第5章

动态链接库封装和调用

CHAPTER 5

LabVIEW项目开发时,经常会遇到仪器设备控制问题,对于 NI 提供的仪器设备,可以 通过安装相关的开发工具包来解决;对于非 NI 提供的设备,可以通过调用动态链接库 (Dynamic Link Library,DLL)的方式来解决。本章将以 RTL-SDR 的 LabVIEW 接口函数 为例,介绍 LabVIEW 中动态链接库的封装和调用方法。

5.1 RTL-SDR 接口函数的封装

LabVIEW 通过调用 RTL-SDR 的接口函数实现对 RTL-SDR 的控制。常用的接口函数有 open 函数、set sample rate 函数、set center freq 函数、read sync 函数和 close 函数。3.2.2 节详细介绍了 RTL-SDR 常用接口函数。3.2.3 节通过数据采集实例介绍了 RTL-SDR 接口函数的使用方法。

打开 RTL-SDR 接口函数的程序框图,可以看到,LabVIEW 是通过调用库函数(Call Library Function,CLF)模块调用动态链接库 rtlsdr.dll 的方式实现设备控制。下面将介绍 动态链接库和 LabVIEW 接口函数的封装方法。

5.1.1 动态链接库简介

动态链接库(DLL)是 Windows 操作系统实现共享函数库的一种方式。在 Windows 操作系统中使用 DLL,能够使应用程序代码变得更简洁,计算机内存资源的使用效率变得更高。

RTL-SDR 的应用程序接口(Application Programming Interface, API)函数是采用 C 语言编写的。通过调用 RTL-SDR 的动态链接库 rtlsdr. dll,上层 SDRsharp、LabVIEW、 MATLAB/Simulink 和 GNURadio 等应用软件就可以控制 RTL-SDR 设备,如图 5-1 所示。



图 5-1 RTL-SDR 接口函数

动态链接库 rtlsdr. dll 实现了 RTL-SDR 的所有接口函数。RTL-SDR 源程序中的头文件 rtl-sdr. h 对所有的接口函数作了定义。

```
# ifndef __RTL_SDR_H
\# define __RTL_SDR H
# ifdef cplusplus
extern "C" {
# endif
# include < stdint. h >
# include < rtl - sdr export. h >
typedef struct rtlsdr dev rtlsdr dev t;
RTLSDR API uint32 t rtlsdr get device count(void);
RTLSDR API const char * rtlsdr get device name (uint32 t index);
RTLSDR_API int rtlsdr_get_device_usb_strings (uint32_t index, char * manufact, char *
product, char * serial);
RTLSDR API int rtlsdr get index by serial (const char * serial);
RTLSDR API int rtlsdr open (rtlsdr dev t ** dev, uint32 t index);
RTLSDR API int rtlsdr close (rtlsdr dev t * dev);
RTLSDR_API int rtlsdr_set_xtal_freq (rtlsdr_dev_t * dev, uint32_t rtl_freq, uint32_t tuner_
frea):
RTLSDR_API int rtlsdr_get_xtal_freq (rtlsdr_dev_t * dev, uint32_t * rtl_freq, uint32_t *
tuner freq);
RTLSDR_API int rtlsdr_get_usb_strings (rtlsdr_dev_t * dev, char * manufact, char * product,
char * serial);
RTLSDR API int rtlsdr write eeprom (rtlsdr dev t * dev, uint8 t * data, uint8 t offset,
uint16 t len);
RTLSDR API int rtlsdr read eeprom (rtlsdr dev t * dev, uint8 t * data, uint8 t offset, uint16
t len);
RTLSDR API int rtlsdr set center freq (rtlsdr dev t * dev, uint32 t freq);
RTLSDR API uint32 t rtlsdr get center freq (rtlsdr dev t * dev);
RTLSDR API int rtlsdr set freq correction (rtlsdr dev t * dev, int ppm);
RTLSDR_API int rtlsdr_get_freq_correction (rtlsdr_dev_t * dev);
enum rtlsdr tuner {
    RTLSDR TUNER UNKNOWN = 0,
    RTLSDR TUNER E4000,
    RTLSDR TUNER FC0012,
    RTLSDR TUNER FC0013,
    RTLSDR TUNER FC2580,
    RTLSDR TUNER R820T,
    RTLSDR TUNER R828D
};
RTLSDR_API enum rtlsdr_tuner rtlsdr_get_tuner_type (rtlsdr_dev_t * dev);
RTLSDR_API int rtlsdr_get_tuner_gains (rtlsdr_dev_t * dev, int * gains);
RTLSDR_API int rtlsdr_set_tuner_gain (rtlsdr_dev_t * dev, int gain);
RTLSDR_API int rtlsdr_set_tuner_bandwidth (rtlsdr_dev_t * dev, uint32_t bw);
RTLSDR API int rtlsdr get tuner gain (rtlsdr dev t * dev);
RTLSDR API int rtlsdr set tuner if gain (rtlsdr dev t * dev, int stage, int gain);
RTLSDR API int rtlsdr set tuner gain mode (rtlsdr dev t * dev, int manual);
RTLSDR API int rtlsdr set sample rate (rtlsdr dev t * dev, uint32 t rate);
```

```
RTLSDR API uint32 t rtlsdr get sample rate (rtlsdr dev t * dev);
RTLSDR API int rtlsdr set testmode (rtlsdr dev t * dev, int on);
RTLSDR API int rtlsdr set agc mode (rtlsdr dev t * dev, int on);
RTLSDR API int rtlsdr set direct sampling (rtlsdr dev t * dev, int on);
RTLSDR API int rtlsdr get direct sampling (rtlsdr dev t * dev);
RTLSDR API int rtlsdr set offset tuning (rtlsdr dev t * dev, int on);
RTLSDR API int rtlsdr get offset tuning (rtlsdr dev t * dev);
RTLSDR API int rtlsdr reset buffer (rtlsdr dev t * dev);
RTLSDR API int rtlsdr read sync (rtlsdr dev t * dev, void * buf, int len, int * n read);
typedef void( * rtlsdr read async cb t) (unsigned char * buf, uint32 t len, void * ctx);
RTLSDR API intrtlsdr wait async (rtlsdr dev t * dev, rtlsdr read async cb t cb, void * ctx);
RTLSDR_API int rtlsdr_read_async (rtlsdr_dev_t * dev, rtlsdr_read_async_cb_t cb, void * ctx,
uint32_t buf_num, uint32_t buf_len);
RTLSDR_API int rtlsdr_cancel_async (rtlsdr_dev_t * dev);
RTLSDR API int rtlsdr set bias tee (rtlsdr dev t * dev, int on);
RTLSDR_API int rtlsdr_set_bias_tee_gpio (rtlsdr_dev_t * dev, int gpio, int on);
# ifdef __cplusplus
}
#endif
#endif / * RTL SDR H * /
```

