Arduino Naza Decoder Driver

Generated by Doxygen 1.8.9.1

Tue Dec 29 2015 15:12:54

Contents

1 Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

NazaDecoder	**2**
NazaDecoder::VersionSchemeType	**9**
NazaDecoder::VersionType	**10**

2 File Index

2.1 File List

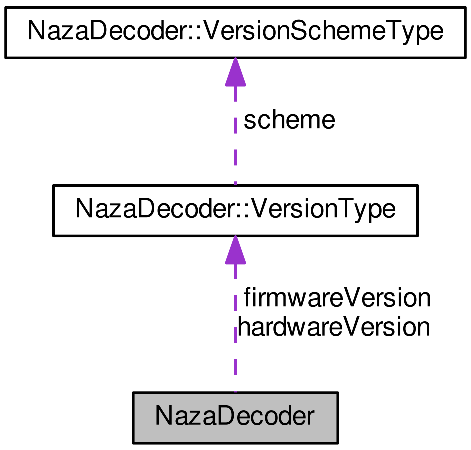
Here is a list of all files with brief descriptions:

3 Class Documentation

3.1 NazaDecoder Class Reference

```
#include <NazaDecoder.h>
```

Collaboration diagram for NazaDecoder:



Classes

- struct VersionSchemeType
- union VersionType

Public Types

- enum GPSPayloadPosition {
NAZA_MESSAGE_POS_DT = 0x04 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_LO = 0x08 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_LA = 0x0c - MESSAGE_HEADER_SIZE, NAZA_M←ESSAGE_POS_AL = 0x10 - MESSAGE_HEADER_SIZE,
NAZA_MESSAGE_POS_HA = 0x14 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_VA = 0x18 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_NV = 0x20 - MESSAGE_HEADER_SIZE, NAZA_←MESSAGE_POS_EV = 0x24 - MESSAGE_HEADER_SIZE,
NAZA_MESSAGE_POS_DV = 0x28 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_PD = 0x2c - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_VD = 0x2e - MESSAGE_HEADER_SIZE, NAZA_←MESSAGE_POS_ND = 0x30 - MESSAGE_HEADER_SIZE,
NAZA_MESSAGE_POS_ED = 0x32 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_NS = 0x34 -

MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_FT = 0x36 - MESSAGE_HEADER_SIZE, NAZA_M↩ESSAGE_POS_SF = 0x38 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_XM = 0x3b - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_SN = 0x3c - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_CS = 0x3e - MESSAGE_HEADER_SIZE }

- enum CompassPayloadPosition { NAZA_MESSAGE_POS_CX = 0x04 - MESSAGE_HEADER_SIZE, NAZ↩A_MESSAGE_POS_CY = 0x06 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_CZ = 0x08 - ME↩SSAGE_HEADER_SIZE }
- enum ModuleVersionPayloadPosition { NAZA_MESSAGE_POS_FW = 0x08 - MESSAGE_HEADER_SIZE, NAZA_MESSAGE_POS_HW = 0x0c - MESSAGE_HEADER_SIZE }
- enum MessageType { NAZA_MESSAGE_NONE_TYPE = 0x00, NAZA_MESSAGE_GPS_TYPE = 0x10, N↩AZA_MESSAGE_COMPASS_TYPE = 0x20, NAZA_MESSAGE_MODULE_VERSION_TYPE = 0x30 }
- enum MessageSize { NAZA_MESSAGE_GPS_SIZE = 0x3a, NAZA_MESSAGE_COMPASS_SIZE = 0x06, NAZA_MESSAGE_MODULE_VERSION_SIZE = 0x0c }
- enum FixType { NO_FIX = 0, FIX_2D = 2, FIX_3D = 3, FIX_DGPS = 4 }

Public Member Functions

- NazaDecoder ()
- uint8_t decode (int16_t input)
- double getLat ()
- double getLon ()
- double getGpsAlt ()
- double getSpeed ()
- FixType getFixType ()
- uint8_t getNumSat ()
- double getHeading ()
- double getCog ()
- double getGpsVsi ()
- double getHdop ()
- double getVdop ()
- uint8_t getYear ()
- uint8_t getMonth ()
- uint8_t getDay ()
- uint8_t getHour ()
- uint8_t getMinute ()
- uint8_t getSecond ()
- VersionType getFirmwareVersion ()
- VersionType getHardwareVersion ()
- uint8_t isLocked ()

