Processing the NAV-PVT Message on ZEDF9P

Previously we read data from the ZEDF9P NAV-PVT message (page 92).

We didn't get actual data from it though.

See page 45 for the payload contents of the NAV-PVT message.

If we want to efficiently process the data, we will use a Struct.

A struct is like an array but it can contain values of different types.

```
4  typedef struct {
5    uint16_t year;
6    uint8_t  month;
7    uint8_t  day;
8    uint8_t  hour;
9    uint8_t  minute;
10   uint8_t  second;
11   long  lat;          // Scaling 1e-7
12   long  lon;          // Scaling 1e-7
13   long  headMot;      // Scaling 1e-5
14 } NavData;
15
16 NavData gpsData;
```

members

struct name

Type

Name

→ declaring a Struct.

To access members of a struct we use the "." dot operator instead of brackets and an index like an array.

```
gpsData.Year = 2025;
gpsData.Month = 06;
gpsData.day = 10;
```

Processing the NAV-PVT Message on ZED F9P

06/10/25 Wesley Cochu

We can write the code from page 92 as a method to request the NAV PVT packet.

```
128 void requestNAVPVT()
129 {
130   // This method asks the GPS for the NAV PVT message
131   // by sending it an empty NAV PVT packet.
132   // Ctrl-f 0x01 0x07 in the interface description for more info
133
134   // Message to send to request NAV_PVT.
135   // 0x01 is the Class
136   // 0x07 is the ID
137   // 0x00 and 0x00 is the length
138   // Per the data sheet: we send an empty packet to request the data.
139   uint8_t nav_pvt[4] = {0x01, 0x07, 0x00, 0x00};
140
141   // Checksums
142   byte CK_A = 0;
143   byte CK_B = 0;
144
145   // Compute Checksum
146   for (int ii=0; ii<4; ii++)
147   {
148     CK_A = CK_A + nav_pvt[ii];
149     CK_B = CK_B + CK_A;
150   }
151
152   // Send the packet
153   Wire.beginTransmission(gps_add);
154   Wire.write(0xB5); // Sync Char1
155   Wire.write(0x62); // Sync Char2
156   Wire.write(0x01); // Class
157   Wire.write(0x07); // ID
158   Wire.write(0x00); // Length in Little Endian Order!
159   Wire.write(0x00);
160   Wire.write(CK_A); // Check sum!
161   Wire.write(CK_B);
162   Wire.endTransmission();
163   delay(100);
164 }
```

To Make it easier to work with all the data we want to return, we can pass our structure to the a method that gets new values. We can pass by reference or by value.

Pass by value: getGPSData(NavData myData);
↳ This will create a copy of the data.
↳ Like writing all my notes on a new notebook and then editing.

Pass by Reference: getGPSData(NavData &myData)
↳ The & Tells uso that we are passing the structure by reference
↳ This is like handing anynotebook over and you writing in it.

```
18 int getGPSData(NavData &myData)
19 {
20   while(GPSAvailable() < 0)
21   {delay(100);}
22
23   uint16_t availableBytes = GPSAvailable();
24
25   Serial.print(availableBytes); Serial.println();
26
27   if (availableBytes == 100)
28   {
29     Serial.println("Got 100 Bytes!");
30
31     byte header[6];
32     byte payload[92]; //Store these separate so we can
33     byte CK_A;        use the byte OFF Sets From Page 45
34     byte CK_B;
35
36     // Read the bytes available from the GPS and print them in HEX
37     Wire.requestFrom(gps_add, availableBytes);
38     for (int ii=0; ii<6; ii++)   // Read the first 6 bytes
39     {
40       header[ii] = Wire.read();
41       Serial.print(header[ii], HEX);  Serial.print(" ");
42     }
43     for (int ii=0; ii<92; ii++)  // Read the 92 payload bytes
44     {
45       payload[ii] = Wire.read();
46       Serial.print(payload[ii], HEX); Serial.print(" ");
47     }
48     // new line between readings.
49     Serial.println();
50     CK_A = Wire.read();      // Read the 2 checksum bytes
51     CK_B = Wire.read();
```