rtlsdr.h头文件定义了 34 个接口函数。常用的接口函数有 10 个,如 rtlsdr_open(获取 设备句柄)函数、rtlsdr_set_center_freq(设置中心频率)函数、rtlsdr_set_sample_rate(设置 采样率)函数、rtlsdr_read_async(同步读取 I/Q 数据)函数、rtlsdr_close(关闭设备句柄)函数等。从头文件的函数声明中,可以查到函数输入参数和输出参数的名称和类型。

5.1.2 RTL-SDR 接口函数封装

LabVIEW 编程中,提供了一套调用动态链接库的方法。根据头文件提供的函数声明, 就可以利用 CLF 模块将头文件中的接口函数封装成子 VI,接下来将以 RTLSDR_API int rtlsdr_open(rtlsdr_dev_t ** dev, uint32_t index)函数为例说明动态链接库的封装和调 用过程。

(1) 准备两个文件,一个是 RTL-SDR 的库文件 rtlsdr. dll,另一个是头文件 RTL-SDR. h。 需要特别注意的是,如果主机安装的 LabVIEW 是 32 位的,库文件 rtlsdr. dll 必须是 32 位 的,否则 LabVIEW 将无法调用 rtlsdr. dll 中的函数。

(2) 新建一个 VI,在 Connectivity→Libraries & Executables 路径下找到 CLF 模块,如
 图 5-2 所示。通过 CLF 模块,就可以实现动态链接库的调用。

Data Communication	•
Connectivity	
Addons	- 🏳 Connectivity
Select a VI	Libraries & Executables
FPGA Interface	
Measurement I/O	Libraries & Executables
*	
	.NET Call Library System Exe Vindows R
	, ,

图 5-2 调用库函数(CLF)模块

(3) 在程序框图中创建一个 CLF 模块,双击该模块,就会弹出该模块的参数配置对话框,在这个对话框中,可以配置 DLL 文件路径、被调函数名、输入和输出参数等,如图 5-3 所示。

🖥 Call Librar	y Function			
Function	Parameters	Callbacks	Error Checking	
Library (C:\Prog 2013\in Speci Function rtlsdr_c	name or path gram Files (x86) str.lib\rtlsdr\rti ify path on diag n name open	\\National Ins Isdr.dll gram	truments\LabVIEW	Thread Run in UI thread Run in any thread Calling convention stdcall (WINAPI) C
Function pr	ototype			
void rtlsdr	_open(void);			
Consider u	sing a wizard i	nstead		OK Cancel Help

图 5-3 CLF 参数配置

在 Function 选项卡中,配置被调用函数的信息。在 Library name or path 文本框中设置 rtlsdr. dll 的路径(注意选择 rtlsdr. dll 文件的存放路径)。在 Function name 下拉列表中选择需要调用的接口函数,本例中选择 rtlsdr_open。在 Thread 选项中选择运行的线程范围,本例中默认选择 Run in UI thread。在 Calling convention 选项中选择被调用函数的调用约定,本例中默认选择 C。

