```
//Revision of JBOX with moxa and stride ioexpanders

#include <Wire.h> //For I2C comm
#include <Adafruit_GFX.h>
#include <LedHelper.h>
#include "Adafruit_LEDBackpack.h"
#include "Adafruit_LiquidCrystal.h"
#include<SPI.h>//serial peripheral interface
#include<Ethernet3.h>//for ethernet
#include<ArduinoRS485.h> //ArduinoModbus depends on the ArduinoRS485 lib
#include<ArduinoModbus.h>//for modbus tcp/ip
#include<ModbusMaster.h>//for oci417.10 rs485 interface with BMM

#include<pid.h>
#include "DEFINES.h"
#include "GLOBALS.h"

// OCI417.10 rs485 interface with BMM
ModbusMaster NODE;

//modbus tcp/ip communication declarations


byte mac[] = { 0xdE, 0xAD, 0xBE, 0xEF, 0xED};

IPAddress ip(128, 1, 1, 110); //needs to be similar to Acromag because will not connect

EthernetClient ethClient1;
EthernetClient ethClient2;
EthernetClient ethClient3;
EthernetClient ethClient4;
EthernetClient ethClient5;
EthernetClient ethClient6;
EthernetClient ethClient7;
EthernetClient ethClient8;

ModbusTCPClient modbusTCPClient1(ethClient1);
ModbusTCPClient modbusTCPClient2(ethClient2);
ModbusTCPClient modbusTCPClient3(ethClient3);
ModbusTCPClient modbusTCPClient4(ethClient4);
ModbusTCPClient modbusTCPClient5(ethClient5);
ModbusTCPClient modbusTCPClient6(ethClient6);
ModbusTCPClient modbusTCPClient7(ethClient7);
ModbusTCPClient modbusTCPClient8(ethClient8);

IPAddress serverIOEX1(128, 1, 1, 101); // IP of ioex1 STRIDE SIO-MB04DAS 4CH Analog Output
IPAddress serverIOEX2(128, 1, 1, 102); // IP of ioex2 STRIDE SIO-MB04DAS 4CH Analog Output
IPAddress serverIOEX3(128, 1, 1, 103); // IP of ioex3 STRIDE SIO-MB04DAS 8CH TC/mV Input
IPAddress serverIOEX4(128, 1, 1, 104); // IP of ioex4 STRIDE SIO-MB04DAS 8CH TC/mV Input
IPAddress serverIOEX5(128, 1, 1, 105); // IP of ioex5 STRIDE SIO-MB04DAS 8CH TC/mV Input
IPAddress serverIOEX6(128, 1, 1, 106); // IP of ioex6 MOXA IOLOGIK E1240 8CH Analog Input
IPAddress serverIOEX7(128, 1, 1, 107); // IP of ioex7 MOXA IOLOGIK E1211 16CH Digital Output
IPAddress serverIOEX8(128, 1, 1, 108); // IP of ioex8 MOXA IOLOGIK E1210 16CH Digital Input

//PID controllers
PID SUPER_HEAT_TT303 = PID(0.5, RO_PUMP_TOP_SPEED, RO_PUMP_AT_10_GRAMS_PER_SEC, 500, 0.01, 0.5);
//dt,max,min,kp,kd,ki
PID SUPER_HEAT_TT306 = PID(0.5, RO_PUMP_TOP_SPEED, RO_PUMP_AT_10_GRAMS_PER_SEC, 500, 0.01, 0.5);
//dt,max,min,kp,kd,ki
PID STEAM_GEN_PID = PID(0.5, BLOWER_TOP_SPEED, BLOWER_RAMP_BEGIN , 500, 0.1, .3);//dt,max,min,kp,kd,ki;//0.1,100,-
100,0.8,0.01,0.5
PID STEAM_GEN_PID_soft = PID(0.5, BLOWER_TOP_SPEED, BLOWER_RAMP_BEGIN , 100, 0, 0.5);
PID BURNER_TEMP_CROSSOVER_PID = PID(0.5, 10000, 6500, 1000, 0, 0); //dt,max,min,kp,kd,ki;//
PID OPEN_SR_FUEL_PID = PID(0.5, 10000, 0, 100, 0, 0);//dt,max,min,kp,kd,ki;//
PID FCV205_PID = PID(0.5, 3000, 5200, 500, 0, 0);

//Daughter Board periferals