» defined on page 88

Processing the NAV-PVT Message on ZEDF9P


```ino
// Sync Bytes should be equal to 0x8562
uint16_t headerCheck  = (header[0]<<8 | header[1]);
uint16_t classIDCheck = (header[2]<<8 | header[3]);
uint16_t lengthCheck  = (header[5]<<8 | header[4]);

byte CK_CK_A = 0;
byte CK_CK_B = 0;
// Make sure the header information is correct for the NAV PVT Message
if (headerCheck==0x8562 && (classIDCheck==0x0107 && lengthCheck==0x005C)) //Make sure we are reading the right message.
{
  // Check the checksum! → Page 89 for details
  for(int ii=2; ii<6; ii++)
  {
    CK_CK_A += header[ii];
    CK_CK_B += CK_CK_A;
  }
  for(int ii=0; ii<92; ii++)
  {
    CK_CK_A += payload[ii];
    CK_CK_B += CK_CK_A;
  }

  if (CK_CK_A == CK_A && CK_CK_B == CK_B)
  {
    Serial.println("CHECK SUMS PASSED!");

    // Read the data into our struct!
    myData.year     = payload[5]<<8 | payload[4];
    myData.month    = payload[6];
    myData.day      = payload[7];
    myData.hour     = payload[8];
    myData.minute   = payload[9];
    myData.second   = payload[10];

    myData.lon      =  payload[27]<< 24 | payload[26]<<16 | payload[25]<<8  | payload[24];
    myData.lat      =  payload[31]<< 24 | payload[30]<<16 | payload[29]<< 8 | payload[28];
    myData.headMot  =  payload[67]<< 24 | payload[66]<<16 | payload[65]<<8  | payload[64];

    return 1; → tell the program we succeeded
  }
}
// We expect 100 bytes because there are 2 sync bytes, a class, an id, 2 length bytes, 92 payload bytes, and 2 checksum bytes.
// 2 + 1 + 1 + 2 + 92 + 2 = 92 + 8 = 100!
}
else
{
  Serial.println("Weird number of bytes...");
  // Read the availableBytes out if we didn't get 100 like we expected.
  Wire.requestFrom(gps_add, availableBytes);
  for (int jj=0; jj<availableBytes; jj++)
  {
    Wire.read();
  }
  return -1;→ tell the Program we failed.
}
return -1;
}
```

Update our copy of gps data.

Processing the NAV-PVT Message on ZEDF9P

Example Driver Code and Output:

```
172 void loop()
173 {
174   requestNAVPVT();
175   int gpsSuccess = getGPSData(gpsData);
176
177   if (gpsSuccess == 1)
178   {
179     Serial.print("Lat: "); Serial.println(gpsData.lat);
180     Serial.print("Lon: "); Serial.println(gpsData.lon);
181     Serial.print("Time: "); Serial.print(gpsData.hour);  Serial.print(":"); Serial.print(gpsData.minute); Serial.print(":"); Serial.println(gpsData.second);
182     Serial.print("Date: "); Serial.print(gpsData.month); Serial.print("/"); Serial.print(gpsData.day);    Serial.print("/"); Serial.println(gpsData.year);
183   }
184   else
185   {
186     Serial.println("Something went wrong reading the gps!");
187   }
188   // Wait for the GPS to tell us it has a response
189   delay(1500);
190 }
```

```
20:09:07.678 -> Got 100 Bytes!
20:09:07.678 -> B5 62 1 7 5C 0 8 6B C8 19 E9 7
20:09:07.678 -> CHECK SUMS PASSED!
20:09:07.678 -> Lat: 334679517
20:09:07.678 -> Lon: -819906964
20:09:07.678 -> Time: 0:9:7
20:09:07.678 -> Date: 6/13/2025
20:09:09.277 -> 100
20:09:09.277 -> Got 100 Bytes!
20:09:09.277 -> B5 62 1 7 5C 0 D8 72 C8 19 E9 7
20:09:09.325 -> CHECK SUMS PASSED!
20:09:09.325 -> Lat: 334679479
20:09:09.325 -> Lon: -819906750
20:09:09.325 -> Time: 0:9:9
20:09:09.325 -> Date: 6/13/2025
```

After enough Satellites are in view we get good coordinate information.