软件参考手册

（RapidIO 高速总线仿真卡）

Tel：(86)-28-8514 8273　Fax：(86)-28-8514 8287　**Http://www.enpht.com**
成都恩菲特科技有限公司

目 录

第1章. 版权声明

第2章. 更新版本

所有版本和手册更新及发行时间都列举在下表。手册的初始版本是 Ver1.00。不论何时更新手册，版本号都在尾数加 1。当更新涉及到较为重要的内容时，版本号中间的数加 1，当更新涉及到核心内容时，版本号第一位数加 1。更新的内容通过手册发行，包括修改的内容及对手册新增加的内容。新版本均包括了对旧版本更改的内容。每个新版本或更新后版本都有一页标注该文献的更改情况。

版本号	日期	硬件版本	版本说明	备注
Ver 1.00	2018.11	V1.00	初稿	
Ver 1.01	2019.12	V1.01	1、 增加处理消息类型数据接口； 2、 接口名称优化调整；	
Ver 1.10	2020.8	V1.10	1、 增加本地 ID 获取函数； 2、 更改中断注册函数名字； 3、 增加获取 1X 或 4X 模式函数； 4、 增加块写函数	

第3章. 函数开发环境

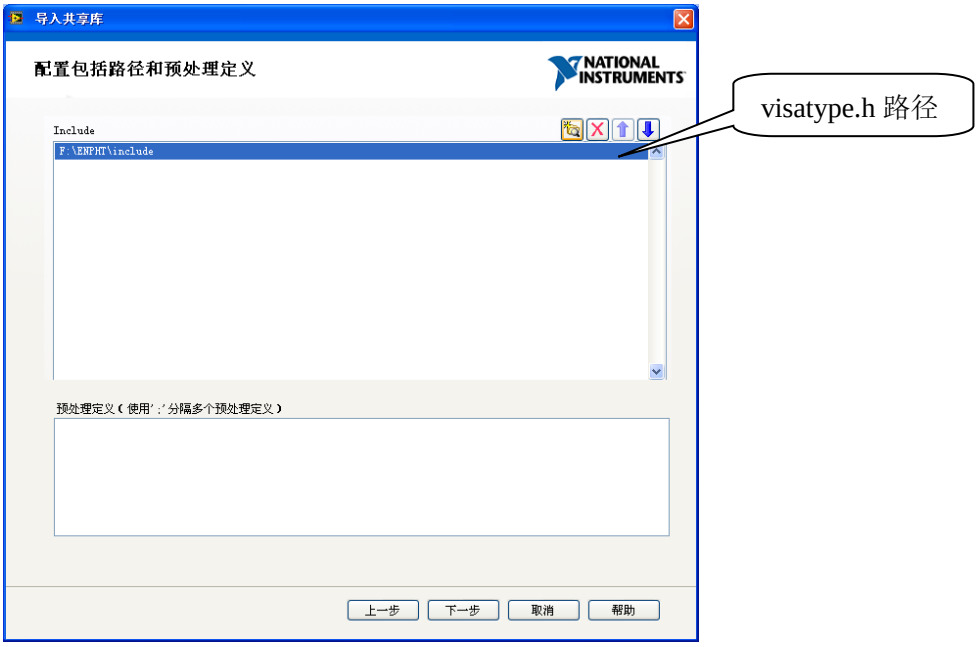
在本手册中描述的函数为 C 和 C++格式。此 API 函数可以使用在 Windows 环境下所有 32Bit 开发工具和图形化开发工具。例如，您可以在 VC++6.0、Labwindows/CVI、Labview、VB、Borland C++等多种编译环境下使用我们提供的开发库进行二次开发。开发库包括头文件（*.h）、库文件（*.lib）和动态库文件（*.dll）。下面介绍在各种编译环境下使用开发库的方法。

- **Visual C++ & Labwindows/CVI**

首先，将开发库文件（*.h、*.lib、*.dll）文件拷贝在您的开发工程文件中；然后在您的源文件中包含头文件，如#include “EphRapidIO.h”；最后再将库文件(EphRapidIO.lib)加入你的工程中即可编译。

- **Labview**

在 Labview 下进行开发时，您可以将我们已经生成好的共享库放在你的 Labview 软件安装目录的 user.lib 目录中，也可以手动生成共享库。点击 Labview 的“工具->导入->共享库(.dll)”菜单功能将动态库导入即可。需要注意的是，在“配置包含路径和预处理定义”时，需指定头文件“visatype.h”的路径：



注：visatype.h 文件在产品配套光盘的 INC 文件夹中可以找到。

第4章. 模块 API Routines

《RIO 协议光纤通信仿真模块软件参考手册》是基于 RIO 协议光纤通信仿真模块函数库编写的。目的是为了让读者了解 API 函数库中的各类函数，便于用户在 API 函数库的基础上编程。其中各个函数的使用范例都是在 Labwindows/CVI7.0 平台下编写的，且较为简略，详细使用可参考与板卡配套的示例程序。