在 Parameters 选项卡中,根据头文件中的函数声明对输入和输出参数进行设置。需要特别注意的是,为了正确设置这些参数,需要参考 rtl-sdr.h 头文件中的函数声明。例如,在 rtl-sdr.h 中,可以找到 rtlsdr_open 函数的输入输出参数以及它们的类型:

RTLSDR_API int rtlsdr_open (rtlsdr_dev_t ** dev, uint32_t index);

在 Parameters 选项卡中,默认创建了一个 return type 参数,如图 5-4 所示。根据头文件,rtlsdr_open 函数返回值是 int 类型,在图 5-4 的参数设置 Type 下拉列表中选择 Numeric 类型,在 Data type 下拉列表中选择 Signed 32-bit Integer 类型。

在 CLF 的 Parameters 选项卡中,还需要创建两个参数,一个是指向设备的句柄指针, 另一个是设备索引。选中左侧列表中的 DevRefnum,单击页面中的 **还** 按钮,在 Current parameter 栏中设置参数,Name 可以自定义,如设置为 DevRefnum,Type 选择为 Numeric, Data type 选择为 Unsigned Pointer-sized Integer, Pass 选择为 Point to Value,如图 5-5 所示。

以同样的方式创建设备索引 index。注意 index 是一个输入参数。Name 设置为 index, Type 选择为 Numeric, Data type 选择为 Unsigned 32-bit Integer, Pass 选择为 Value。 DevRefnum 和 index 均创建完成之后,在页面左下角的 Function prototype 区域可以看到

🛂 Call Library Fur	nction					×
Function Para	ameters Callbacks	Error Checking Current parameter	Name return Type Num onstant ta type Signe	n type eric ed 32-bit Integer	Y	
Function prototy int32_t rtlsdr_op Consider using a	pe en(uintptr_t *DevRefnu a wizard instead	m, uint32_t index);		ОК	Cancel	lelp

图 5-4 Parameters 选项卡

Function	Parameters	Callbacks	Error Checking		
return f DevRef index	type num	▲ ★ ★ ★	Current parameter Nam Typ Consta Data typ Par	e DevRefnum e Numeric ht Unsigned Pointer-sized Integer s Pointer to Value	>
unction pr ht32_t rtlsd	ototype Ir_open(uintptr	vt *DevRefnu	m, uint32_t index);		

图 5-5 DevRefnum 配置

函数声明,此声明应当与头文件中的函数保持一致。

(4) CLF 模块配置完成之后,需要定义输入和输出端口,在 CLF 的输入和输出端口分 别创建数值输入控件和显示控件,如图 5-6 所示。需要注意的是,在前面板中需要完成端口 关联,才可以作为子 VI 被其他 VI 调用。

(5)按照同样的方法,可以对 rtl-sdr.h头文件中的其他函数进行封装。将需要的函数 都封装完成之后,可以将封装后的子 VI 打包成一个库文件,方便维护。执行 File→New 菜 单命令,新建一个 Library,如图 5-7 所示。

然后在库文件名上右击,在弹出的菜单中选择 Add→File,选择已经封装好的 VI。更改库 文件名,就完成了库文件的创建,如图 5-8 所示。需要注意的是,rtlsdr. dll 文件和 rtlsdr. lvlib

112 ◀ 軟件无线电入门教程——使用LabVIEW设计与实现







图 5-7 Library 创建

File Edit View Items Files	Project Operate Tools	Wind	ow Help	File Edit View Project Operate Tools Window Help
- C Unitided U	New Add Save Find Show Error Window Multiple Variable Editor Arrange By Properties		File Folder (Snapshot) Hyperlink	riteris riteris close.vi copen.vi read sync.vi read sync.vi set agc mode.vi set center freq.vi set center freq.vi set freq correction.vi set streq correction.vi set stuner gain mode.vi

图 5-8 rtlsdr. lvlib 创建

一般放在同一个文件夹下。

最后将整个库文件夹复制到 LabVIEW 的\instr. lib 文件夹中,这样,在程序框图的函数选板中,就可以直接调用相应的子 VI,如图 5-9 所示。



图 5-9 RTL-SDR 子 VI

5.2 导入共享库向导

手动封装动态链装库中的接口函数是比较麻烦的,尤其是在库文件中的函数比较多的时候。LabVIEW提供了自动封装的方法——导入共享库向导。接下来,本节将以 rtlsdr. dll 为例,介绍基于导入共享库向导的封装方法。

5.2.1 导入前准备

在使用导入共享库向导之前,需要准备两个文件,一个是 RTL-SDR 的库文件 rtlsdr. dll, 另一个是头文件 RTL-SDR. h。

5.2.2 导入共享库向导过程

(1) 执行 Tools(工具)→Import(导人)→Shared Library(.dll)...(共享库)菜单命令,如图 5-10 所示。

🔯 LabVIEW		- 🗆 X
File Operate	Tools Help	
	Measurement & Automation Explorer Instrumentation	Ceamh O
	Merge Security User Name	Open Existing
	Convert Build Script Source Control	Show Vis
Blank Proj	LLB Manager	receiver.vi
Simple NI-	Import	.NET Controls to Palette laseband-Sine) .vi
	Shared Variable Distributed System Manager	ActiveX Controls to Palette) - Measurement.vi Shared Library (.dll) c).vi
-	Find VIs on Disk	Web Service

