /*
1081 error map
1082 1-estop
1083 2-fcv134 timeout integrity check
1084 3-pt318 over 250psi
1085 4-pt213 over 250psi
1086 5-pt420 over 250psi
1087 6-pt304 over 250psi
1088 7-super heat error
1089 8-No Flame detected
1090 9-Steam gen timeout
1091 10-Loss of Flame
1092 11-Burner reach 800 c timeout
1093
1094 12-Temperature > 1000C
1095 13-temp error
1096 14-
1097 15-
1098 16-
1099 17-
```

```
1102    18-
1103    19-
1104    20-
1105    21-
1106    22-
1107    23-
1108    24-
1109    25-
1110    26-
1111    27-
1112    28-
1113    29-
1114    30-
1115    31-
1116    32-
1117    33-
1118    34-
1119    35-
1120
1121    36-
1122
1123 */
```