Private Member Functions

- int32_t pack4 (uint8_t i, uint8_t mask)
- int16_t pack2 (uint8_t i, uint8_t mask)
- void updateChecksum (int16_t input)

Private Attributes

- int16_t payload [58]
- int16_t seq
- int16_t cnt
- int16_t msgId
- int16_t msgLen
- uint8_t cs1

- uint8_t cs2
- int16_t magXMin
- int16_t magXMax
- int16_t magYMin
- int16_t magYMax
- double lon
- double lat
- double gpsAlt
- double spd
- FixType fix
- uint8_t sat
- double heading
- double cog
- double gpsVsi
- double hdop
- double vdop
- uint8_t year
- uint8_t month
- uint8_t day
- uint8_t hour
- uint8_t minute
- uint8_t second
- VersionType firmwareVersion
- VersionType hardwareVersion
- uint16_t lastLock
- uint8_t locked

3.1.1 Detailed Description

Definition at line 14 of file NazaDecoder.h.

3.1.2 Member Enumeration Documentation

3.1.2.1 enum NazaDecoder::CompassPayloadPosition

Enumerator

NAZA_MESSAGE_POS_CX

NAZA_MESSAGE_POS_CY

NAZA_MESSAGE_POS_CZ

Definition at line 78 of file NazaDecoder.h.

3.1.2.2 enum NazaDecoder::FixType

Enumerator

NO_FIX

FIX_2D

FIX_3D

FIX_DGPS

Definition at line 112 of file NazaDecoder.h.

3.1.2.3 enum NazaDecoder::GPSPayloadPosition

Enumerator

- ***NAZA_MESSAGE_POS_DT***
- ***NAZA_MESSAGE_POS_LO***
- ***NAZA_MESSAGE_POS_LA***
- ***NAZA_MESSAGE_POS_AL***
- ***NAZA_MESSAGE_POS_HA***
- ***NAZA_MESSAGE_POS_VA***
- ***NAZA_MESSAGE_POS_NV***
- ***NAZA_MESSAGE_POS_EV***
- ***NAZA_MESSAGE_POS_DV***
- ***NAZA_MESSAGE_POS_PD***
- ***NAZA_MESSAGE_POS_VD***
- ***NAZA_MESSAGE_POS_ND***
- ***NAZA_MESSAGE_POS_ED***
- ***NAZA_MESSAGE_POS_NS***
- ***NAZA_MESSAGE_POS_FT***
- ***NAZA_MESSAGE_POS_SF***
- ***NAZA_MESSAGE_POS_XM***
- ***NAZA_MESSAGE_POS_SN***
- ***NAZA_MESSAGE_POS_CS***

Definition at line 18 of file NazaDecoder.h.

3.1.2.4 enum NazaDecoder::MessageSize

Enumerator

- ***NAZA_MESSAGE_GPS_SIZE***
- ***NAZA_MESSAGE_COMPASS_SIZE***
- ***NAZA_MESSAGE_MODULE_VERSION_SIZE***

Definition at line 106 of file NazaDecoder.h.

3.1.2.5 enum NazaDecoder::MessageType

Enumerator

- ***NAZA_MESSAGE_NONE_TYPE***
- ***NAZA_MESSAGE_GPS_TYPE***
- ***NAZA_MESSAGE_COMPASS_TYPE***
- ***NAZA_MESSAGE_MODULE_VERSION_TYPE***

Definition at line 99 of file NazaDecoder.h.

3.1.2.6 enum NazaDecoder::ModuleVersionPayloadPosition

Enumerator

- ***NAZA_MESSAGE_POS_FW***
- ***NAZA_MESSAGE_POS_HW***

Definition at line 90 of file NazaDecoder.h.

3.1.3 Constructor & Destructor Documentation

3.1.3.1 NazaDecoder::NazaDecoder ( )

Definition at line 4 of file NazaDecoder.cpp.

3.1.4 Member Function Documentation

3.1.4.1 uint8_t NazaDecoder::decode (int16_t *input*)

Definition at line 92 of file NazaDecoder.cpp.

3.1.4.2 double NazaDecoder::getCog ( )

Definition at line 40 of file NazaDecoder.cpp.

3.1.4.3 uint8_t NazaDecoder::getDay ( )

Definition at line 64 of file NazaDecoder.cpp.

3.1.4.4 NazaDecoder::VersionType NazaDecoder::getFirmwareVersion ( )

Definition at line 80 of file NazaDecoder.cpp.

