CyberTrack™ II

CT-3.2 Sourceless Head Tracker

Developer Manual

Revision 1.04
December 14, 1996

124 Race Street
San Jose, CA 95126
408-289-8340
http://www.genreality.com
email: Support@genreality.com

Contents

1. Quick Start

1.1 Introduction

Incorporating the CyberTrack headtracker into your application is a simple process. The instructions provided here are intended to get you off to a quick start. This method uses the default settings of the tracker. More advanced information regarding changingthe tracker's settings are provided in the later sections of this manual.

1.2 Default Settings

The default settings for the CyberTrack are as follows:

Absolute Coordinate	Binary	9600 Baud
Sampling Upon Request	Checksum OFF	

To change these settings, refer to the information provided in Section 2.

1.3 Incorporating the CTM

To make your application use the CyberTrack's head tracking capabilities, it is necessary to incorporate the CyberTrack Tracking Module (CTM). The CTM is a function that reads and transmits head tracking data from the CyberTrack to the host computer. To incorporate the CTM:

1. Add the files CTM.C and CTM.H (in Watcom C) to your Project file

   (these are needed for the CTM functions)

2. Detect the CTM

   ```
   err = detectCTM(&port);
   ```

3. Initialize the CTM

   ```
   err = initCTM (port);
   ```

4. Read the data from the CTM

   ```
   err = readCTM(port, &yaw, &pitch, &roll);
   ```

5. Use the data obtained in the step 3 to control the player's view

6. Close the CTM

   ```
   closeCTM(port);
   ```

And that's it—that's all you need to do

2. The CyberTrack Tracker Interface

2.1 Introduction

The CyberTrack Tracking Module (CTM) measures the orientation of the user's head for yaw, pitch and roll. It transmits this data to the host computer over an `RS-232` serial interface, running at either `9600` or `19200` baud with 8 data bits, l stop bit and no parity.

The CTM can operate in either polled or streaming mode, ASCII or binary data formats, with or without checksum and with or without filtering.

2.2 Data Formats

CyberTrack supports two data formats: ASCII and binary. Both formats send a packet of information containing three angles (yaw, pitch and roll). Yaw is measured over a full 360 degrees using a three-axis magnetometer. Pitch and roll are measured over a range ol -45 to +45 degrees using a two-axis inclinometer. An on-board processor performs the necessary mathematical transformations to convert the magnetometer outputs into an actual yaw value based on the pitch and roll. The processor also can filter the data. The only difference between the two tormats is the way in which data is represented. Binary folmat is the CTM default.

2.2.1 ASCII Format

ASCII format is useful for verifying tracker operation, typically by using a

communications program. In this format, each packet consists of a text string of the torm

Yyyy . yPspp . pRs r r . r

followed by a carriage retum (hex 0D) and a linefeed (hex 0A). The yyy.y value is the yaw value in ASCII, from 000.0 to 360.0. The s represents a sign (either ‘’+’ or ‘-’) and the *pp.p* and rr.r values are the pitch and roll respectively. To enter ASCII format, send the character ‘E’ to the CTM.

2.2.2 Binary Format

Most applications will use binary format, since the packets are smaller and easier to

parse. To enter binary format, send the character ‘F’ to the CTM.

In binary format, each packet consists of a series of at least 8 bytes. The first two bytes are both hexadecimal FF (0xFF in C); these mark the start of the packet. The next six values are the yaw, pitch and roll. Each is sent as a two-byte value, with the most significant (i.e., high-order) byte sent first. The top bit of the high-order byte is zero, preventing two consecutive 0xFF values from appearing in the data.

In other words, think of a binary packet as four 16-bit values:

11111111	11111111
0yyyyyyy	yyyyyyyy
0ppppppp	pppppppp
0rrrrrrr	rrrrrrrr

The 15-bit yaw value (yyyyyyy yyyyyyyy) is mapped onto the range 0 to 359.959.

If 0x8000 could be represented, it would be 360 degrees. In other words, 0 is zero and 0x7FFF is 359.959 degrees.

The 15-bit pitch and roll values aue mapped onto the range -90 to +90; in other words, 0 is -90 degrees and 0x7FFF is +90 degrees.

2.3 Sampling Modes

CyberTrack supports two modes of operation, continuous and polled.

2.3.1 Continuous Mode

Continuous Mode continuously sends data packets to the host. It can result in many interrupts, which may be a problem tor some applications. A missed packet can be ignored since another one will follow immediately with more up-to-date information.

2.3.2 Polled Mode

In polled mode, the CTM does not send data to the host until the host explicitly requests it. Polled mode can be entered by sending an ASCII ‘G’ to the CTM. To request a sample, send an ASCII ‘S’. To return to continuous mode, send an ‘H’. The advantage of using polled mode is fewer interrupts occur, but it does cause increased latency.

2.4 Filtering

The microprocessor in the CTM can filter data from the tracker before sending it to the host.

2.4.1 Turning Filtering On

To enable filtering, send an ASCII ‘T’ to the CTM. With filtering enabled, the tracker data averages a number of data samples. This reduces instability (jitter), but increases latency (lag).

2.4.2 Turning Filtering Off

To disable filtering, send an ASCII ‘W’ to the CTM. In this mode, jitter may be more evident. If CTM-based filtering is not used, some form of smoothing filter in the application may be indicated.