图 5-10 导入共享库向导

(2) 在弹出的对话框中选择 Create VIs for a shared library,如图 5-11 所示。如果需要 对已经生成的 VI 进行更新,选择第 2 项 Update VIs for a shared library。

Þ	Import Shared Library			×
	Specify Create or Update M	ode		ATIONAL STRUMENTS
	Create VIs for a shared li Creates VIs based on the Update VIs for a shared Updates previously impo	brary · header file and shared library file library rted VIs for the following project l	you provide. ibraries	
	Project	DLL File	Date	^
	rtlsdr.lvlib	rtlsdr.dll	15:15:33 02/01/2020	
	plutosdr.lvlib	libiio.dll	18:17:19 11/25/2019	
	hackrf.lvlib	hackrf.dll	14:27:37 11/23/2019	

图 5-11 创建或更新模式

114 ◀II 软件无线电入门教程——使用LabVIEW设计与实现

(3) 进入路径配置对话框,设置 rtlsdr. dll 的路径和头文件的路径。注意这里设置的头 文件是 rtl-sdr. h,如图 5-12 所示。

Import Shared Library	×
Select Shared Library and Header File	NATIONAL INSTRUMENTS
Shared Library (.dll) File D:\SDRDriver\RTLSDR\rtlsdr.dll	
Shared library file is not on the local machine	
Header (.h) File	
D:\SDRDriver\RTLSDR\rtl-sdr.h	

图 5-12 选择库文件和头文件

(4) 用记事本打开 rtl-sdr. h,可以看到头文件 rtl-sdr. h 还包含其他两个头文件:

```
# include < stdint.h>
# include < rtl - sdr_export.h>
```

在 Include 文件夹中, 配置两个头文件 stdint. h 和 rtl-sdr_export. h(本书配套程序)的路径, 如图 5-13 所示。

Import Shared Library	X
Configure Include Paths and Preprocessor Definitions	
Include D:\SDRDriver\RTLSDR	

图 5-13 Include 文件夹

(5)如果路径配置正确,共享库向导就会根据库文件和头文件生成一个函数列表。正常情况下,头文件中的33个接口函数能够被识别和封装,如图5-14所示。对于FM接收机,只会用到其中的少部分函数,如rtlsdr_open()、rtlsdr_set_center_freq()、rtlsdr_set_sample_rate()函数等。

(6) 选定需要生成的函数之后,设置 LabVIEW 库函数的文件名和保存路径。本例中 设置文件名为 rtlsdr,路径为 C 盘安装文件夹\user. lib 下,如图 5-15 所示。

(7) 在 Select Error Handing Mode 页面中,可以配置错误输出模式,如图 5-16 所示。 有 3 种模式可以选择,本例中选择 Simple Error Handing。