SmallMatrix smallMatrix[3] = {SmallMatrix(0x70), SmallMatrix(0x71), SmallMatrix(0x72) };
LargeMatrix bigMatrix[3] = {LargeMatrix(0x73), LargeMatrix(0x74), LargeMatrix(0x75) };
Adafruit_LiquidCrystal lcd(0);

//telemetry
union floatToBytes  {
  char  asBytes[4] = {0};
```

```
  float asFloat;
};

//local viewer
long txGUI[36] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, //tcs
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, //tcs
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 //analog inputs
                 };
long oldtxGUI[36] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, //tcs
                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, //tcs
                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 //analog inputs
                    };

char myChars[36] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
                    'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
                    'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'
                   };


void setup() {

  pinMode(MAX485_RE_NEG, OUTPUT);
  pinMode(MAX485_DE, OUTPUT);

  // Init in receive mode rs485
  digitalWrite(MAX485_RE_NEG, 0);
  digitalWrite(MAX485_DE, 0);

  Serial.begin(9600); // for dedbugging
  Serial.write('_'); Serial.write('!'); Serial.println(F("Serial Connected "));
  Serial1.begin(9600);//FOR OCI MODULE rs485
  NODE.begin(1, Serial1); //unit id1 on serial1
  NODE.preTransmission(preTransmission);
  NODE.postTransmission(postTransmission);
  Serial.write('_'); Serial.write('!'); Serial.println(F("Connected to OCI "));
  delay(500);//wait for system to boot up

  pinMode(LED_PWR, OUTPUT);
  digitalWrite(LED_PWR, HIGH);
  pinMode(TRACO_24VDC, OUTPUT);
  digitalWrite(TRACO_24VDC, HIGH);
  pinMode(ESTOP_BREAK, OUTPUT);
  digitalWrite(ESTOP_BREAK, HIGH);

  // reseting wifichip
  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH);
  delay(50);
  Serial.write('_'); Serial.write('!'); Serial.println(F("Have Reset WIFI chip "));
  //start the Ethernet connection and the server:


  Ethernet.begin(mac, ip);
  delay(5);
  //Ethernet.phyMode(FULL_DUPLEX_100_AUTONEG);
  Ethernet.phyMode(FULL_DUPLEX_100);
  // Ethernet.phyMode(HALF_DUPLEX_100);
  //  Ethernet.phyMode(FULL_DUPLEX_10);
  //  Ethernet.phyMode(HALF_DUPLEX_10);

  connect_IO_Expanders();
  Serial.write('_'); Serial.write('!'); Serial.println(F("have connected io expanders "));
  //integrityCheck();
  Serial.write('_'); Serial.write('!'); Serial.println(F("integrity check done  "));
  readBtn();
  Serial.write('_'); Serial.write('!'); Serial.println(F("read buttons done  "));
  readOCI();
  Serial.write('_'); Serial.write('!'); Serial.println(F("read oci done  "));
  Serial.begin(9600);
  CURRENT_MILLIS = millis(); //So as to not divide by zero on first looptime.