为了更方便、更直观的了解函数功能及使用方法，将 RIO 协议光纤通信仿真模块的 API 函数前缀统一为：EphRapidIO_，文中函数说明格式如下：

EphRapidIO_Close

函数名

函数功能

此函数关闭板卡连接释放资源。调用此函数同时对模块复位。

函数原型

参数类型

ViStatus _VI_FUNC EphRapidIO_Close (ViUInt32 cardnum);

参数说明

参数输入/输出方向
in:输入 out:输出

cardnum

[in] 模块句柄，由连接函数获取。

返回值

API_SUCCESS （函数调用成功）

API_FAILED （函数调用失败）

使用例程

```
ViUInt32 cardnum=0;
hr = EphRapidIO_Close (cardnum);
if((hr!=0)||(cardnum==0)) printf("模块连接失败");
```

第5章. 使用纲要

使用纲要主要对产品功能及软件功能进行一个概括性描述，使用户对产品的主要性能指标，以及如何使用 API 函数实现这些功能有一个整体的认知。

产品功能描述

EP-X6295 是恩菲特科技自主研发的多通道串行 RIO 协议光纤通信仿真模块，可用于 RapidIO 总线协议与 PCI-e 总线协议互相转换的应用场景。

软件功能描述

1、 关于 CardNum 参数

恩菲特板卡的每个 API 函数都有一个 cardnum 参数(个别 API 函数除外)，这是一个句柄参数，由板卡的连接函数 EphRapidIO_Open 函数获取后，供其他 API 函数使用。此参数非常重要，如果没有正确获得此参数，则表示软件没有获得对硬件的控制权，那么使用其他 API 函数对硬件进行操作是无效的。

使用此函数时，建议将其定义为一个静态的全局变量，并且将变量的初始化值赋值为 0，这样就可以通过判断该值是否大于 0，来确定板卡连接函数是否正确连接上板卡（即软件获得对硬件的控制权），如下所示：

```
ViUInt32 cardnum = 0;

hr = EphRapidIO_Open (TRUE, 3, 14, &cardnum);
if((cardnum >0) && (hr==0))
{
    //板卡连接成功
}
```

注意：在程序中，只需要调用一次连接函数以获得 cardnum 供其他函数使用，不需要每次使用其它 API 函数时，都调用一次连接函数。

2、 设备 ID

RapidIO 属于网络通信，需要远端设备地址才能将数据发往对方。而接收数据时需要验证消息中的地址是否匹配本设备的地址，不匹配时丢弃数据。

设备地址有效范围 0～255。

调用 EphRapidIO_SetNodeDeviceId()可以设置本设备的地址。

3、 门铃事件

RapidIO 的门铃事件，附带一个 16bit 数据，通常用于标示类型。通常门铃事件通常是配合消息、块传输，在传输完成时发送一条门铃事件，通知对方已经传输完成，相当于向通信远端发起一次中断。

调用 EphRapidIO_WriteDoorbell ()可以向通信远端发起门铃事件。

调用 EphRapidIO_ReadDoorbell ()可以读取已接收到的门铃事件数据。

4、 消息传输

RapidIO 的消息提供最多 2048 条消息缓存，调用 EphRapidIO_ReadMsg()读取已收到的消息内容。

调用 EphRapidIO_WriteMsg()可向远端发送一条消息。

5、 块传输

RapidIO 的块传输支持 8B～64MB，调用 EphRapidIO_NWrite ()或者 EphRapidIO_NWriteR()或者 EphRapidIO_NWriteEx()函数，向远端发送一块数据。其中 EphRapidIO_NWrite()函数仅发送数据，等待远端响应；EphRapidIO_NWriteR()函数发送每一包数据均等待远端响应后再发下一包数据；EphRapidIO_NwriteEx()函数仅等待远端响应最后一包或者两包数据。