Select Functions to Convert The shared library contains 34 function(s). The declarations of 33 function(s) are found and rec the header file and these function(s) can be wrapped. The remaining function(s) cannot be wray you want to import these functions, please review the warning messages next to the functions le will need to fix the problems before you can continue with the wizard. The rtlsdr_cancel_async 0 Ttlsdr_cancel_async 0 Ttlsdr_get_center_freq 0 Ttlsdr_get_device_count 0 Ttlsdr_get_device_usb_strings 0 Ttlsdr_get_device_usb_strings 0 Ttlsdr_get_index_by_serial 0 Ttlsdr_get_index_by_serial 0 Ttlsdr_get_tuner_gain 0 Ttlsdr_get_tuner_gain 0 Ttlsdr_get_uner_gain 0 Ttlsdr_get_uner_gain 0 Ttlsdr_get_uner_gain 0 Ttlsdr_get_uner_gain 0 Ttlsdr_get_uner_gain 0 Ttlsdr_get_tuner_gain 0 Ttlsdr_g	×			Magnet Shared Library
The shared library contains 34 function(s). The declarations of 33 function(s) are found and rec the header file and these function(s) can be wrapped. The remaining function(s) cannot be wray you want to import these functions, please review the warning messages next to the functions I will need to fix the problems before you can continue with the wizard. Titsdr_cancel_async 0 Titsdr_close 0 Titsdr_get_center_freq 0 Titsdr_get_device_count 0 Titsdr_get_device_name 0 Titsdr_get_device_usb_strings 0 Titsdr_get_device_usb_strings 0 Titsdr_get_index_by_serial 0 Titsdr_get_index_by_serial 0 Titsdr_get_timer_gain 0 Titsdr_get_utner_gain 0 Titsdr_get_utner_gain 0 Titsdr_get_timer_gain 0 Titsdr_get_timer_gain 0 Titsdr_get_timer_gain 0 Titsdr_get_timer_gain 0	NATIONAL INSTRUMENTS	NATIO INSTRU		Select Functions to Convert
<pre>✓ rtlsdr_cancel_async 0 ✓ rtlsdr_close 0 ✓ rtlsdr_get_center_freq 0 ✓ rtlsdr_get_device_count 0 ✓ rtlsdr_get_device_name 0 ✓ rtlsdr_get_device_usb_strings 0 ✓ rtlsdr_get_device_usb_strings 0 ✓ rtlsdr_get_freq_correction 0 ✓ rtlsdr_get_index_by_serial 0 ✓ rtlsdr_get_offset_tuning 0 ✓ rtlsdr_get_effset_tuning 0 ✓ rtlsdr_get_uner_gain 0 ✓ rtlsdr_get_tuner_gain 0 ✓ rtlsdr_get_uner_gain 0 ✓ r</pre>	gnized in A ped. If elow. You	ction(s) are found and recognized function(s) cannot be wrapped. If iges next to the functions below. Yo d.	of 33 funct maining fi g messag e wizard.	The shared library contains 34 function(s). The declarations of 3 the header file and these function(s) can be wrapped. The rema you want to import these functions, please review the warning r will need to fix the problems before you can continue with the w
 ✓ rtlsdr_read_async () ✓ rtlsdr_read_eeprom () ✓ rtlsdr_read_sync () 	c(void	rtlsdr_cancel_async long * rtlsdr_cancel_async(void *dev);	^	<pre> rtlsdr_cancel_async () rtlsdr_close () rtlsdr_get_center_freq () rtlsdr_get_device_count () rtlsdr_get_device_name () rtlsdr_get_device_usb_strings () rtlsdr_get_direct_sampling () rtlsdr_get_index_by_serial () rtlsdr_get_index_by_serial () rtlsdr_get_index_by_serial () rtlsdr_get_sample_rate () rtlsdr_get_tuner_gain () rtlsdr_get_usb_strings () rtlsdr_get_async () rtlsdr_get_async () rtlsdr_read_exprom () rtlsdr_read_sync () </pre>

图 5-14 选择需要封装的函数

Import Shared Library	×
Configure Project Library Settings	
Project Library Name (.lvlib)	
rtlsdr	
Project Library Path	
C:\Program Files (x86)\National Instruments\LabVIEW 2013\user.lib\rtlsdr	a
☑ Copy the shared library file to the destination directory.	

图 5-15 设置库文件名和保存路径

(8) 在 Configure VIs and Controls页面中,可以对每个函数的输入和输出参数进行重 配置。例如,选择 rtlsdr_set_center_freq()这个函数,在默认的情况下,Control Type 设置 为 Numeric,Pass Type 设置为 Pass by Value,Representation 设置为 Unsigned Long,如 图 5-17 所示。



图 5-16 配置错误输出模式



图 5-17 重新配置输入输出参数

(9) 配置完成每个函数的输入和输出参数,单击 Next 按钮,就可以进行封装。等待几分钟,将返回封装后的 rtlsdr 库,如图 5-18 所示。



图 5-18 封装后的 rtlsdr 库

(10)检查接口函数封装的正确性。打开任意一个子 VI,如 set center freq. vi。进一步 打开 CLF,可以看到其配置,如图 5-19 所示。

🛂 Call Librar	y Function					×
Function return t dev free	Parameters type	Callbacks	Error Checking Current parameter	r Name	dev	
ineq		4 4	ſ	Type Constant Data type Pass	Numeric Unsigned 32-bit Integer Value	
Function pro int32_t rtlsd	ototype lr_set_center_fre	♥ eq(uint32_t d	ev, uint32_t freq);			
<u>Consider u</u>	sing a wizard in	nstead			OK Cancel He	lp

图 5-19 封装的正确性验证

5.3 动态链接库编译

在 5.1.2 节和 5.2.2 节中,不论是手动方式还是导入共享库向导的方式,都需要动态链 接库文件 rtlsdr.dll。接下来,本节将讨论如何从源程序中编译动态链接库文件。

5.3.1 编译前准备

首先在 GitHub 网站中搜索到 rtlsdr 的源文件,有多个版本可以选择,如可以选择 osmocom/rtl-sdr,网址为 https://github.com/osmocom/rtl-sdr。

在 rtl-sdr-master\src 文件下,可以找到 rtlsdr. dll 对应的源程序 librtlsdr. c。预先安装 pthread-win32 和 libusbx-1.0.18-win(本书配套程序)。预先准备编译依赖的静态库文件 libusb-1.0.lib 和 pthreadVC2.lib(本书配套程序)。

5.3.2 编译步骤

将 librtlsdr.c文件编译成动态链接库文件,需要预先安装 C 程序的编译软件,本例中选择 VS 2013。Microsoft Visual Studio 简称 VS,是美国微软公司提供的软件开发工具。librtlsdr.c 的编译步骤如下。