3.1.4.5 NazaDecoder::FixType NazaDecoder::getFixType ( )

Definition at line 28 of file NazaDecoder.cpp.

3.1.4.6 double NazaDecoder::getGpsAlt ( )

Definition at line 20 of file NazaDecoder.cpp.

3.1.4.7 double NazaDecoder::getGpsVsi ( )

Definition at line 44 of file NazaDecoder.cpp.

3.1.4.8 NazaDecoder::VersionType NazaDecoder::getHardwareVersion ( )

Definition at line 84 of file NazaDecoder.cpp.

3.1.4.9 double NazaDecoder::getHdop ( )

Definition at line 48 of file NazaDecoder.cpp.

3.1.4.10 double NazaDecoder::getHeading ( )

Definition at line 36 of file NazaDecoder.cpp.

3.1.4.11 uint8_t NazaDecoder::getHour ( )

Definition at line 68 of file NazaDecoder.cpp.

3.1.4.12 double NazaDecoder::getLat ( )

Definition at line 12 of file NazaDecoder.cpp.

3.1.4.13 double NazaDecoder::getLon ( )

Definition at line 16 of file NazaDecoder.cpp.

3.1.4.14 uint8_t NazaDecoder::getMinute ( )

Definition at line 72 of file NazaDecoder.cpp.

3.1.4.15 uint8_t NazaDecoder::getMonth ( )

Definition at line 60 of file NazaDecoder.cpp.

3.1.4.16 uint8_t NazaDecoder::getNumSat ( )

Definition at line 32 of file NazaDecoder.cpp.

3.1.4.17 uint8_t NazaDecoder::getSecond ( )

Definition at line 76 of file NazaDecoder.cpp.

3.1.4.18 double NazaDecoder::getSpeed ( )

Definition at line 24 of file NazaDecoder.cpp.

3.1.4.19 double NazaDecoder::getVdop ( )

Definition at line 52 of file NazaDecoder.cpp.

3.1.4.20 uint8_t NazaDecoder::getYear ( )

Definition at line 56 of file NazaDecoder.cpp.

3.1.4.21 uint8_t NazaDecoder::isLocked ( )

Definition at line 88 of file NazaDecoder.cpp.

3.1.4.22 int16_t NazaDecoder::pack2 (uint8_t *i*, uint8_t *mask*) `[private]`

Definition at line 240 of file NazaDecoder.cpp.

3.1.4.23 int32_t NazaDecoder::pack4 (uint8_t *i*, uint8_t *mask*) `[private]`

Definition at line 230 of file NazaDecoder.cpp.

3.1.4.24 void NazaDecoder::updateChecksum (int16_t *input*) `[private]`

Definition at line 7 of file NazaDecoder.cpp.

3.1.5 Member Data Documentation

3.1.5.1 int16_t NazaDecoder::cnt `[private]`

Definition at line 162 of file NazaDecoder.h.

3.1.5.2 double NazaDecoder::cog `[private]`

Definition at line 198 of file NazaDecoder.h.

3.1.5.3 uint8_t NazaDecoder::cs1 `[private]`

Definition at line 167 of file NazaDecoder.h.

3.1.5.4 uint8_t NazaDecoder::cs2 `[private]`

Definition at line 170 of file NazaDecoder.h.

3.1.5.5 uint8_t NazaDecoder::day [private]

Definition at line 210 of file NazaDecoder.h.

3.1.5.6 VersionType NazaDecoder::firmwareVersion [private]

Definition at line 215 of file NazaDecoder.h.

3.1.5.7 FixType NazaDecoder::fix [private]

Definition at line 189 of file NazaDecoder.h.

3.1.5.8 double NazaDecoder::gpsAlt [private]

Definition at line 183 of file NazaDecoder.h.

3.1.5.9 double NazaDecoder::gpsVsi [private]

Definition at line 201 of file NazaDecoder.h.

3.1.5.10 VersionType NazaDecoder::hardwareVersion [private]

Definition at line 216 of file NazaDecoder.h.

3.1.5.11 double NazaDecoder::hdop [private]

Definition at line 204 of file NazaDecoder.h.

3.1.5.12 double NazaDecoder::heading [private]

Definition at line 195 of file NazaDecoder.h.

3.1.5.13 uint8_t NazaDecoder::hour [private]

Definition at line 211 of file NazaDecoder.h.