  LOOP_MILLIS = millis();

  lcd.begin(16, 2);
  Serial.write('_'); Serial.write('!'); Serial.println(F("lcd on  "));
```

```
146
147   DB_INIT();
148   Serial.write('_'); Serial.write('!'); Serial.println(F("DB Init  "));
149   SENSOR_INTEGRITY_CHECK = true;
150
151 }
152
153 void loop() {
154
155   if (FSM_STATE == 0) {
156     FSM_STATE_STRING = "INITIALIZE";
157   }
158   if (FSM_STATE == 1) {
159     FSM_STATE_STRING = "DEPRESSURIZE";
160   }
161   if (FSM_STATE == 2) {
162     FSM_STATE_STRING = "SUPERHEAT.TEST";
163   }
164   if (FSM_STATE == 3) {
165     FSM_STATE_STRING = "BMM.OFF";
166   }
167   if (FSM_STATE == 4) {
168     FSM_STATE_STRING = "BMM.ON";
169   }
170   if (FSM_STATE == 5) {
171     FSM_STATE_STRING = "BMM.PURGE";
172   }
173   if (FSM_STATE == 6) {
174     FSM_STATE_STRING = "BMM.IGNITION";
175   }
176   if (FSM_STATE == 7) {
177     FSM_STATE_STRING = "BURNER.RAMP";
178   }
179   if (FSM_STATE == 8) {
180     FSM_STATE_STRING = "STEAM.GEN";
181   }
182   if (FSM_STATE == 9) {
183     FSM_STATE_STRING = "OPEN.SR.FUEL";
184   }
185   if (FSM_STATE == 10) {
186     FSM_STATE_STRING = "IDLE";
187   }
188   if (FSM_STATE == 11) {
189     FSM_STATE_STRING = "STABILIZE";
190   }
191   LOOP_MILLIS -=  CURRENT_MILLIS;
192   Serial.write('_'); Serial.write('@'); Serial.println(FSM_STATE_STRING);
193   Serial.write('_'); Serial.write('!'); Serial.print("LOOP.MILLIS: "); Serial.println(LOOP_MILLIS);
194
195
196
197   CURRENT_MILLIS = millis();
198
199   DB_RX();
200   error_Checker();
201   connect_IO_Expanders();
202   integrityCheck();
203   superheatTest();
204   readTCs();
205   readBtn();
206   readOCI();
207   readPTs();
208   blinkGRN();
209   blinkAMB();
210   readOut();
211   readAI();
212   rampWaterPump();
213   GUI();
214
215   if (GRN_BTN_FLAG && !ESTOP_FLAG) {
216
217
218
219     switch (FSM_STATE) {
```

```
220
221
222
223       case INITIALIZE: //initialize//0
224
225         if (!SENSOR_INTEGRITY_CHECK) {
226
227           break;
228         }
229         modbusTCPClient3.coilWrite(2, 1); //CH2 digital output xv909//ON
230
231
232         if (!COMBUSTION_PRESSURE_SWITCH) {
233
234           modbusTCPClient1.holdingRegisterWrite(40, BLOWER_PURGE_SPEED); //write channel 0 (BLWRSpeed)
235           modbusTCPClient7.coilWrite(9, 1); //BLWR_EN..The logic is opposite
236           break;
237         }
238
239         if (DUN_PSL) {
240           // Serial.print("Break due to DUN_PSL Switch: ");
241           // Serial.println(DUN_PSL);
242           break;
243         }
244
245         //check for RO water???//need another sensor
246         modbusTCPClient1.holdingRegisterWrite(40, OFF); //write channel 0 (BLWRSpeed)
247         modbusTCPClient7.coilWrite(9, 0); //BLWR_EN..
248
249         if (BMM_PROOF_OF_FLAME) { //get that flame out before proceed
250
251           if (!modbusTCPClient1.connected()) {
252             modbusTCPClient1.begin(serverIOEX1, 502);
253           }
254           if (!modbusTCPClient2.connected()) {
255             modbusTCPClient2.begin(serverIOEX2, 502);
256           }
257
258           modbusTCPClient1.holdingRegisterWrite(42, OFF); //write channel 2 ioex1 (FCV134)
259           modbusTCPClient2.holdingRegisterWrite(40, OFF); //write channel 0 ioex2 (FCV141)
260
261           if (!modbusTCPClient7.connected()) {
262             modbusTCPClient7.begin(serverIOEX7, 502);
263           }
264           modbusTCPClient7.coilWrite(6, OFF); //TWV308
265           modbusTCPClient7.coilWrite(4, OFF); //XV1100
266           modbusTCPClient7.coilWrite(3, OFF); //XV501
267           modbusTCPClient7.coilWrite(8, OFF); //BMM_CR2
268           modbusTCPClient7.coilWrite(7, OFF); //TWV901
269           modbusTCPClient7.coilWrite(2, ON); //CH4 digital output xv909
270           break;
271
272         }
273         FSM_STATE = DEPRESSURIZE;
274         break;
275
276       case DEPRESSURIZE: //depressurise reformer//1
277
278         modbusTCPClient7.coilWrite(6, OFF); //TWV308
279
280         if (PT420_STEAM_EJECTOR_PRESSURE >= 20
281             || PT213_RO_PRESSURE >= 70
282             || PT318_HX406_OUTPUT_PRESSURE >= 70
283             || PT304_TWV308_INPUT_PRESSURE >= 70) {
284
285           modbusTCPClient7.coilWrite(2, ON); //CH2 digital output xv909
286
287           break;
288         }
289
290         modbusTCPClient7.coilWrite(2, OFF); //CH2 digital output xv909
291
292         //250 psi
293         if (PT213_RO_PRESSURE < 250) {
```

```
294
295           //just guessing on the fcv205 flow, as long as we stay beneath 250psi
296           modbusTCPClient1.holdingRegisterWrite(40, BLOWER_PURGE_SPEED); //write channel 0 (BLWRSpeed)
297           modbusTCPClient1.holdingRegisterWrite(42, FCV205_AT_35_PERCENT); //write channel 2 (FCV205)//made steam at
    1000//last 3000//~about 35 percent
298           // modbusTCPClient1.holdingRegisterWrite(41, RO_PUMP_AT_10_GRAMS_PER_SEC); //write channel 1 (WP_Speed)
    made steam at 1*2000//last 3*2000
299           modbusTCPClient7.coilWrite(10, ON); //WP_EN..
