```
//Arduino NANO (use old Bootloader for Chinese clones)
//all debug lines switched OFF in this version
// Digital in 9 is used for a speed switch in this version
// use a simple ON-OFF switch for speed select
//BugFix Version working with EKOS-KSTARS-INDI
//läuft auch mit ASCOM; nicht mit Moonlite manual!
//C mit 1µF RST nach GND
//old Bootloader bei CH340
//normal Bootload bei FT323

//#include <SoftwareSerial.h> //war auskommentiert
#include <AccelStepper.h>
#include <EEPROM.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <TimerOne.h>

#define BTN_IN 7
#define BTN_OUT 8
#define BTN_STEP 32   //basic value
#define BTN_TURBS 9   //input pin for turbo switch

#define PIN_DRIVER_ENABLE 4
#define PIN_DRIVER_DIR 3
#define PIN_DRIVER_STEP 5

#define ONE_WIRE_BUS 2

#define PERIOD  2000  //neu in dieser BugFix Version!

//#define USE_DRIVER
```

```
//SoftwareSerial debugSerial(11, 12);

// initialize the stepper library on pins 8 through 11:
#ifdef USE_DRIVER
AccelStepper stepper(AccelStepper::DRIVER, PIN_DRIVER_STEP, PIN_DRIVER_DIR);
#else
AccelStepper stepper(AccelStepper::FULL4WIRE, 6, 4, 5, 3, false);
#endif

// temperature
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// multiplier of SPEEDMUX, currently max speed is 480.
int speedFactor = 16;
int speedFactorRaw = 4;
int speedMult = 30;

float t_coeff = 0;

// button
long currentPosition = 0;
long targetPosition = 0;
long lastSavedPosition = 0;
long millisLastMove = 0;
const long millisDisableDelay = 15000;
bool isRunning = false;

int turbo_multip = 1;  //multiplier basic value for turbo switch if set to ON

// read commands
bool eoc = false;
```

```
String line;

void setup() {
  Serial.begin(9600);
 //debugSerial.begin(9600);

  // initalize motor
  stepper.setMaxSpeed(speedFactor * speedMult);
  stepper.setAcceleration(100);
  #ifdef USE_DRIVER
  stepper.setEnablePin(PIN_DRIVER_ENABLE);
  #endif
  millisLastMove = millis();

  // read saved position from EEPROM
 // EEPROM.put(0, (long)0);  //war auskommentiert!!!
  EEPROM.get(0, currentPosition);

  // prevent negative values if EEPROM is empty //neu in dieser BugFix Version!
  currentPosition = max(0, currentPosition); //neu in dieser Bug Fix Version!

  stepper.setCurrentPosition(currentPosition);
  lastSavedPosition = currentPosition;
//  debugSerial.print("Load last position from EEPROM...");
//  debugSerial.println(lastSavedPosition);

  // init temperature sensor
  sensors.begin();

  // setup buttons
  pinMode(BTN_IN, INPUT_PULLUP);
  pinMode(BTN_OUT, INPUT_PULLUP);
```

```
  pinMode(BTN_TURBS, INPUT_PULLUP);

//folgende 3 Zeilen neu in dieser BugFix Version
// init timer
  Timer1.initialize(PERIOD);
  Timer1.attachInterrupt(intHandler);
}

void loop() {

  // process the command we got
  if (eoc) {
   //serial.println(line);//entfernt in dieser BugFix Version

  //neu in dieser BugFix Version
    if(line.startsWith("2")){
      //debugSerial.println("Got Dual focuser command(?) starting with 2. Send values of first motor");
      // remove first character and parse command
      line = line.substring(1);
    }

    String cmd, param;
    int len = line.length();
    if (len >= 2) {
      cmd = line.substring(0, 2);
    }
    if (len > 2) {
      param = line.substring(2);
    }

   // debugSerial.print(cmd);
   // debugSerial.print(":");
```

```
// debugSerial.println(param);
// debugSerial.println(line);

 line = "";
 eoc = false;

 // LED backlight value, always return "00"
 if (cmd.equalsIgnoreCase("GB")) {
  Serial.print("00#");
 }
 // home the motor, hard-coded, ignore parameters since we only have one motor
 if (cmd.equalsIgnoreCase("PH")) {
  stepper.setCurrentPosition(8000);
  stepper.moveTo(0);
  isRunning = true;
 }