3.1.5.14 uint16_t NazaDecoder::lastLock [private]

Definition at line 218 of file NazaDecoder.h.

3.1.5.15 double NazaDecoder::lat [private]

Definition at line 180 of file NazaDecoder.h.

3.1.5.16 uint8_t NazaDecoder::locked [private]

Definition at line 219 of file NazaDecoder.h.

3.1.5.17 double NazaDecoder::lon [private]

Definition at line 177 of file NazaDecoder.h.

3.1.5.18 int16_t NazaDecoder::magXMax [private]

Definition at line 172 of file NazaDecoder.h.

3.1.5.19 int16_t NazaDecoder::magXMin [private]

Definition at line 171 of file NazaDecoder.h.

3.1.5.20 int16_t NazaDecoder::magYMax [private]

Definition at line 174 of file NazaDecoder.h.

3.1.5.21 int16_t NazaDecoder::magYMin [private]

Definition at line 173 of file NazaDecoder.h.

3.1.5.22 uint8_t NazaDecoder::minute [private]

Definition at line 212 of file NazaDecoder.h.

3.1.5.23 uint8_t NazaDecoder::month [private]

Definition at line 209 of file NazaDecoder.h.

3.1.5.24 int16_t NazaDecoder::msgId [private]

Definition at line 163 of file NazaDecoder.h.

3.1.5.25 int16_t NazaDecoder::msgLen [private]

Definition at line 164 of file NazaDecoder.h.

3.1.5.26 int16_t NazaDecoder::payload[58] [private]

Definition at line 160 of file NazaDecoder.h.

3.1.5.27 uint8_t NazaDecoder::sat [private]

Definition at line 192 of file NazaDecoder.h.

3.1.5.28 uint8_t NazaDecoder::second [private]

Definition at line 213 of file NazaDecoder.h.

3.1.5.29 int16_t NazaDecoder::seq [private]

Definition at line 161 of file NazaDecoder.h.

3.1.5.30 double NazaDecoder::spd [private]

Definition at line 186 of file NazaDecoder.h.

3.1.5.31 double NazaDecoder::vdop [private]

Definition at line 207 of file NazaDecoder.h.

3.1.5.32 uint8_t NazaDecoder::year [private]

Definition at line 208 of file NazaDecoder.h.

The documentation for this class was generated from the following files:

- NazaDecoder.h
- NazaDecoder.cpp

3.2 NazaDecoder::VersionSchemeType Struct Reference

```
#include <NazaDecoder.h>
```

Public Attributes

- uint8_t revision
- uint8_t build
- uint8_t minor
- uint8_t major

3.2.1 Detailed Description

Definition at line 119 of file NazaDecoder.h.

3.2.2 Member Data Documentation

3.2.2.1 uint8_t NazaDecoder::VersionSchemeType::build

Definition at line 121 of file NazaDecoder.h.

3.2.2.2 uint8_t NazaDecoder::VersionSchemeType::major

Definition at line 123 of file NazaDecoder.h.

3.2.2.3 uint8_t NazaDecoder::VersionSchemeType::minor

Definition at line 122 of file NazaDecoder.h.

3.2.2.4 uint8_t NazaDecoder::VersionSchemeType::revision

Definition at line 120 of file NazaDecoder.h.

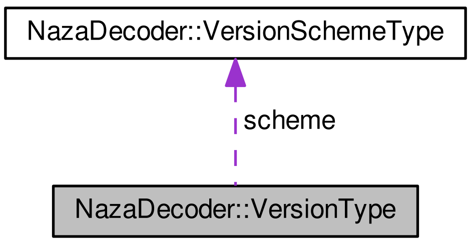
The documentation for this struct was generated from the following file:

- NazaDecoder.h

3.3 NazaDecoder::VersionType Union Reference

```
#include <NazaDecoder.h>
```

Collaboration diagram for NazaDecoder::VersionType:



Public Attributes

- uint32_t version

- VersionSchemeType scheme

3.3.1 Detailed Description

Definition at line 126 of file NazaDecoder.h.

3.3.2 Member Data Documentation

3.3.2.1 VersionSchemeType NazaDecoder::VersionType::scheme

Definition at line 128 of file NazaDecoder.h.

3.3.2.2 uint32_t NazaDecoder::VersionType::version

Definition at line 127 of file NazaDecoder.h.

