/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This is our GPS library

Adafruit invests time and resources providing this open source code,

please support Adafruit and open-source hardware by purchasing

products from Adafruit!

Written by Limor Fried/Ladyada for Adafruit Industries.

BSD license, check license.txt for more information

All text above must be included in any redistribution

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <Adafruit\_GPS.h>

// how long are max NMEA lines to parse?

#define MAXLINELENGTH 120

// we double buffer: read one line in and leave one for the main program

volatile char line1[MAXLINELENGTH];

volatile char line2[MAXLINELENGTH];

// our index into filling the current line

volatile uint8\_t lineidx=0;

// pointers to the double buffers

volatile char \*currentline;

volatile char \*lastline;

volatile boolean recvdflag;

volatile boolean inStandbyMode;

boolean Adafruit\_GPS::parse(char \*nmea) {

// do checksum check

// first look if we even have one

if (nmea[strlen(nmea)-4] == '\*') {

uint16\_t sum = parseHex(nmea[strlen(nmea)-3]) \* 16;

sum += parseHex(nmea[strlen(nmea)-2]);

// check checksum

for (uint8\_t i=1; i < (strlen(nmea)-4); i++) {

sum ^= nmea[i];

}

if (sum != 0) {

// bad checksum :(

//return false;

}

}

// look for a few common sentences

if (strstr(nmea, "$GPGGA")) {

// found GGA

char \*p = nmea;

// get time

p = strchr(p, ',')+1;

float timef = atof(p);

uint32\_t time = timef;

hour = time / 10000;

minute = (time % 10000) / 100;

seconds = (time % 100);

milliseconds = fmod(timef, 1.0) \* 1000;

// parse out latitude

p = strchr(p, ',')+1;

latitude = atof(p);

p = strchr(p, ',')+1;

if (p[0] == 'N') lat = 'N';

else if (p[0] == 'S') lat = 'S';

else if (p[0] == ',') lat = 0;

else return false;

// parse out longitude

p = strchr(p, ',')+1;

longitude = atof(p);

p = strchr(p, ',')+1;

if (p[0] == 'W') lon = 'W';

else if (p[0] == 'E') lon = 'E';

else if (p[0] == ',') lon = 0;

else return false;

p = strchr(p, ',')+1;

fixquality = atoi(p);

p = strchr(p, ',')+1;

satellites = atoi(p);

p = strchr(p, ',')+1;

HDOP = atof(p);

p = strchr(p, ',')+1;

altitude = atof(p);

p = strchr(p, ',')+1;

p = strchr(p, ',')+1;

geoidheight = atof(p);

return true;

}

if (strstr(nmea, "$GPRMC")) {

// found RMC

char \*p = nmea;

// get time

p = strchr(p, ',')+1;

float timef = atof(p);

uint32\_t time = timef;

hour = time / 10000;

minute = (time % 10000) / 100;

seconds = (time % 100);

milliseconds = fmod(timef, 1.0) \* 1000;

p = strchr(p, ',')+1;

// Serial.println(p);

if (p[0] == 'A')

fix = true;

else if (p[0] == 'V')

fix = false;

else

return false;

// parse out latitude

p = strchr(p, ',')+1;

latitude = atof(p);

p = strchr(p, ',')+1;

if (p[0] == 'N') lat = 'N';

else if (p[0] == 'S') lat = 'S';

else if (p[0] == ',') lat = 0;

else return false;

// parse out longitude

p = strchr(p, ',')+1;

longitude = atof(p);

p = strchr(p, ',')+1;

if (p[0] == 'W') lon = 'W';

else if (p[0] == 'E') lon = 'E';

else if (p[0] == ',') lon = 0;

else return false;

// speed

p = strchr(p, ',')+1;

speed = atof(p);

// angle

p = strchr(p, ',')+1;

angle = atof(p);

p = strchr(p, ',')+1;

uint32\_t fulldate = atof(p);

day = fulldate / 10000;

month = (fulldate % 10000) / 100;

year = (fulldate % 100);

// we dont parse the remaining, yet!

return true;

}

return false;

}

char Adafruit\_GPS::read(void) {

char c = 0;

if (paused) return c;

#ifdef \_\_AVR\_\_

if(gpsSwSerial) {

if(!gpsSwSerial->available()) return c;

c = gpsSwSerial->read();

}

else

{

if(!gpsHwSerial->available()) return c;

c = gpsHwSerial->read();

}

#else

// if(!gpsHwSerial->available()) return c;

// c = gpsHwSerial->read();

if(!Serial1.available()) return c;

c = Serial1.read();

#endif

//Serial.print(c);

if (c == '$') {

currentline[lineidx] = 0;

lineidx = 0;

}

if (c == '\n') {

currentline[lineidx] = 0;

if (currentline == line1) {

currentline = line2;

lastline = line1;

} else {

currentline = line1;

lastline = line2;

}

//Serial.println("----");

//Serial.println((char \*)lastline);

//Serial.println("----");

lineidx = 0;

recvdflag = true;

}

currentline[lineidx++] = c;

if (lineidx >= MAXLINELENGTH)

lineidx = MAXLINELENGTH-1;

return c;

}

#ifdef \_\_AVR\_\_

// Constructor when using SoftwareSerial or NewSoftSerial

#if ARDUINO >= 100

Adafruit\_GPS::Adafruit\_GPS(SoftwareSerial \*ser)

#else

Adafruit\_GPS::Adafruit\_GPS(NewSoftSerial \*ser)

#endif

{

common\_init(); // Set everything to common state, then...

gpsSwSerial = ser; // ...override gpsSwSerial with value passed.