 // firmware value, always return "10"
 if (cmd.equalsIgnoreCase("GV")) {
  Serial.print("10#");
 }
 // get the current motor position
 if (cmd.equalsIgnoreCase("GP")) {
  currentPosition = stepper.currentPosition();
  char tempString[6];
  sprintf(tempString, "%04X", currentPosition);
  Serial.print(tempString);
  Serial.print("#");

 //   debugSerial.print("current motor position: 0x");
 //  debugSerial.print(tempString);
 //  debugSerial.print(" = ");
 // debugSerial.println(currentPosition);
```

```
 }
 // get the new motor position (target)
 if (cmd.equalsIgnoreCase("GN")) {
  //pos = stepper.targetPosition();
  char tempString[6];
  sprintf(tempString, "%04X", targetPosition);
  Serial.print(tempString);
  Serial.print("#");

//   debugSerial.print("target motor position: ");
//   debugSerial.println(tempString);
 }
 // get the current temperature from DS1820 temperature sensor
 if (cmd.equalsIgnoreCase("GT")) {
  sensors.requestTemperatures();
  float temperature = sensors.getTempCByIndex(0);
  //debugSerial.print("temperature: ");
  //debugSerial.println(temperature);
  if(temperature > 100 || temperature < -50){
    // error
    temperature = 0;
  }
  byte t_int = (byte)temperature << 1;
  t_int += round(temperature - (byte) temperature);
  Serial.print(t_int, HEX);
  Serial.print('#');
 }

 // get the temperature coefficient
 if (cmd.equalsIgnoreCase("GC")) {
  Serial.print("02#");
  Serial.print((byte)t_coeff, HEX);
```

```
  Serial.print('#');
 }

 // set the temperature coefficient
 if (cmd.equalsIgnoreCase("SC")) {
  //debugSerial.println(param);
  if(param.length() > 4){
    param = param.substring(param.length()-4);
  }
//   debugSerial.println(param);

  if(param.startsWith("F")){
 //    debugSerial.println("negative");
 //   debugSerial.println(strtol("FFFF", NULL, 16));
 //  debugSerial.println(strtol(param.c_str(), NULL, 16));
    t_coeff = ((0xFFFF - strtol(param.c_str(), NULL, 16)) / -2.0f) -0.5f;
  } else {
    t_coeff = strtol(param.c_str(), NULL, 16) / 2.0f;
  }
 //  debugSerial.print("t_coeff: ");
 //  debugSerial.println(t_coeff);
 //  Serial.print("02#"); //entfernt in dieser BugFix Version!
 }

 // get the current motor speed, only values of 02, 04, 08, 10, 20
 if (cmd.equalsIgnoreCase("GD")) {
  char tempString[6];
  sprintf(tempString, "%02X", speedFactorRaw);
  Serial.print(tempString);
  Serial.print("#");

//   debugSerial.print("current motor speed: ");
```

```
//  debugSerial.println(tempString);
}

// set speed, only acceptable values are 02, 04, 08, 10, 20
if (cmd.equalsIgnoreCase("SD")) {
  speedFactorRaw = hexstr2long(param);

  // SpeedFactor: smaller value means faster
  speedFactor = 32 / speedFactorRaw;
  stepper.setMaxSpeed(speedFactor * speedMult); //speed multiplier set to 1 or 256
}

// whether half-step is enabled or not, always return "00"
if (cmd.equalsIgnoreCase("GH")) {
  Serial.print("00#");
}

// motor is moving - 01 if moving, 00 otherwise
if (cmd.equalsIgnoreCase("GI")) {
  if(abs(targetPosition - currentPosition) > 0){
    Serial.print("01#");
  } else {
    Serial.print("00#");
  }
}
// set current motor position
if (cmd.equalsIgnoreCase("SP")) {
  currentPosition = hexstr2long(param); // entfernt in dieser BugFix Version: * 16;
  stepper.setCurrentPosition(currentPosition);
}
// set new motor position
if (cmd.equalsIgnoreCase("SN")) {
```

```
    Serial.println(param);
   //  debugSerial.print("new target position ");
   // debugSerial.print(targetPosition);
   //  debugSerial.print(" -> ");
    targetPosition = hexstr2long(param); // entfernt in dieser BugFix Version!: * 16;
   //  debugSerial.println(targetPosition);
    //Serial.println(targetPosition); entfernt in dieser BugFix Version!