2.5 Raw Data Mode

The final feature is raw data mode. In this mode, the magnetometers and inclinometers send raw values to the host. This mode’s data consists of a 16-byte binary packet of the following form:

11111111	11111111
0xxxxxxxx	xxxxxxxx
0yyyyyyy	yyyyyyyy
0zzzzzzz	zzzzzzzz
0ppppppp	pppppppp
0ppppppp	pppppppp
0rrrrrrr	rrrrrrrr
0rrrrrrr	rrrrrrrr

The x, y and z values are the outputs ot the X-axis, Y-axis and Z-axis magnetometers.

To enter this mode, send an ASCII ‘B’. To return to the default absolute-coordinate mode, send an ASCII ‘A’.

Do not use this mode unless all processing of data is in the application.

The ASCII mode format of the raw data mode is not defined and subject to change without notice. If raw data mode is used, use binary format only.

2.6 Resetting the CTM

Sending an ASCII ‘R’ to the CTM resets it to its default state (absolute coordinates,

ASCII folmat, continuous sampling, no filtering).

2.7 Writing Tracker Software

1. Initialize the serial port to 9600 baud, 8 data bits, 1 stop bit and no pality.

2. Send an ‘R’ to reset the CTM (in case a previous application left it in a strange state). The CTM will send back a copyright message.

3. Send an ‘F’ to enter binaly mode (delay a few hundred milliseconds to make sure the CTM responds).

4. When data comes in, check lor two consecutive 0xFF’s. If present, reset your input buffer pointer to zero.

5. After receiving six bytes, get the yaw, pitch and roll values from the packet butter and scale them to the correct range.

3 Tracking Functions (CTM.C)

3.1 detectCTM

3.1.1 Syntax

```
int detectCTM(int * port);
```

3.1.2 Purpose

The detectCTM function returns the COM Port to which the CTM is connected.

3.1.3 Parameters

None.

3.1.4 Returns

TRUE or FALSE	Depending on success
*int *port:*	The Com Port to which the CyberTrack is connected

3.1.5 Example Code

```
int comport; int err;
err=findCTM(&comport);
if (err==O) printf("CTM not detected.\n");
else printf("CTM detected on COM%d\n",comport);
```

3.2 initCTM

3.2.1 Syntax

```
int initCTM(int comport);
```

3.2.2 Purpose

The initCTM function initializes the CTM, enabling the Interrupt Service Routine and requesting CyberTrack checksum mode.

3.2.3 Parameters

int comport: The COM Port to which the CTM is connected.

3.2.4 Returns

TRUE or FALSE Depending on whether initialization is successful

3.2.5 Example code

```
int comport=2; int err;
err=initCTM(comport);
if (err==0) printf ("CyberTrack failed
     initialization.");
else printf("CyberTrack is initialized on
COM%d.",comport);
```

3.3 readCTM

3.3.1 Syntax

```
int readCTM(int comport, float *yaw, float *pitch,
     float *roll);
```

3.3.2 Purpose

The *readCTM* function reads and decodes a CTM data packet.

3.3.3 Parameters

comport	The Com Port to which the CyberTrack is connected

3.3.4 Returns

TRUE or FALSE	Depending on success
*float * yaw:*	Compass heading of CTM (0 to 360) in degrees
*float * pitch:*	Chin-to-chest tilt angle (-45 to 45) in degrees
*float * roll:*	Ear-to-shoulder tilt angle (-45 to 45) in degrees

3.3.5 Example Code

```
int comport=2; int err; float yaw, pitch, roll;
err = readCTM(comport, &yaw, &pitch, &roll);
if (err==!O) printf (“Yaw: %i  Pitch: %i  Roll: %i”,
          yaw,pitch, roll);
```

3.4 versionCTM

3.4.1 Syntax

```
int versionCTM(int comport, int *version, int *revision,
char *subrevision);
```

3.4.2 Purpose

The *versionCTM* function will give you the version, revision and subrevision of the tracker’s firmware.

3.4.3 Parameters

int comport: The Com Port to which your CyberTrack is connected

3.4.4 Returns

TRUE or FALSE depending on whether version read is successful

*int *version:* version of the firmware

*nt *revision:* revision of the firmware

*char *subrevision:* subrevision of the firmware

3.4.5 Notes

The complete firmware "version" is made up thus:

v.rrs (example 1.23d)

v = version (ex. 1)

rr = revision (ex. 23)

s = subrevision (ex. d)

3.4.6 Example Code

```
int version, revision; char subrevision; int flag;
int comport=2;
flag= versionCTM (comport, &version, &revision,
     &subrevision);
if (flag==TRUE) printf ("Firmware Version is: %i.%i%c",
     version, revision, subrevision);
```

3.5 closeCTM

3.5.1 Syntax

```
     void closeCTM(int comport);
```

3.5.2 Purpose

The *closeCTM* function turns off the interrupt service routine and closes the serial connection to the CyberTrack.

3.5.3 Returns

nothing

3.5.4 Example Code

```
int port=2;
```

3.6.1 Short Example Program

```
#include “ctm.h n
#include “ctm.c”
main ()
{
float y,p,r;
int port;
int err;
err= detectCTM(&port);
err= initCTM (port);
err=readCTM (port, &y, &p, &r);
printf (“y%f p%f r%f” ,y,p,r);
closeCTM(port);
}
```

3.6.2 Important Notes:

[1] This library is designed for Watcom compiler.

[2] At present this library only supports COM l and COM2.

[3] This library may not function properly it you are using protected mode. You will have to install a bimodal interrupt handler in place of the Interrupt Service Routine.

General Reality Company
124 Race Street
San Jose, CA 95126
408-289-8340 Phone
www.genreality.com