}

#endif

Adafruit\_GPS::Adafruit\_GPS(HardwareSerial \*ser) {

common\_init(); // Set everything to common state, then...

gpsHwSerial = ser; // ...override gpsHwSerial with value passed.

}

// Initialization code used by all constructor types

void Adafruit\_GPS::common\_init(void) {

recvdflag = false;

paused = false;

lineidx = 0;

currentline = line1;

lastline = line2;

hour = minute = seconds = year = month = day =

fixquality = satellites = 0; // uint8\_t

lat = lon = mag = 0; // char

fix = false; // boolean

milliseconds = 0; // uint16\_t

latitude = longitude = geoidheight = altitude =

speed = angle = magvariation = HDOP = 0.0; // float

}

void Adafruit\_GPS::begin(uint16\_t baud)

{

#ifdef \_\_AVR\_\_

if(gpsSwSerial)

gpsSwSerial->begin(baud);

else

gpsHwSerial->begin(baud);

#else

// gpsHwSerial->begin(baud);

Serial1.begin(baud);

#endif

delay(10);

}

void Adafruit\_GPS::sendCommand(char \*str) {

#ifdef \_\_AVR\_\_

if(gpsSwSerial)

gpsSwSerial->println(str);

else

gpsHwSerial->println(str);

#else

// gpsHwSerial->println(str);

Serial1.println(str);

#endif

}

boolean Adafruit\_GPS::newNMEAreceived(void) {

return recvdflag;

}

void Adafruit\_GPS::pause(boolean p) {

paused = p;

}

char \*Adafruit\_GPS::lastNMEA(void) {

recvdflag = false;

return (char \*)lastline;

}

// read a Hex value and return the decimal equivalent

uint8\_t Adafruit\_GPS::parseHex(char c) {

if (c < '0')

return 0;

if (c <= '9')

return c - '0';

if (c < 'A')

return 0;

if (c <= 'F')

return (c - 'A')+10;

}

boolean Adafruit\_GPS::waitForSentence(char \*wait4me, uint8\_t max) {

char str[20];

uint8\_t i=0;

while (i < max) {

if (newNMEAreceived()) {

char \*nmea = lastNMEA();

strncpy(str, nmea, 20);

str[19] = 0;

i++;

if (strstr(str, wait4me))

return true;

}

}

return false;

}

boolean Adafruit\_GPS::LOCUS\_StartLogger(void) {

sendCommand(PMTK\_LOCUS\_STARTLOG);

recvdflag = false;

return waitForSentence(PMTK\_LOCUS\_LOGSTARTED);

}

boolean Adafruit\_GPS::LOCUS\_ReadStatus(void) {

sendCommand(PMTK\_LOCUS\_QUERY\_STATUS);

if (! waitForSentence("$PMTKLOG"))

return false;

char \*response = lastNMEA();

uint16\_t parsed[10];

uint8\_t i;

for (i=0; i<10; i++) parsed[i] = -1;

response = strchr(response, ',');

for (i=0; i<10; i++) {

if (!response || (response[0] == 0) || (response[0] == '\*'))

break;

response++;

parsed[i]=0;

while ((response[0] != ',') &&

(response[0] != '\*') && (response[0] != 0)) {

parsed[i] \*= 10;

char c = response[0];

if (isDigit(c))

parsed[i] += c - '0';

else

parsed[i] = c;

response++;

}

}

LOCUS\_serial = parsed[0];

LOCUS\_type = parsed[1];

if (isAlpha(parsed[2])) {

parsed[2] = parsed[2] - 'a' + 10;

}

LOCUS\_mode = parsed[2];

LOCUS\_config = parsed[3];

LOCUS\_interval = parsed[4];

LOCUS\_distance = parsed[5];

LOCUS\_speed = parsed[6];

LOCUS\_status = !parsed[7];

LOCUS\_records = parsed[8];

LOCUS\_percent = parsed[9];

return true;

}

// Standby Mode Switches

boolean Adafruit\_GPS::standby(void) {

if (inStandbyMode) {

return false; // Returns false if already in standby mode, so that you do not wake it up by sending commands to GPS

}

else {

inStandbyMode = true;

sendCommand(PMTK\_STANDBY);

//return waitForSentence(PMTK\_STANDBY\_SUCCESS); // don't seem to be fast enough to catch the message, or something else just is not working

return true;

}

}

boolean Adafruit\_GPS::wakeup(void) {

if (inStandbyMode) {

inStandbyMode = false;

sendCommand(""); // send byte to wake it up

return waitForSentence(PMTK\_AWAKE);

}

else {

return false; // Returns false if not in standby mode, nothing to wakeup

}

}