The documentation for this union was generated from the following file:

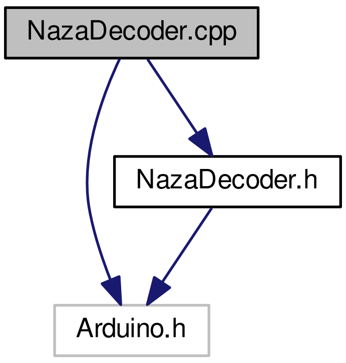
- NazaDecoder.h

4 File Documentation

4.1 NazaDecoder.cpp File Reference

```
#include <Arduino.h>
#include "NazaDecoder.h"
```

Include dependency graph for NazaDecoder.cpp:



4.2 NazaDecoder.cpp

```
#include <Arduino.h>
#include "NazaDecoder.h"

NazaDecoder::NazaDecoder()
        : seq(0), cnt(0), msgId(0), msgLen(0), cs1(0), cs2(0), magXMin(0), magXMax(0), magYMin(0), magYMax(
0), lon(0), lat(0), gpsAlt(0), spd(0), fix(NO_FIX), sat(0), heading(0), cog(0), gpsVsi(0), hdop(0), vdop(0),
 year(0), month(0), day(0), hour(0), minute(0), second(0) {
}
```

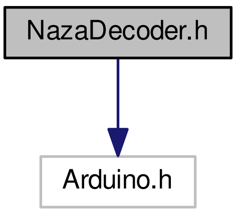
```
void NazaDecoder::updateChecksum(int16_t input) {
    cs1 += input;
    cs2 += cs1;
}

double NazaDecoder::getLat() {
    return lat;
}

double NazaDecoder::getLon() {
    return lon;
}

double NazaDecoder::getGpsAlt() {
    return gpsAlt;
}

double NazaDecoder::getSpeed() {
    return spd;
}

NazaDecoder::FixType NazaDecoder::getFixType() {
    return fix;
}

uint8_t NazaDecoder::getNumSat() {
    return sat;
}

double NazaDecoder::getHeading() {
    return heading;
}

double NazaDecoder::getCog() {
    return cog;
}

double NazaDecoder::getGpsVsi() {
    return gpsVsi;
}

double NazaDecoder::getHdop() {
    return hdop;
}

double NazaDecoder::getVdop() {
    return vdop;
}

uint8_t NazaDecoder::getYear() {
    return year;
}

uint8_t NazaDecoder::getMonth() {
    return month;
}

uint8_t NazaDecoder::getDay() {
    return day;
}

uint8_t NazaDecoder::getHour() {
    return hour;
}

uint8_t NazaDecoder::getMinute() {
    return minute;
}

uint8_t NazaDecoder::getSecond() {
    return second;
}

NazaDecoder::VersionType NazaDecoder::getFirmwareVersion
() {
    return firmwareVersion;
}

NazaDecoder::VersionType NazaDecoder::getHardwareVersion
() {
    return hardwareVersion;
}

uint8_t NazaDecoder::isLocked() {
    return locked;
}

