```
/************************************************************
                  CMPS11 Compass driver
               Arduino UNO & Compass CMPS11
                 CMPS11 running I2C mode
        RS2323 MAX3323 Feeding NMEA to NMEA-Network
  Ver 3 2017-11-26 /A. Walther based on the sketch from Hakan
************************************************************/
#include <Wire.h>                                          // To drive I2C bus to CMPS11
#include <SoftwareSerial.h>                                // To drive the extra serial port for RS232 to NMEA-Netw.
#define ADDRESS 0x60                                       // Defines address of CMPS11 the address specified in the data sheet is 192
(0xC0) but i2c addressing uses the high 7 bits so it's 96(0x60)
#define rxPin 2                                            // Defines the pins for the in-/output to NMEA-Netw.
#define txPin 3
#define buttonPin 5                                        // Defines the pin for HDG / HDM switch
#define calSwitch 7                                        // Defines the pin for the Calibration Switch
#define calStore 6                                         // Defines the pin for the Store Switch
#define ledPin 4                                           // Defines the Pin for the LED

SoftwareSerial NMEASerial = SoftwareSerial(rxPin, txPin);          // RX, TX to NMEA-Netw.
int HDGKorrFactor = 0;                                     // offset correction in degrees if needed. -for westerly and +for easterly
offset
String MagBer;
int buttonState = 0;                                       // variable for reading the status of the HDG / HDM switch
int stcalSwitch = 0;                                       // variable for reading the Status of the Calibration Switch
int stcalStore = 0;                                        // variable for reading the Status of the Store Switch

void setup() {
  pinMode(rxPin, INPUT);                                   // Define the i/o pins for the RS232
  pinMode(txPin, OUTPUT);
  pinMode(buttonPin, INPUT);                               // Define the pin for the HDG / HDM switch as input
  pinMode(calSwitch, INPUT);                               // Define the pin for the Cal. Switch as input
  pinMode(calStore, INPUT);                                // Define the pin for the Store Switch as input
  pinMode(ledPin, OUTPUT);                                 // Define the LED Pin as output
  Wire.begin();                                            // Conects I2C (to CMPS11)
  Serial.begin(9600);                                      // To USB (for monitoring the data only)
  delay(50);
  NMEASerial.begin(4800);                                  // To RS232 with 4800 baud
}

void loop() {
  stcalSwitch = digitalRead(calSwitch);                    // reads the status of the Calibration Switch
```

```
  if (stcalSwitch == HIGH) {                                  // if the Calibration Switch is pressed jump to subroutine calibration
    calibrate();
  }
  byte highByte, lowByte, fine;                               // highByte and lowByte store high and low bytes of the bearing and fine
stores decimal place of bearing
  char pitch, roll;                                           // Stores pitch and roll values of CMPS11, chars are used because they
support signed value
  int bearing;
  Wire.beginTransmission(ADDRESS);                             // starts communication with CMPS11
  Wire.write(2);                                               // Sends the register we wish to start reading from
 Wire.endTransmission();
  Wire.requestFrom(ADDRESS, 4);                                // Request 4 bytes from CMPS10 e.g. register 2, 3 ,4, 5
  while (Wire.available() < 4);                                // Wait for bytes to become available
  highByte = Wire.read();
  lowByte = Wire.read();
  pitch = Wire.read();
  roll = Wire.read();
  bearing = ((highByte << 8) + lowByte) / 10;                  // Calculate full bearing
  bearing += HDGKorrFactor;                                    // Correction
  fine = ((highByte << 8) + lowByte) % 10;                     // Calculate decimal place of bearing
  if ((fine) > 5 ) bearing += 1;                               // Don't display comma but do a proper round off
  if ((bearing) >= 360 ) bearing += -360;                      // Correction if Bearing >= 360
  if ((bearing) < 0 ) bearing += 360;                          // Correction if Bearing < 0
  MagBer = String(bearing, DEC);
  //MagBer += '.';                                             // Rounding off earlier Instead
  //MagBer += String(fine, DEC);                               // Left uncommend end commend the upper lines for heading with value behind
comma
  buttonState = digitalRead(buttonPin);                        // read the state of the HDG/HDM switch
  if (buttonState == HIGH) {                                   // display HDM if switch is connected
  display_data(bearing, fine, pitch, roll, NMEA_HDM(MagBer));  // NMEA_HDM
  delay(500);                                                  // Keep uneven to avoid crash in IS12??? Achtung!! zum Anpassen auf 4 Hz
evt. auf 250 ändern!!
  } else {                                                     // display HDG if switch is disconnected
  display_data(bearing, fine, pitch, roll, NMEA_HDG(MagBer));  // NMEA_HDG
  delay(500);
  }
}

void display_data(int b, int f, int p, int r, String n) {      // pitch and roll (p, r) are received as int instead of bytes so that they
will display corectly as signed values.
