



Cirocomm Technology Corp.

No. 3, Tze-Chiang 1st Road, Jungli Industrial Park, Jungli City, T
aoyuan Hsien, Taiwan 320, R.O.C.
TEL: 886-3-4524146 · FAX: 886-3-4511048
<http://www.cirocomm.com.tw>
E-mail :cirocomm@cirocomm.com.tw

Application Notes: AP-1 (For PE36)

To make sure PE36 can be perfect fit into your design, the following I/O signals, hardware and software design must take care.

1. Reset# (pin 2):

- a. With battery backup, If Reset# activate, the Pe36 will go to hot-start, warm-start or cold-start procedure depending on the ephemeris data and Almanac data is valid.
- b. Without battery backup, the Pe36 will go to cold-start procedure whenever the Reset# signal is activated.

2. The battery selection:

We highly recommend of using the Lithium battery instead of the super capacitors. Because the “super capacitor” doesn’t have much enough energy to keep the SRAM data valid longer and will make the CPU in the engine board go to dead loop which can only be wake up by cold-start command.

3. The RDX0 signal (pin 4)

We highly recommend the user’s control unit must connect to RDX0 to keep sending the command to the GPS engine board. It is better that host can send command to the GPS receiver.

4. RDX1 signal (pin 5)

This pin is used for the DGPS. You can leave it open if you don’t have DGPS function.

5. Battery (pin 7)

You can connect to battery as showing on the schematic or leave it open without connecting.

6. The filtering function.

Please see the filtering program on the second page. With this filtering program running on the host CPU side will make performance better.

7. The PE36 engine board interfaces recommend circuit on last page.

SPEED FILTERING CODE

This algorithm uses vector addition to smooth out speed data. The heading from the input message provides a directional vector and the speed is used for the vector magnitude. The vectors are then summed over a number of samples to reduce noise.

Other options include:

- 1: An average bias can be subtracted from the speed.
- 2: The filtering can be turned off at some speed threshold.

Here is the source code written in C.

// Structure for holding the heading data

```
typedef struct {  
    double x;  
    double y;  
} VECTOR;
```

// variables used

```
BOOL g_bSpeedAveraging = FALSE;    // this is a flag that enables or disables Speed  
Averaging  
int g_iSpeedSamples = 5;           // number of samples to average over  
int g_iSpeedThreshold = 20;       // speed at which averaging is disabled  
double g_iSpeedBias = 1.0;        // an average bias value to subtract from speed  
  
double m_dSpeedOverGround;        // variable that holds speed value from input  
message  
double m_dCourseOverGround;       // variable that holds heading value from input  
message  
  
#define NUM_VECTORS 1024          // must be greater than or equal to the number of  
samples  
static VECTOR sVector[NUM_VECTORS];  
static int nVectorPos = 0;  
  
// during some initialization call  
void InitSpeedData()  
{
```

```

for(int i = 0; i<NUM_VECTORS; i++)
{
    sVector[i].x = 0;
    sVector[i].y = 0;
}
nVectorPos = 0;
}

// after each message with speed
if(g_bSpeedAveraging)
    AverageSpeedData();

// here are the routines
void AverageSpeedData()
{
    double totalx = 0, totaly = 0, newmag, course;

    // if speed is above threshold, don't do averaging
    if(m_dSpeedOverGround > g_iSpeedThreshold)
        return;

    // check for pointer wrap
    if(nVectorPos == g_iSpeedSamples)
        nVectorPos = 0;

    // convert heading degrees to radians
    double theta = m_dCourseOverGround*3.14159/180;

    // create vector with direction from heading and magnitude from speed
    sVector[nVectorPos].x = m_dSpeedOverGround*sin(theta);
    sVector[nVectorPos].y = m_dSpeedOverGround*cos(theta);
    nVectorPos++;

    // do the vector addition
    for(int i = 0; i<g_iSpeedSamples; i++) {
        totalx += sVector[i].x;
        totaly += sVector[i].y;
    }
}

```

```
totalx /= g_iSpeedSamples;
totaly /= g_iSpeedSamples;

// new magnitude and course
newmag = sqrt(totalx*totalx + totaly*totaly);
course = atan(totalx/totaly)*180/3.14159;
if(course < 0)
    course = 360 + course;

// adjust for speed bias
newmag -= g_iSpeedBias;
if(newmag < 0)
    newmag = 0;

// this is filtered speed data
m_dSpeedOverGround = newmag;

// filtered heading data
m_dCourseOverGround = course;
}
```