(1)首先启动 VS 2013,新建一个 Win32 项目,设置项目名称和保存路径,如图 5-20 所示。

新建项目						? ×
▶ 最近		.NET Framework 4.5	▼ 排序依据: 默认值		搜索已安装模板(Ctrl+E)	.م
▲ 已安装				Manual Cu u	类型: Visual C++	
▲ 模板	^	wins2 亞制百 [1] Win32 项目		Visual C++	用于创建 Win32 应用程序、控序、DLL 或其他静态库的项目	制台应用程
 Visual C# 						
▲ Visual C++						
ATL						
CLR						
常规						
MFC						
測试						
Visual F#						
SQL Server						
TypeScript						
Python						
▷ 其他项目类型						
建模项目						
示例	-		約十山山にいませの計本も	1840		
▶ 联机			里古瓜处以联机开重找	<u> </u>		
名称(<u>N</u>):	librtlsdr					
位置(L):	D:\VS2013Test	۱		•	浏览(B)	
解决方案(S):	创建新解决方案			•		
解决方案名称(M):	librtlsdr				✓ 为解决方案创建目录(D)	
					□ 添加到源代码管理(U)	
					<u> </u>	同時当
					WEAL	-10/15

图 5-20 新建一个 Win32 项目

(2) 在弹出的应用程序向导中选择应用程序类型,本例中选择 DLL,在附加选项中选择

"空项目"。

(3) 右击"源文件"文件夹,在弹出的快 捷菜单中依次选择"添加"→"现有项",将 rtl-sdr-master\src 文件夹下的 librtlsdr.c、 tuner_e4k.c、tuner_fc0012.c、tuner_fc0013.c、 tuner_fc2580.c、tuner_r82xx.c6个C文件 导入,如图 5-21 所示。

(4) 将静态库文件 libusb-1.0. lib 和 pthreadVC2. lib 添加到 VS 的 VC++目录 中,如图 5-22 所示。右击图 5-21 所示的项 目名称 rtlsdrdll,在弹出的菜单中选择"配置 属性",在"VC++目录"页面中配置包含目录 和库目录。

解决方案	资	原管理器 ▼ ₽ ×
00	۵	`o - ≠ 司 🗿 🕨 🗕
搜索解决	坊	案资源管理器(Ctrl+;)
■ 解 ▲ 国	大方 rtl: 調	案 "rtlsdrdll" (1 个项目) sdrdll ↓ 项目名称 头文件 外部依赖项
4		源文件
1	>	*+ librtlsdr.c
1	>	*+ tuner_e4k.c
1	>	*+ tuner_fc0012.c
1	>	++ tuner_fc0013.c
1	>	*+ tuner_fc2580.c
1	>	++ tuner_r82xx.c
	ţ.	资源文件

图 5-21 解决方案资源管理器

rtlsdrdll 属性页		? 🗙
配置(C): 活动(Debug)	▼ 平台(P): 活动	(Win32) ▼ 配置管理器(O)
 ▶ 通用属性 ▲ 配置属性 常規 调试 VC++目录 ▶ C/C++ > 链接器 ▶ 清单工具 ▶ XML 文档生成器 ▶ 浏览信息 ▶ 生成事件 ▶ 自定义生成步骤 ▶ 代码分析 	 常規規 可执行文件目录 包含目录 <u>引用目录 库目录 驳们可以行文件目录 联合目录 联目录 联目录 </u>	\$(VC_ExecutablePath_x86);\$(WindowsSDK_Executable C:\libraries\pthreads-include\Pre-built_2\include;C \$(VC_ReferencesPath_x86); C:\libraries\pthreads-include\Pre-built_2\lib\x86;C \$(WindowsSDK_MetadataPath); \$(VC_SourcePath); \$(VC_SourcePath); \$(VC_IncludePath);\$(WindowsSDK_IncludePath);\$(MSI
		施定 取消 应用(A)

图 5-22 配置包含目录和库目录

编辑包含目录,将 Pre-built. 2\include 文件夹、libusbx-1.0.18-win 文件夹下的 include\ libusbx-1.0 和 rtl-sdr-master\include 文件夹添加到包含目录中,如图 5-23 所示。

包含目录		8	×
		* 🗙 🗸	
C:\librarie	s\pthreads-inc	:lude\Pre-built.2\include	
C:\librarie	s\libusb-includ	le\libusbx-1.0.18-win\include\libusbx-1.0	
C:\rtl-sdr	·master\include	e	+
•	III		•
•			
继承的值:			
迷承的值: \$(VC_Inclu	ıdePath)		*

图 5-23 配置包含目录

编辑库目录,将 Pre-built. 2\lib\x86 文件夹、libusbx-1.0.18-win\MS32\static 文件夹 添加到库目录中,如图 5-24 所示。

库目录	? ×
C:\libraries\pthreads-include\Pre-built.2\lib\x86 C:\libraries\libusb-include\libusbx-1.0.18-win\MS32\static	*
<	•
继承的值:	
\$(VC_LibraryPath_x86) \$(WindowsSDK_LibraryPath_x86)	^