  //Serial.print("Bearing = ");                                  // Display the full bearing and fine bearing separated by a decimal point
```

```
on the seriell Monitor
  //Serial.print(b);                                                    // left uncommend what needed eg. for monitoring the values for pitch and
roll
  //Serial.print(".");
  //Serial.print(f);
  //Serial.print(" ");
  //Serial.print("Pitch = ");
  //Serial.print(p);
  //Serial.print(" ");
  //Serial.print("Roll = ");
  //Serial.println(r);
   Serial.print(n);                                                   // Print NMEA string to USB
   delay (45);
   NMEASerial.print(n);                                               // Print NMEA String to NMEA-Network
}

//Create NMEA String: $HCHDM,238.5,M*hh/CR/LF
String NMEA_HDM(String Mag) {
  String nmea = "$HCHDM,";
  nmea += Mag, DEC;
  nmea += ',';
  nmea += 'M';
  nmea += '*';
  nmea += String(testsum(nmea), HEX);
  nmea += '\r';
  nmea += '\n';
  return nmea;
}

//Create the NMEA string: $HCHDG,238.5,Dev_Gr,Dev_E,Var_Gr,Var_E*hh/CR/LF
String NMEA_HDG(String Mag) {
  String nmea = "$HCHDG,";
  nmea += Mag, DEC;
  nmea += ',';
  nmea += ',';
  nmea += ',';
  //nmea += "2.8";
  nmea += ',';
  //nmea += 'E';
  nmea += '*';
  nmea += String(testsum(nmea), HEX);
```

```
  nmea += '\r';
  nmea += '\n';
  return nmea;
}

// Calculates the Checksum for the NMEA string
int testsum(String strN) {
  int i;
  int XOR;
  int c;
  // Calculate testsum ignoring any $'s in the string
  for (XOR = 0, i = 0; i < 80; i++) {                                    // strlen(strN)
    c = (unsigned char)strN[i];
    if (c == '*') break;
    if (c != '$') XOR ^= c;
  }
  return XOR;
}

int soft_ver() {
  int data;                                                             // Software version of CMPS11 is read into data and then returned
  Wire.beginTransmission(ADDRESS);                                      // Values of 0 being sent with write need to be masked as a byte so
they are not misinterpreted as NULL this is a bug in arduino 1.0
  Wire.write((byte)0);                                                  // Sends the register we wish to start reading from
 Wire.endTransmission();
  Wire.requestFrom(ADDRESS, 1);                                         // Request byte from CMPS11
  while (Wire.available() < 1);
  data = Wire.read();
  return (data);
}

void calibrate() {                                                      // Subroutine for Calibration Mode
  stcalSwitch = digitalRead(calSwitch);                                 // Read Calibration Switch
  stcalStore = digitalRead(calStore);                                   // Read Store Switch
  if (stcalSwitch == HIGH && stcalStore == HIGH) {                      // If the two switches pressed together do factory reset and flashes
the LED
    for (int zaehlera=1; zaehlera<20 ; zaehlera = zaehlera+1){
    digitalWrite(ledPin, HIGH);
    delay (75);
    digitalWrite(ledPin, LOW);
    delay (75);
```

```
  }
 Wire.beginTransmission(ADDRESS);                    // Send three blocks with the factory reset sequence to CMPS11
 Wire.write(0);
 Wire.write(0x20);
 Wire.endTransmission();
 delay(25);

Wire.beginTransmission(ADDRESS);
 Wire.write(0);
 Wire.write(0x2A);
 Wire.endTransmission();
 delay(25);

Wire.beginTransmission(ADDRESS);
 Wire.write(0);
 Wire.write(0x60);
 Wire.endTransmission();
 delay(25);

} else {                                             // Enter Calibration Mode send the calibration sequence to CMPS11
 digitalWrite(ledPin, HIGH);                         // let the LED lights for calibration prozess
Wire.beginTransmission(ADDRESS);
 Wire.write(0);
 Wire.write(0xF0);
 Wire.endTransmission();
 delay(25);

Wire.beginTransmission(ADDRESS);
 Wire.write(0);
 Wire.write(0xF5);
 Wire.endTransmission();
 delay(25);

Wire.beginTransmission(ADDRESS);
 Wire.write(0);
 Wire.write(0xF7);                                   // use 0xF6 instead 0xF7 for full three-axis calibration
 Wire.endTransmission();                             // otherwise it is calibration in the horizontal plane
 delay(25);

 for (int zaehlerb=1; zaehlerb<600; zaehlerb = zaehlerb+1){          // Calibration Mode for 600 sek
  stcalStore = digitalRead(calStore);
```

```
    if (stcalStore == HIGH) {                                          // Check if Store Swtich is pressed, if so then store the calibration
and left calibration mode
      for (int zaehlerc=1; zaehlerc<20; zaehlerc = zaehlerc+1){         // and left the LED blinks
      digitalWrite(ledPin, HIGH);
      delay (100);
      digitalWrite(ledPin, LOW);
      delay (100);
       }
      Wire.beginTransmission(ADDRESS);                                 // Send the calibration store sentence
      Wire.write(0);
      Wire.write(0xF8);
      Wire.endTransmission();
      delay(25);

      return calibrate;
    } else {                                                           // makes that the value of zaehler b counts in seconds
      delay (1000);
       }
    }
  for (int zaehlerd=1; zaehlerd<40; zaehlerd = zaehlerd+1){            // automatic end of calibration mode with long time blinking LED
      digitalWrite(ledPin, HIGH);
      delay (100);
      digitalWrite(ledPin, LOW);
      delay (100);
       }
   Wire.beginTransmission(ADDRESS);                                    // Send the calibration store sentence
   Wire.write(0);
   Wire.write(0xF8);
   Wire.endTransmission();
   delay(25);
    }
}
```