300           modbusTCPClient7.coilWrite(9, ON); //BLWR_EN..
301           FSM_STATE = SUPERHEAT_TEST;
302           PREVIOUS_MILLIS = millis();
303           break;
304         }
305         break;
306
307       case SUPERHEAT_TEST://Superheat Test //2
308
309         //make sure we are closed//xv909
310         modbusTCPClient7.coilWrite(2, OFF); //CH4 digital output xv909
311         //monitor tc 407 & 408
312         //need a timer
313
314         if (CURRENT_MILLIS - PREVIOUS_MILLIS <= SUPERHEAT_TIMER) {
315
316           TT303_HX504_STEAM_OUT = modbusTCPClient3.holdingRegisterRead(42) * (0.095);
317           //guessing on 300 degrees centigrade
318           if (TT303_HX504_STEAM_OUT > 300) {
319             ERROR = 7; FSM_STATE = INITIALIZE; break;
320           }
321           break;
322         }
323         FSM_STATE = BMM_OFF;
324         PREVIOUS_MILLIS = millis();
325         break;
326
327       case BMM_OFF://bmm off //30 seconds;
328
329         if (CURRENT_MILLIS - PREVIOUS_MILLIS <= BMM_OFF_TIMER) {
330
331           modbusTCPClient1.holdingRegisterWrite(40, OFF); //write channel 0 (BLWRSpeed)
332           modbusTCPClient7.coilWrite(8, OFF); //BMM_CR2 bmm off
333           break;
334         }
335         FSM_STATE = BMM_ON;
336         PREVIOUS_MILLIS = millis();
337         break;
338
339       case BMM_ON:
340         //bmm on //5 seconds
341         modbusTCPClient7.coilWrite(8, ON); //BMM_CR2 turn on
342         if (CURRENT_MILLIS - PREVIOUS_MILLIS <= BMM_START_TIMER) {
343           break;
344         }
345         PREVIOUS_MILLIS = millis();
346         FSM_STATE = BMM_PURGE;
347         break;
348
349       case BMM_PURGE:
350         //bmm purge
351
352         if (CURRENT_MILLIS - PREVIOUS_MILLIS <= BMM_PURGE_TIMER) {
353           modbusTCPClient1.holdingRegisterWrite(40, BLOWER_PURGE_SPEED); //write channel 0 (BLWRSpeed)
354           break;
355         }
356         else {
357           PREVIOUS_MILLIS_7 = millis();
358           PREVIOUS_MILLIS = millis();
359           FSM_STATE = BMM_IGNITION;
360           old_TT511 =  modbusTCPClient5.holdingRegisterRead(41) * (0.095);
361           modbusTCPClient1.holdingRegisterWrite(40, BLOWER_IGNITION_SPEED); //write channel 0 (BLWRSpeed)
362           modbusTCPClient1.holdingRegisterWrite(42, FCV134_BURNER_FUEL_FLOW_IGNITION); //write channel 2 (FCV134)
363           break;
364         }
365         break;
```

```
366
367       case BMM_IGNITION:
368         //BMM ignition
369
370         current_TT511 =  modbusTCPClient5.holdingRegisterRead(41) * (0.095);
371
372
373         if (current_TT511 >= (old_TT511 + 40) || BMM_PROOF_OF_FLAME) {
374
375           FSM_STATE = BURNER_RAMP;
376           PREVIOUS_MILLIS = millis();
377           PREVIOUS_MILLIS_2 = millis();
378           modbusTCPClient1.holdingRegisterWrite(42, FCV134_BURNER_FUEL_FLOW_RAMP_BEGIN); //16121 best thus far, fuel
379           modbusTCPClient1.holdingRegisterWrite(40, BLOWER_RAMP_BEGIN); //7100 best thus far, blwr
380           break;
381         }
382
383
384         if (CURRENT_MILLIS - PREVIOUS_MILLIS >= BMM_IGNITION_TIMER) {
385
386           if (!BMM_PROOF_OF_FLAME) {
387             ERROR = 8;