```

```
uint8_t NazaDecoder::decode(int16_t input) {

    // header (part 1 - 0x55)
    if ((seq == 0) && (input == 0x55)) {
        seq++;
    }

    // header (part 2 - 0xaa)
    else if ((seq == 1) && (input == 0xaa)) {
        cs1 = 0;
        cs2 = 0;
        seq++;
    } else if (seq == 2) {
        msgId = input;
        updateChecksum(input);
        seq++;
    }

    // message id
    // message payload length (should match message id)
    // store payload in buffer
    else if ((seq == 3) && (((msgId == NAZA_MESSAGE_GPS_TYPE) && (input ==
NAZA_MESSAGE_GPS_SIZE)) || ((msgId ==
NAZA_MESSAGE_COMPASS_TYPE) && (input ==
NAZA_MESSAGE_COMPASS_SIZE)) || ((msgId ==
NAZA_MESSAGE_MODULE_VERSION_TYPE) && (input ==
NAZA_MESSAGE_MODULE_VERSION_SIZE)))) {
        msgLen = input;
        cnt = 0;
        updateChecksum(input);
        seq++;
    } else if (seq == 4) {
        payload[cnt++] = input;
        updateChecksum(input);
        if (cnt >= msgLen) {
            seq++;
        }
    }

    // verify checksum #1
    else if ((seq == 5) && (input == cs1)) {
        seq++;
    }

    // verify checksum #2
    else if ((seq == 6) && (input == cs2)) {
        seq++;
    } else {
        seq = 0;
    }

    // all data in buffer
    if (seq == 7) {
        seq = 0;

        // Decode GPS data
        if (msgId == NAZA_MESSAGE_GPS_TYPE) {
            uint8_t mask = payload[NAZA_MESSAGE_POS_XM];
            uint32_t time = pack4(NAZA_MESSAGE_POS_DT, mask);
            second = time & 0x3f;
            time >>= 6;
            minute = time & 0x3f;
            time >>= 6;
            hour = time & 0x0f;
            time >>= 4;
            day = time & 0x1f;
            time >>= 5;
            if (hour > 7) {
                day++;
            }
            month = time & 0x0f;
            time >>= 4;
            year = time & 0x7f;
            lon = (double) pack4(NAZA_MESSAGE_POS_LO, mask) / 10000000;
            lat = (double) pack4(NAZA_MESSAGE_POS_LA, mask) / 10000000;
            gpsAlt = (double) pack4(NAZA_MESSAGE_POS_AL, mask) / 1000;
            double nVel = (double) pack4(NAZA_MESSAGE_POS_NV, mask) / 100;
            double eVel = (double) pack4(NAZA_MESSAGE_POS_EV, mask) / 100;
            spd = sqrt(nVel * nVel + eVel * eVel);
            cog = atan2(eVel, nVel) * 180.0 / M_PI;
            if (cog < 0) {
                cog += 360.0;
            }
            gpsVsi = -(double) pack4(NAZA_MESSAGE_POS_DV, mask) / 100;
            vdop = (double) pack2(NAZA_MESSAGE_POS_VD, mask) / 100;
            double ndop = (double) pack2(NAZA_MESSAGE_POS_ND, mask) / 100;
            double edop = (double) pack2(NAZA_MESSAGE_POS_ED, mask) / 100;
            hdop = sqrt(ndop * ndop + edop * edop);
            sat = payload[NAZA_MESSAGE_POS_NS];
            uint8_t fixType = payload[NAZA_MESSAGE_POS_FT] ^ mask;
            uint8_t fixFlags = payload[NAZA_MESSAGE_POS_SF] ^ mask;
            switch (fixType) {
            case 2:
                fix = FIX_2D;
                break;
            case 3:
                fix = FIX_3D;
                break;
            default:
                fix = NO_FIX;
                break;
            }
            if ((fix != NO_FIX) && (fixFlags & 0x02)) {
                fix = FIX_DGPS;
            }
            uint16_t lock = pack2(NAZA_MESSAGE_POS_SN, 0x00);
            locked = (lock == lastLock + 1);
            lastLock = lock;
        }

        // Decode compass data (not tilt compensated)
        // To calculate the heading (not tilt compensated) you need to do atan2 on the resulting y any a values, convert radians to degrees and add 360 if the result is negative.
        else if (msgId == NAZA_MESSAGE_COMPASS_TYPE) {
            uint8_t mask = payload[4];
            mask = (((mask ^ (mask >> 4)) & 0x0F) | ((mask << 3) & 0xF0)) ^ (((mask & 0x01) << 3) | ((mask & 0x01) << 7));
            int16_t x = pack2(NAZA_MESSAGE_POS_CX, mask);
            int16_t y = pack2(NAZA_MESSAGE_POS_CY, mask);
            if (x > magXMax) {
                magXMax = x;
            }
            if (x < magXMin) {
                magXMin = x;
            }
            if (y > magYMax) {
                magYMax = y;
            }
            if (y < magYMin) {
                magYMin = y;
            }
            heading = -atan2(y - ((magYMax + magYMin) / 2), x - ((magXMax + magXMin) / 2)) * 180.0 / M_PI;
            if (heading < 0) {
                heading += 360.0;
            }
        } else if (msgId == NAZA_MESSAGE_MODULE_VERSION_TYPE) {
            firmwareVersion.version = pack4(NAZA_MESSAGE_POS_FW, 0x00);
            hardwareVersion.version = pack4(NAZA_MESSAGE_POS_HW, 0x00);
        }
        return msgId;
    } else {
        return NAZA_MESSAGE_NONE_TYPE;
    }
}

int32_t NazaDecoder::pack4(uint8_t i, uint8_t mask) {
    union {
        uint32_t d;
        uint8_t b[4];
    } v;
    for (int j = 0; j < 4; j++)
        v.b[j] = payload[i + j] ^ mask;
    return v.d;
}

int16_t NazaDecoder::pack2(uint8_t i, uint8_t mask) {
    union {
        uint16_t d;
        uint8_t b[2];
    } v;
    for (int j = 0; j < 2; j++) {
        v.b[j] = payload[i + j] ^ mask;
    }
    return v.d;
}