    //stepper.moveTo(pos);
   }
   // initiate a move
   if (cmd.equalsIgnoreCase("FG")) {
    isRunning = 1;
//     running = true;
    stepper.enableOutputs();
    stepper.moveTo(targetPosition);
   }
   // stop a move
   if (cmd.equalsIgnoreCase("FQ")) {
   //  isRunning = 0;  folgende 2 Zeilen auch entfernt in dieser BugFix Version
   //  stepper.moveTo(stepper.currentPosition());
   //   stepper.run();
//     running = false;
    stepper.stop();
   }
  }

  int btn_in = digitalRead(BTN_IN);
  int btn_out = digitalRead(BTN_OUT);
  int btn_turbs = digitalRead(BTN_TURBS); //read position of turbo switch ON or OFF

  // move motor if not done
  if (stepper.distanceToGo() != 0) {
```

```
  isRunning = true;
  millisLastMove = millis();
  //stepper.run();  entfernt in dieser BugFix version!
}
// handle manual buttons
else if( btn_in == LOW || btn_out == LOW ) {
    stepper.enableOutputs();
    while (btn_in == LOW || btn_out == LOW) {

        //für SVBony Turboswitch festgesetzt
        //btn_turbs = LOW;

        if (btn_turbs == LOW) {     //if turbo switch is set ON = LOW
          turbo_multip = 15;       //set multiplier to 256
        } else { turbo_multip = 1;  // ...or to 1
        }

        if (btn_in == LOW) {
            stepper.move(BTN_STEP * turbo_multip);
        } else {
            stepper.move(-BTN_STEP * turbo_multip);
        }
        stepper.runSpeedToPosition();
        btn_in = digitalRead(BTN_IN);
        btn_out = digitalRead(BTN_OUT);
        btn_turbs = digitalRead(BTN_TURBS);
    }
    stepper.stop();
    stepper.disableOutputs();
    millisLastMove = millis();
    currentPosition = stepper.currentPosition();
}  else {
```

```
    isRunning = false;
    if(millis() - millisLastMove > millisDisableDelay){
      // Save current location in EEPROM
      if (lastSavedPosition != currentPosition) {
        EEPROM.put(0, currentPosition); //aktiviert in dieser BugFix Version!
        lastSavedPosition = currentPosition;
      //  debugSerial.println("Save last position to EEPROM");
        stepper.disableOutputs();
     //   debugSerial.println("Disabled output pins");
      }
    }
  }

  //delay(20);
}

// read the command until the terminating # character
void serialEvent () {
  // read the command until the terminating # character
  while (Serial.available() && !eoc) {
    char c = Serial.read();
    if (c != '#' && c != ':') {
      line = line + c;
    }
    else {
      if (c == '#') {
        eoc = true;
      }
    }
  }
}
```

```
long hexstr2long(String line) {
  char buf[line.length()+1];
  line.toCharArray(buf, line.length()+1);
  long ret = 0;

  ret = strtol(buf, NULL, 16);
  return ret;
}

//folgendes neu in dieser BugFix Version
static void intHandler() {
      stepper.run();
}
```