图 5-24 配置库目录

在"链接器"→"附加依赖项"中添加 libusb-1.0. lib 和 pthreadVC2. lib 静态库文件,如 图 5-25 所示。

		rtlsdrdll 属性页
配置管理器(O)	 ▼台(P): 活动(Win32) ▼ 配置管理器(O) 	配置(C): 活动(Debug)
潮값 附加依較项 VC++目录 ②略所有武以本 ● C/C++ ③略所有武以本 ● 性接路 一個快速 常規 ● 供表: 電規 ● 供表: 第規 ● 供表: 「清单文件 ● 出表: 「湯菜、 ● 出表: 「小村、 ● 出表: 「小村、 ● 出表: 「小村、 ● 出表: 「小村、 ● 日本: 「小村、 ● 日本: ● 日本: ● 日本:	附加依赖项 Iibusb-1.0.lib;pthreadVC2.lib;%(AdditionalDepend 忽略所有默认库 附加依赖项 ? 忽略符定默认库 模块定义文件 将模块添加到程序集 Iibusb-1.0.lib 材描体物可 pthreadVC2.lib ^ 建了集链接资源 * * 继承的值: * * 修加休赖项 # * 增序集链接资源 * * 增加休赖项 * * 增速 * * 增加休赖项 * * 增速32.lib * * 增定 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *	週试 VC++目录 ▷ C/C++ ▲ 链接器 常規 輸入 清单文件 调试 系统 优化 碳入的 IDL Windows 元数据 高级 所有选项 命令行 ▷ 清单工具 ▼

图 5-25 在链接器中添加附加依赖项

(5)编译生成动态链接库 rtlsdrdll. dll, VS 编译输出如图 5-26 所示。需要特别注意, VS 默认生成的动态链接库是 32 位的。

输出			
显示输出来源(S):	生成	• <u>•</u> • • •	** 2
1≻ 已启动: 1> LINK:没有: 1> 正在创建: 1> rtlsdrdll.v ========= 生成	主成: 项目: rtlsdrdll, 配罟: Debug Win32 戌到 D:\VS2013Test\rtlsdrdll\Debug\rtlsdrdl 库 D:\VS2013Test\rtlsdrdll\Debug\rtlsdrdl. cxproj -> D:\VS2013Test\rtlsdrdll\Debug\rtls : 成功 1 个,失败 0 个,最新 0 个,跳过 0	 L dll 或上一个増里链接没有 lib 和对象 D:\VS2013Test\r sdrdll.dll 个 ========	生成它: 正在执行完全链接 tlsdrdll\Debug\rtlsdrdll.exp ,

5.4 调用库函数的配置

在接口函数封装过程中,使用了 CLF 模块。接下来,本节将对 CLF 配置对话框中的 Function(函数)、Parameters(参数)、Callbacks(回调)、Error Checking(错误检测)4个选项 卡加以说明。

5.4.1 函数配置

在 Function(函数)选项卡中,可以设置 DLL 文件路径。在 Library name or path(库 名/路径)文本框中,单击文件夹图标,选择 DLL 所在的路径,就可以完成设置。当调用包含 DLL 的 VI 被装入内存时,DLL 被自动装入内存。在 Library name or path 文本框的下方, 还有一个 Specify path on diagram(在程序框图中指定路径)选项,选中该选项,DLL 将会被 "动态加载":只有程序运行到需要使用该 DLL 中的函数时,DLL 才会被装入内存。需要注 意的是,选中 Specify path on diagram 之后,Library name or path 中输入的路径将无效。 与此同时,CLF 模块将多出一对输入和输出端口,用于指定 DLL 的路径。

设置完 DLL 文件路径之后,在 Function name(函数名称)下拉列表中选择被调用函数的名称。单击下拉列表,将显示 DLL 中所有的可用函数。

在 Function 选项卡中可以配置 Thread(线程)。在 Thread 选项下,有 Run in UI thread 和 Run in any thread 两个单选按钮。Run in UI thread 表示仅运行在 UI 线程中; Run in any thread 表示运行在任意线程中。在本例的程序中,选择默认的 Run in UI thread。

在 Function 选项卡中还可以指明被调用函数的 Calling convention(调用约定)。CLF 支持两种调用约定: stdcall 和 C。stdcall 由被调用者负责清理堆栈,C由调用者清理堆栈。 Windows API 一般使用的都是 stdcall,标准 C 的库函数则大多使用 C。

5.4.2 端口参数配置

在 CLF 配置中,端口参数(Parameters)的配置比较复杂。DLL 调用出现的问题,大多 是由于 Parameters 配置错误所引起的。接下来,本节将通过 rtlsdr 实例介绍该页面的 配置。

在 Parameters 选项卡中,可以添加相应的参数并修改它们的返回值类型,直到页面底 部的 Function prototype(函数原型)与 DLL 头文件中的函数定义相匹配,如图 5-27 所示。 在 Type(类型)下拉列表中选择函数返回值的类型,如 Void(空)、Numeric(数值)或 String (字符串)。本例中选择 Numeric。

