Technical Note

TN214

Connecting the Rabbit 2000 to a Garmin GPS25 Receiver

This technical note covers the hardware issues of connecting the Rabbit 2000 chip to a GPS (Global Positioning System) receiver and the software routines for decoding the data from a GPS receiver. The receiver used in this example is a Garmin GPS25-LVC or GPS25-LVS.

GPS Hardware and Connections

All standard GPS hardware communicates using the NMEA-0183 standard for marine electronics devices. Most GPS hardware also supports a variety of additional proprietary communications protocols, but NMEA-0183 is the ubiquitous standard that is supported by the software libraries described below.

The NMEA-0183 standard specifies that communications between devices is through a standard serial link running at 4800 bps. Both versions of the GPS25 use this standard. The GPS25-LVC transmits at CMOS levels (0 V, 5 V), and the GPS25-LVS transmits at both CMOS and RS-232 levels.

- If you are using a Jackrabbit board, you must use the RS-232 line with the GPS25-LVS. Consult the pinout for the receiver and connect TXD1 on the GPS receiver to RXC on the Jackrabbit board.
- If you are using a RabbitCore module, you can use the CMOS line from either receiver. Connect the NMEA output pin on the receiver to PC3 on the RabbitCore board.

GPS systems often will not work properly indoors. If you have access to a portable computer, you can test the GPS system outside or in a car. Both the Jackrabbit and the RabbitCore prototyping boards have voltage regulators that will allow you to power the board and the GPS receiver from a car's cigarette lighter. You will need to build an adapter to route power from the cigarette lighter port to the prototyping board's power header. Once this is done, you can use the regulated power on the prototyping board (GND and Vcc) to power the receiver.

NMEA-0183 Protocol

The NMEA-0183 protocol consists of ASCII “sentences” sent repeatedly by the GPS receiver. These sentences always start with the character **$** and end with a carriage return/newline sequence. The format is:

```
${talker id}{sentence id},{comma separated list of fields...}{optional
  checksum}\r\n
```

The talker id is a two-letter code that indicates the type of device sending the message. This will always be “GP” when reading data from a GPS receiver.

The sentence id is a three-letter code that indicates the type of information being sent and the format of the following data fields. The different sentence types that can be parsed by the Dynamic C library, **GPS.LIB**, are:

- GGA
- GLL
- RMC

These are common sentence types supported by almost all GPS systems.

All these sentence formats provide the geographical location, which can be extracted using the function **gps_get_position()**. The RMC type contains full UTC time information, which can be parsed by the **gps_get_utc()** function. Both functions are available in the library **GPS.LIB** described in the next section.

Software

The GPS utility library, **GPS.LIB**, provides common processing functions to the user. The first set of functions parses NMEA-0183 sentences and extracts the desired fields from them

gps_get_position

```
int gps_get_position(GPSPositon *newpos, char *sentence);
```

DESCRIPTION

Retrieves geographic position information from an NMEA-0183 sentence and fills in a **GPSPosition** structure. The format of a **GPSPosition** structure is:

```
typedef struct {
    int lat_degrees;
    int lon_degrees;
    float lat_minutes;
    float lon_minutes;
    char lat_direction;
    char lon_direction;
} GPSPosition;
```

lat_direction can be "N" or "S" and **lon_direction** can be "E" or "W."

PARAMETERS

newpos	GPSPosition struct to fill with the relevant fields in the NMEA-0183 sentence.
sentence	The NMEA-0183 sentence from the GPS receiver to be parsed

RETURN VALUE

0: Success
-1: Parsing error
-2: Sentence marked invalid

gps_get_utc

```
int gps_get_utc(struct tm *newtime, char *sentence);
```

DESCRIPTION

Parses an RMC sentence for the UTC time and date, and loads it into a **tm** structure.

PARAMETERS

newtime	A **tm** structure (see **mktm()** in the *Dynamic C Function Reference Manual*) that will be filled with the time and date reported in the sentence.
sentence	The NMEA-0183 sentence from the GPS receiver to be parsed. This must be an RMC sentence.

RETURN VALUE

0: Success
-1: Parsing error
-2: Sentence marked invalid

Calculating distance over land given a pair of geographical coordinates is a nontrivial calculation, and so a utility function is provided in the library to do this calculation.

gps_ground_distance

```
float gps_ground_distance(GPSPosition *a, GPSPosition *b);
```

DESCRIPTION

Calculates the distance over the surface of the earth between two geographical points. Uses a spherical model of the earth with a radius of 6371km.

PARAMETERS

a, b	The two geographical coordinates between which the distance is to be calculated. They are the same structures used by **gps_get_position()**.

RETURN VALUE

The distance in kilometers between the two points.

Example

The following sample, **gps_test.c**, is an example of a program receiving data from a GPS. It assumes that a GPS receiver is connected to Serial Port C on the Rabbit 2000 chip.

```
#use "gps.lib"
  #define CINBUFSIZE 127
  #define COUTBUFSIZE 127
  #define MAX_SENTENCE 100

const char dayname[7][4]={"Sun","Mon","Tue","Wed","Thu","Fri","Sat"};

const char monthname[12][4] = {"Jan", "Feb", "Mar", "Apr", "May",
   "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

GPSPosition current_pos;
  struct tm current_time;

main() {
   char sentence[MAX_SENTENCE];
   int input_char;
   int string_pos;
   char dir_string[2];

   serCopen(4800);
   string_pos = 0;
   dir_string[1] = 0;
```

```
while(1) {
   input_char = serCgetc();
   if(input_char == '\r' || input_char == '\n') {
      sentence[string_pos] = 0;              // add null
      printf("%s\n", sentence);
      if(gps_get_position(&current_pos, sentence) == 0) {
         dir_string[0] = current_pos.lat_direction;
         printf("Latitude: %d %f' %s\n", current_pos.lat_degrees,
            current_pos.lat_minutes, dir_string);
         dir_string[0] = current_pos.lon_direction;
         printf("Longitude: %d %f' %s\n", current_pos.lon_degrees,
            current_pos.lon_minutes, dir_string);
      }
      if(gps_get_utc(&current_time, sentence) == 0) {
         printf("UTC: %s %d-%s-%d %02d:%02d:%02d\n",
            dayname[current_time.tm_wday], current_time.tm_mday,
            monthname[current_time.tm_mon - 1], 1900 +
            current_time.tm_year, current_time.tm_hour,
            current_time.tm_min, current_time.tm_sec );
      }
      string_pos = 0;
   }
   else if(input_char > 0) {
      sentence[string_pos] = input_char;
      string_pos++;
      if(string_pos == MAX_SENTENCE)
         string_pos = 0;                     // reset string if too large
   }
 }
}
```

References

There is a large amount of information about GPS and the NMEA-0183 standard available on the Web.

Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Fax: (530) 757-3792

www.zworld.com

Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-8400
Fax: (530) 757-8402

www.rabbitsemiconductor.com