```

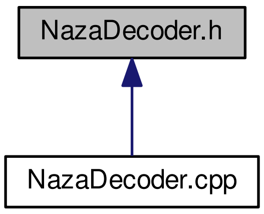
4.3 NazaDecoder.h File Reference

```
#include <Arduino.h>
```

Include dependency graph for NazaDecoder.h:



This graph shows which files directly or indirectly include this file:



Classes

- class NazaDecoder
- struct NazaDecoder::VersionSchemeType
- union NazaDecoder::VersionType

Macros

- #define MESSAGE_HEADER_SIZE 0x04

4.3.1 Macro Definition Documentation

4.3.1.1 #define MESSAGE_HEADER_SIZE 0x04

Arduino Naza Decoder.

Inspired by the Pawelsky's work.

Definition at line 12 of file NazaDecoder.h.

4.4 NazaDecoder.h

```

#ifndef __ARDUINO_NAZA_DECODER_H__
#define __ARDUINO_NAZA_DECODER_H__

#include <Arduino.h>

#define MESSAGE_HEADER_SIZE 0x04

class NazaDecoder {

public:

    typedef enum {

        // date and time
        NAZA_MESSAGE_POS_DT = 0x04 - MESSAGE_HEADER_SIZE,

        // longitude (x10^7, degree decimal)
        NAZA_MESSAGE_POS_LO = 0x08 - MESSAGE_HEADER_SIZE,

        // latitude (x10^7, degree decimal)
        NAZA_MESSAGE_POS_LA = 0x0c - MESSAGE_HEADER_SIZE,

        // altitude (in milimeters)
        NAZA_MESSAGE_POS_AL = 0x10 - MESSAGE_HEADER_SIZE,

        // horizontal accuracy estimate (see uBlox NAV-POSLLH message for details)
        NAZA_MESSAGE_POS_HA = 0x14 - MESSAGE_HEADER_SIZE,

        // vertical accuracy estimate (see uBlox NAV-POSLLH message for details)
        NAZA_MESSAGE_POS_VA = 0x18 - MESSAGE_HEADER_SIZE,

        // NED north velocity (see uBlox NAV-VELNED message for details)
        NAZA_MESSAGE_POS_NV = 0x20 - MESSAGE_HEADER_SIZE,

        // NED east velocity (see uBlox NAV-VELNED message for details)
        NAZA_MESSAGE_POS_EV = 0x24 - MESSAGE_HEADER_SIZE,

        // NED down velocity (see uBlox NAV-VELNED message for details)
        NAZA_MESSAGE_POS_DV = 0x28 - MESSAGE_HEADER_SIZE,

        // position DOP (see uBlox NAV-DOP message for details)
        NAZA_MESSAGE_POS_PD = 0x2c - MESSAGE_HEADER_SIZE,

        // vertical DOP (see uBlox NAV-DOP message for details)
        NAZA_MESSAGE_POS_VD = 0x2e - MESSAGE_HEADER_SIZE,

        // northing DOP (see uBlox NAV-DOP message for details)
        NAZA_MESSAGE_POS_ND = 0x30 - MESSAGE_HEADER_SIZE,

        //easting DOP (see uBlox NAV-DOP message for details)
        NAZA_MESSAGE_POS_ED = 0x32 - MESSAGE_HEADER_SIZE,

        // number of satellites (not XORed)
        NAZA_MESSAGE_POS_NS = 0x34 - MESSAGE_HEADER_SIZE,

        // fix type (0 - no lock, 2 - 2D lock, 3 - 3D lock, not sure if other values can be expected - see uBlox NAV-SOL message for details)
        NAZA_MESSAGE_POS_FT = 0x36 - MESSAGE_HEADER_SIZE,

        // fix status flags (see uBlox NAV-SOL message for details)
        NAZA_MESSAGE_POS_SF = 0x38 - MESSAGE_HEADER_SIZE,

        // XOR mask
        NAZA_MESSAGE_POS_XM = 0x3b - MESSAGE_HEADER_SIZE,

        // sequence number (not XORed), once there is a lock - increases with every message. When the lock is lost later LSB and MSB are swapped with every message.
        NAZA_MESSAGE_POS_SN = 0x3c - MESSAGE_HEADER_SIZE,

        // checksum, calculated the same way as for uBlox binary messages
        NAZA_MESSAGE_POS_CS = 0x3e - MESSAGE_HEADER_SIZE
    } GPSPayloadPosition;

    typedef enum {

        // compass X axis data (signed)
