Dynamic Linking with ANT DLL

ABSTRACT

Dynamic linking can be used to integrate the ANT PC Library into a PC application. This application note details the recommended use of the dynamic linking method for the ANT PC Library. Sample C++ code is provided to illustrate the recommended use.

COPYRIGHT INFORMATION

TABLE OF CONTENTS

1 Introduction

For the purposes of developing custom PC ANT applications for the ANT USB stick, an ANT DLL is supplied on the development kit CD ROM. The ANT DLL, source code and example PC applications utilizing Borland Builder and Visual C++ can be found on the development kit CD-ROM. This document provides the basic steps required to load and the use the DLL within a PC application.

2 Basics of Dynamic Linking

Dynamic linking allows an application to load a library at run time rather than at time of compilation. The application usually loads the addresses of functions located in the library and assigns pointers so that these functions can be called during the execution of the application.

The ANT DLL is a library that contains the definitions of the functions listed in ANT_Interface.h. These functions allow an application to make full use of ANT features from a PC through the ANT USB device.

3 Dynamic Linking With the ANT DLL

As described above, the ANT DLL must be loaded and utilized at run time by the application that wishes to use dynamic linking. Most of this is accomplished using windows functions that allow loading and handling of a DLL.

The sequence of steps required for dynamically linking to and using the ANT DLL are listed below.

3.1 Loading the DLL

Before the ANT DLL can be used, it must be loaded during the execution / loading of the application. This is performed by the function named LoadLibrary() that returns a handle to the DLL file if it's loaded successfully and is NULL otherwise.

```
// Example Usage

HMODULE hDLL = NULL; // declaration of handle

hDLL = LoadLibrary("ANT_DLL.dll");

if (hDLL == NULL)
        // error message and handling
Else
        // continue
```

3.2 Loading addresses of functions

Upon successful load of the DLL, it is necessary to load the addresses of the functions defined in the DLL into local pointers in the application. Only then can the functions in the DLL be called and used to perform tasks required by the application.

The loading of the addresses of the functions is performed using the GetProcAddress() function as shown below. It is important to note that the types of the function pointers must match the type of functions to which they will point.

```
// Example Usage

Typedef void (*ANT_vFn) ();
// no arguments

ANT_vFn ANT_Close = NULL;

ANT_Close = (ANT_vFn) GetProcAddress(hDLL,
"ANT_Close");
```

3.3 Using the DLL

The functions of the DLL can be called in the same way as any other C/C++ function call with the arguments and return values as specified by the function prototypes provided in ANT_Interface.h.

```
// Example Usage

ANT_AssignChannel(0, 0x00, 0);
// Receive Channel on network number 0
```

3.4 Closing the link to DLL

The connection to the ANT DLL can be terminated using the standard method for disconnecting from a DLL – by using the FreeLibrary() function. The argument to function is the handle which was initialized by the LoadLibrary() function.