388             FSM_STATE = INITIALIZE;
389             break;
390           }
391           else {
392             FSM_STATE = BURNER_RAMP;
393             modbusTCPClient1.holdingRegisterWrite(42, FCV134_BURNER_FUEL_FLOW_RAMP_BEGIN); //16121 best thus far,
    fuel
394             modbusTCPClient1.holdingRegisterWrite(40, BLOWER_RAMP_BEGIN); //7100 best thus far, blwr
395             PREVIOUS_MILLIS = millis();
396             PREVIOUS_MILLIS_2 = millis();
397             break;
398           }
399         }
400         break;
401
402       case BURNER_RAMP:
403         //start burner ramp//15 minutes//900 seconds
404         //burner ramp is 4mins
405         //timing to reach 800c is 15 mins
406
407         if (CURRENT_MILLIS - PREVIOUS_MILLIS >= BURNER_REACH_END_TIMER) {
408
409           if (TT511_SILICON_CARBIDE_OUT >= BURNER_TEMP_RAMP_END) {
410             FSM_STATE = STEAM_GEN;
411             PREVIOUS_MILLIS = millis();
412             break;
413           }
414
415           if ( TT513_HX504_IN >= BURNER_TEMP_RAMP_END) {
416             FSM_STATE = STEAM_GEN;
417             PREVIOUS_MILLIS = millis();
418             break;
419           }
420
421           ERROR = 11; FSM_STATE = INITIALIZE; break;
422         }
423
424         if (TT511_SILICON_CARBIDE_OUT >= BURNER_TEMP_RAMP_END) {
425           FSM_STATE = STEAM_GEN;
426           PREVIOUS_MILLIS = millis();
427           break;
428         }
429
430         if ( TT513_HX504_IN >= BURNER_TEMP_RAMP_END) {
431           FSM_STATE = STEAM_GEN;
432           PREVIOUS_MILLIS = millis();
433           break;
434         }
435
436         BLOWER_SPEED_FEEDBACK = map( modbusTCPClient6.inputRegisterRead(0), 0, 65535, 0, 60); //10volts/10000counts
437         FCV134_BURNER_FUEL_FLOW_FB = map( modbusTCPClient6.inputRegisterRead(7), 100, 65535, 0, 100); //ref:43028
    read Channel 7 FCV134//~10079
```

```
438
439         if (CURRENT_MILLIS - PREVIOUS_MILLIS <= BURNER_RAMP_TIMER) {
440
441           if (CURRENT_MILLIS - PREVIOUS_MILLIS_2 >= 5000) {
442
443             //if(BLOWER_SPEED_FEEDBACK<=14030){
444             //modbusTCPClient1.holdingRegisterWrite(0x12,BLOWER_SPEED_FEEDBACK+220);}//write channel 0 blwr
    speed//44counts/sec
445             //if(FCV134_BURNER_FUEL_FLOW_FB<=16121){
446             //modbusTCPClient1.holdingRegisterWrite(0x14,FCV134_BURNER_FUEL_FLOW_FB+125);}//25counts/sec
447
448             modbusTCPClient1.holdingRegisterWrite(42, FCV134_BURNER_FUEL_FLOW_RAMP_END); //16121 best thus far, fuel
449             modbusTCPClient1.holdingRegisterWrite(40, BLOWER_RAMP_END); //7100 best thus far, blwr
450             PREVIOUS_MILLIS_2 = millis();
451             break;
452           }
453         }
454         else {
455           //modbusTCPClient1.holdingRegisterWrite(0x14,16121);//25counts/sec
456           //modbusTCPClient1.holdingRegisterWrite(0x12,13000);//write channel 0 blwr speed//44counts/sec
457           break;
458         }
459         break;
460
461       case STEAM_GEN://Steam generation
462
463
464         //steam gen timeout 30 mins
465         if (CURRENT_MILLIS - PREVIOUS_MILLIS >= STEAM_GENERATION_TIMER) {
466           ERROR = 9;
467           FSM_STATE = INITIALIZE;