        NAZA_MESSAGE_POS_CX = 0x04 - MESSAGE_HEADER_SIZE,

        // compass Y axis data (signed)
        NAZA_MESSAGE_POS_CY = 0x06 - MESSAGE_HEADER_SIZE,

        // compass Z axis data (signed)
        NAZA_MESSAGE_POS_CZ = 0x08 - MESSAGE_HEADER_SIZE
    } CompassPayloadPosition;

    typedef enum {

        // firmware version
        NAZA_MESSAGE_POS_FW = 0x08 - MESSAGE_HEADER_SIZE,

        // hardware id
        NAZA_MESSAGE_POS_HW = 0x0c - MESSAGE_HEADER_SIZE
    } ModuleVersionPayloadPosition;

    typedef enum {
        NAZA_MESSAGE_NONE_TYPE = 0x00,
        NAZA_MESSAGE_GPS_TYPE = 0x10,
        NAZA_MESSAGE_COMPASS_TYPE = 0x20,
        NAZA_MESSAGE_MODULE_VERSION_TYPE = 0x30
    } MessageType;

    typedef enum {
        NAZA_MESSAGE_GPS_SIZE = 0x3a,
        NAZA_MESSAGE_COMPASS_SIZE = 0x06,
        NAZA_MESSAGE_MODULE_VERSION_SIZE = 0x0c
    } MessageSize;

    typedef enum {
        NO_FIX = 0,
        FIX_2D = 2,
        FIX_3D = 3,
        FIX_DGPS = 4
    } FixType;

    typedef struct {
        uint8_t revision;
        uint8_t build;
        uint8_t minor;
        uint8_t major;
    } VersionSchemeType;

    typedef union {
        uint32_t version;
        VersionSchemeType scheme;
    } VersionType;

    NazaDecoder();

    uint8_t decode(int16_t input);
    double getLat();
    double getLon();
    double getGpsAlt();
    double getSpeed();
    FixType getFixType();
    uint8_t getNumSat();
    double getHeading();
    double getCog();
    double getGpsVsi();
    double getHdop();
    double getVdop();
    uint8_t getYear();
    uint8_t getMonth();
    uint8_t getDay();

    // Note that for time between 16:00 and 23:59 the hour returned from GPS module is actually 00:00 - 7:59.
    uint8_t getHour();
    uint8_t getMinute();
    uint8_t getSecond();

    // Note that you need to read version numbers backwards (02 01 00 06 means v6.0.1.2)
    VersionType getFirmwareVersion();
    VersionType getHardwareVersion();

    uint8_t isLocked();
private:
    int16_t payload[58];
    int16_t seq;
    int16_t cnt;
    int16_t msgId;
    int16_t msgLen;

    // checksum #1
    uint8_t cs1;

    // checksum #2
    uint8_t cs2;
    int16_t magXMin;
    int16_t magXMax;
    int16_t magYMin;
    int16_t magYMax;

    // longitude in degree decimal
    double lon;

    // latitude in degree decimal
    double lat;

    // altitude in m (from GPS)
    double gpsAlt;

    // speed in m/s
    double spd;

    // fix type
    FixType fix;

    // number of satellites
    uint8_t sat;

    // heading (not tilt compensated) in degrees
    double heading;

    // course over ground
    double cog;

    // vertical speed indicator (from GPS) in m/s (a.k.a. climb speed)
    double gpsVsi;

    // horizontal dilution of precision
    double hdop;

    // vertical dilution of precision
    double vdop;
    uint8_t year;
    uint8_t month;
    uint8_t day;
    uint8_t hour;
    uint8_t minute;
    uint8_t second;

    VersionType firmwareVersion;
    VersionType hardwareVersion;

    uint16_t lastLock;
    uint8_t locked;

    int32_t pack4(uint8_t i, uint8_t mask);

    int16_t pack2(uint8_t i, uint8_t mask);

    void updateChecksum(int16_t input);
};

#endif /* __ARDUINO_NAZA_DECODER_H__ */
```

Index