在 Data type 下拉列表中可以看到多种数据类型,如图 5-28 所示。本例中选择 Signed 32-bit Integer。

这里需要特别指出的一种数据类型,就是C语言中的指针类型,如Signed Pointer-sized Integer。对于简单的C函数构成的DLL,LabVIEW利用CLF调用时就比较简单,但是对于参数或返回值是指针类型的DLL函数,LabVIEW调用时就比较复杂。

指针就是变量的地址,将地址作为参数传递到 DLL 函数中,DLL 函数就可以操作这个

122 ◀ 軟件无线电入门教程——使用LabVIEW设计与实现

Function Pa	rameters	Callbacks	Error Checking				
return type		•	Current parameter				
DevRefnum			Name	re	eturn type		
Index		×	Туре	N	lumeric	~	
		-	Constan	t 🗆]		
		Ŷ	Data type	Si	igned 32-bit Integer	~	
		Ŷ			Ļ		
			Ту	pe	Numeric		~
		~	Const	ant	Void		
			Data tv	ne	✓ Numeric		
Function protot	ype		Duta ty	pe	String		
int32_t rtlsdr_op	pen(uintptr_	t *DevRefnu	m, uint32_t index);				

图 5-27 Parameters 选项卡



图 5-28 Data type 下拉列表

地址指向的变量。需要注意的是,在 32 位的操作 系统中,可以使用 int32 数值表示指针。在 64 位 的系统中,只能使用 I64 或 U64 表示指针。如果 无法确知是 32 位还是 64 位的系统,则可以使用 Pointer-sized Integer 这种数据类型。

CLF 中常用指针类型配置和使用如表 5-1 所示,指向数值类型和字符类型的指针配置相对简单,找到与之相对应的选项即可。在使用布尔类型时,由于布尔类型在 DLL 函数和 LabVIEW VI 之间传递没有对应的数据类型,需要利用数值类型来传递,因此输入时先要把布尔量转换为数值,

再传递给 DLL 函数,输出时再把数值转换为布尔量。需要注意的是,如果在 C 语言函数参数声明中有 const 关键字,需要选中 Constant 选项。

C语言声明	CLF 配置	CLF 使用
double * a	Type Numeric Constant Data type 8-byte Double Pass Pointer to Value	DEL DEL
char *a	Type String	
bool * a	Type Adapt to Type V Constant Data format Handles by Value V	

表 5-1 CLF 中指针配置和使用

数组在传递给 DLL 函数时,只能是指针,如表 5-2 所示。在传递数组类型时,Array format(数组格式)要选择为 Array Data Pointer(数组数据指针)。需要注意的是,LabVIEW 只支持 C 语言中的数值型数组。

C语言声明	CLF 配置	CLF 使用
int a[]	Type Array Constant Data type Signed 32-bit Integer Dimensions Array format Array Data Pointer Winimum size <none> V</none>	<u>n 🗄 n</u>
int *a[]	Type Array	

表 5-2 CLF 中数组配置和使用

簇结构在 LabVIEW 中是常用的数据类型,C 语言的 struct(结构体)数据类型与之对应。在 CLF 节点的配置面板中,没有专门命名为 struct 或 cluster 参数类型,选择 Adapt to Type 就可以。需要注意的是,在 cluster 较为复杂的情况下,需要考虑字节对齐问题。

5.4.3 回调函数配置

Callbacks(回调)为 DLL 设置一些回调函数。DLL 文件中实现了各种类型的函数,当 程序需要调用函数时,就要先载入 DLL,取得函数地址然后进行调用。有些情况下,需要将 应用程序的某些功能提供给 DLL 使用,于是就可以使用回调函数。回调函数通常用于程序 初始化、资源清理等工作。

5.4.4 错误检测配置

Error Checking(错误检测)用于设置错误处理方式,有 Maximum(最高级)、Default(默认)和 Disabled(不检测)3个选项。最高级别的检测会对运行过程中检测到的每处错误进行反馈,但同时也会使 CLF 节点的运行速率降低。默认级别的检测会对程序执行最小程度的错误进行排查,对 CLF 节点的运行速率影响较小。不检测即不对程序中的错误进行检测,此时 CLF 节点以最快速度运行。

5.5 本章小结

本章以 RTL-SDR 为例,介绍了非 NI 设备控制方法。通过 CLF 模块,LabVIEW 可以 调用 DLL 中的接口函数,从而实现对设备的控制。

首先以 RTL-SDR 的 LabVIEW 接口函数封装过程为例,介绍了 LabVIEW 中调用库函数模块的使用方法。

然后详细介绍了 LabVIEW 中导入共享库向导的使用方法。

接着介绍了动态链接库文件 rtlsdr. dll 在 VS 2013 环境下的编译过程。

最后介绍了调用库函数模块中 Fuction、Parameters、Callbacks、Error Checking 4 个选项卡的配置方法。