468           break;
469         }
470
471         //PID control blower to acheive 170PSI pt304
472         BLOWER_SPEED_FEEDBACK = map(modbusTCPClient6.inputRegisterRead(0), 0, 65535, 0, 60); //10volts/20000counts
473
474         steamGenPID();
475
476         if (PT304_TWV308_INPUT_PRESSURE <= 172 && PT304_TWV308_INPUT_PRESSURE >= 168) {
477           if (PSI_INIT_TIMER == false) {
478             PREVIOUS_MILLIS_1 = millis();
479             PSI_INIT_TIMER = true;
480             break;
481           }
482
483           if (CURRENT_MILLIS - PREVIOUS_MILLIS_1 >= STEAM_AT_170PSI_TIMER && PSI_INIT_TIMER == true) {
484             BLOWER_SPEED_AT_170PSI = BLOWER_SPEED_FEEDBACK;
485             modbusTCPClient1.holdingRegisterWrite(40, BLOWER_SPEED_AT_170PSI);
486             modbusTCPClient7.coilWrite(3, ON); //XV501 ON
487             FSM_STATE = OPEN_SR_FUEL; PREVIOUS_MILLIS = millis(); PSI_INIT_TIMER = false; BURNER_TEMP_CROSSOVER =
    TT513_HX504_IN; break;
488           }
489         }
490         else {
491           PSI_INIT_TIMER = false;  //If not in bounds of 170-psi, reset timer. Turn on xv501
492           modbusTCPClient6.coilWrite(3, OFF);
493           }
494           break;
495
496       case OPEN_SR_FUEL:
497         //OPEN SR FUEL
498
499         //Point twv308 to reformer
500         if (!modbusTCPClient7.connected()) {
501           modbusTCPClient7.begin(serverIOEX7, 502);
502         }
503         modbusTCPClient7.coilWrite(6, ON); //TWV308//TWV308 ON//direct steam to reformer
504         modbusTCPClient7.coilWrite(5, ON); //XV801 On to induce gas feed to reformer
505
506         //assign percentage to fcv141 sr fuel flow control //fcv205 kicked to 50%
507         if (!modbusTCPClient2.connected()) {
508           modbusTCPClient2.begin(serverIOEX2, 502);
509         }
```

```
510         modbusTCPClient2.holdingRegisterWrite(40, FCV141_SR_FUEL_START_PERCENT); //write channel 4 (FCV141)
511         if (!modbusTCPClient1.connected()) {
512           modbusTCPClient1.begin(serverIOEX1, 502);
513         }
514         modbusTCPClient1.holdingRegisterWrite(43, FCV205_AT_50_PERCENT); //write channel 3 (FCV205)
515
516         //read fcv134 for burner temp control
517         if (!modbusTCPClient6.connected()) {
518           modbusTCPClient6.begin(serverIOEX6, 502);
519         }
520         FCV134_BURNER_FUEL_FLOW_FB = map( modbusTCPClient6.inputRegisterRead(7), 100, 65535, 0, 100); //read Channel
    7 FCV134
521
522         if (CURRENT_MILLIS - PREVIOUS_MILLIS <= OPEN_SR_FUEL_TIMER) {
523
524           //pid burner control for 880C
525           BURNER_FUEL_CUT_OFFSET = BURNER_TEMP_CROSSOVER_PID.calc_reverse(BURNER_TEMP_CROSSOVER , TT513_HX504_IN);
526           // Serial.print("BURNER_FUEL_CUT_OFFSET: "); Serial.println(BURNER_FUEL_CUT_OFFSET);
527           //  Serial.print("TT513_HX504_IN: "); Serial.println(TT513_HX504_IN);
528           if (!modbusTCPClient1.connected()) {
529             modbusTCPClient1.begin(serverIOEX1, 502);
530           }
531           modbusTCPClient1.holdingRegisterWrite(42, BURNER_FUEL_CUT_OFFSET);//write channel 2 (FCV134)
532
533           //pid srfuel control for 0.3g/second
534           if (!modbusTCPClient6.connected()) {
535             modbusTCPClient6.begin(serverIOEX6, 502);
536
537           }
538           FT132_NG_FEED_FLOW = modbusTCPClient6.inputRegisterRead(2);
539           FT132_ADJUSTED_MEASURE = (FT132_NG_FEED_FLOW * FT132_PIPE_DIA_CONV * FT132_COUNTS_TO_G_PER_SEC) -
    FT132_4MA_OFFSET;
540           SR_FUEL_OFFSET = OPEN_SR_FUEL_PID.calculate(SR_FUEL_CUT, FT132_ADJUSTED_MEASURE); //calc offset for FCV141
541           //  Serial.print("SR_FUEL_OFFSET: "); Serial.println(SR_FUEL_OFFSET);
542           //  Serial.print("FT132_ADJUSTED_MEASURE: "); Serial.println(FT132_ADJUSTED_MEASURE);
543           if (!modbusTCPClient2.connected()) {
544             modbusTCPClient2.begin(serverIOEX2, 502);
545           }
546           modbusTCPClient2.holdingRegisterWrite(40, SR_FUEL_OFFSET); //write channel 4 (FCV141)
547
548           //check conformance and restart time if out of bounds
549           if ((FT132_ADJUSTED_MEASURE > (SR_FUEL_CUT + 0.2)) || (BURNER_TEMP_CROSSOVER < (SR_FUEL_CUT - 0.2))) {
550             PREVIOUS_MILLIS = millis();
551             break;
552           }
553
554           //reset timing if temp falls out of threshold
555           if ((BURNER_TEMP_CROSSOVER > (TT513_HX504_IN + 5)) || (BURNER_TEMP_CROSSOVER < (TT513_HX504_IN - 5))) {
556             PREVIOUS_MILLIS = millis();
557             break;
558           }
559
560         }
561
562         else {
563           if (ERROR == 0) {
564             FSM_STATE = IDLE_MODE;
565             PREVIOUS_MILLIS = millis();
566             break;
567           }
568           FSM_STATE = OPEN_SR_FUEL; PREVIOUS_MILLIS = millis(); break;
569         }
570
571         break;
572
573       case IDLE_MODE:
574         //idle hold mode blower/burner pid
575         BLOWER_SPEED_FEEDBACK = map(modbusTCPClient6.inputRegisterRead(0), 0, 65535, 0, 60); //10volts/10000counts
576         FCV134_BURNER_FUEL_FLOW_FB = map(modbusTCPClient6.inputRegisterRead(7), 100, 65535, 0, 100); //ref:43028 read
    Channel 7 FCV134
577
578         //3 mins idle ????
579         if (CURRENT_MILLIS - PREVIOUS_MILLIS <= 180000) {
580
```

```
          //burner fuel PID
          BURNER_FUEL_CUT_OFFSET = OPEN_SR_FUEL_PID.calculate(BURNER_TEMP_CROSSOVER , TT513_HX504_IN);
          BURNER_FUEL_CUT = FCV134_BURNER_FUEL_FLOW_FB + BURNER_FUEL_CUT_OFFSET;
          modbusTCPClient1.holdingRegisterWrite(42, BURNER_FUEL_CUT);//write channel 2 (FCV134)

          //blower pid
          steamGenPID();

          //check if out of threshold
          if ((BURNER_TEMP_CROSSOVER > (TT513_HX504_IN + 5)) || (BURNER_TEMP_CROSSOVER < (TT513_HX504_IN - 5))) {
            PREVIOUS_MILLIS = millis();
            break;
          }
          if (PT304_TWV308_INPUT_PRESSURE <= 171 && PT304_TWV308_INPUT_PRESSURE >= 169) {
            PREVIOUS_MILLIS = millis();
            break;
          }
        }

        else {
          FSM_STATE = STABILIZE_MODE;
          break;
        }
        break;

      case STABILIZE_MODE:
        //reformer stabilize
        break;

      default:
        // Serial.println("Default.");
        break;

    }//end switch

  }//end if(GRN_BTN_FLAG && !ESTOP_FLAG)

  LOOP_MILLIS = millis();
  //end loop
}
```