调用 EphRapidIO_NRead ()从远端读取一包数据，最大 256 字节，且仅支持读取 8、16、32、64、128、256 几种长度的数据。

块传输的读写地址仅支持 64 字节的整数倍地址。

6、 本地数据

本地数据用于缓存远端发起的块传输数据，每通道 256MB，地址从 0 开始。

调用函数 EphRapidIO_MemRead ()时，可读取通道缓存数据。

调用函数 EphRapidIO_MemWrite ()时，可写入通道缓存数据。

7、 维护模式

维护功能暂时保留

调用 EphRapidIO_WriteMaintPkt()可写入维护模式数据。

调用 EphRapidIO_ReadMaintPkt()可读取维护模式数据。

第6章. 函数概要说明

本节对 API 函数进行概括性说明，使用户对 API 函数有一个总体的了解。

系统函数

EphRapidIO_Open	通过总线号和槽号，连接到指定 EP-X6295 模块并初始化
EphRapidIO_Close	关闭模块的连接，并释放资源
EphRapidIO_Reset	复位模块
EphRapidIO_GetManuID	获取厂商 ID，默认 0x41F8
EphRapidIO_GetDevID	获取设备 ID，默认 0x6295
EphRapidIO_GetVersion	获取模块的硬件、软件版本号
EphRapidIO_StatusToString	函数返回值到返回信息的转换
EphRapidIO_IntInstall	注册/注销中断

功能函数

EphRapidIO_SetNodeDeviceId	设备本地节点 ID
EphRapidIO_GetNodeDeviceId	获取设备本地节点
EphRapidIO_SetWorkMode	设置 1X 及 4X 工作模式，保留
EphRapidIO_GetWorkMode	获取 1X 及 4X 工作模式
EphRapidIO_MemRead	读本地存储器
EphRapidIO_MemWrite	写本地存储器
EphRapidIO_WriteMaintPkt	写维护数据
EphRapidIO_ReadMaintaiPkt	读维护数据
EphRapidIO_NWrite	不带响应地向远端发送一块数据
EphRapidIO_NWriteR	带响应地向远端发送一块数据

EphRapidIO_NWriteEx	扩展地向远端发送一块数据
EphRapidIO_NRead	从远端获取一块数据
EphRapidIO_WriteMsg	向远端发送一条消息
EphRapidIO_ReadMsg	取一条已接收的消息
EphRapidIO_WriteDoorbell	向远端发送一个门铃事件
EphRapidIO_ReadDoorbell	读取本地接收到的门铃数据
EphRapidIO_SetTimeOut	设置发送超时时间
EphRapidIO_SetMixMode	设置数据收发模式(混合模式 OR 过滤模式)
EphRapidIO_GetMixMode.	获取工作模式(混合模式 OR 过滤模式)
EphRapidIO_GetDoorBellCnt	获取接收缓存中的门铃个数
EphRapidIO_GetRecvMsgFrameCnt	获取接收缓存中的消息个数

函数详细说明

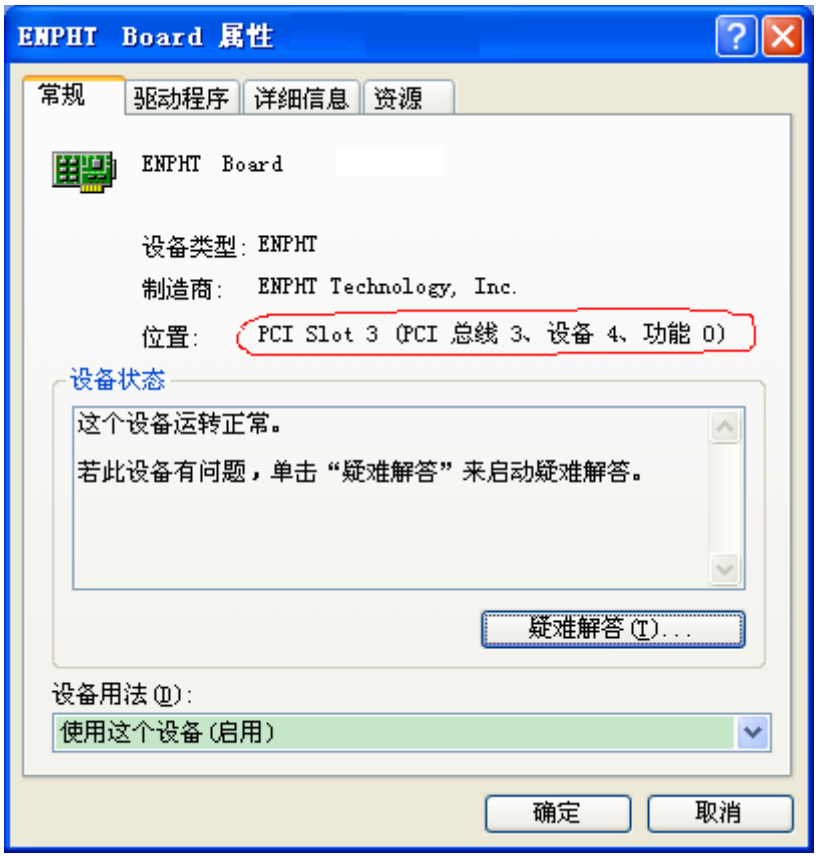
本章详细介绍 RIO 协议光纤通信仿真模块的 API 函数及使用方法。

EphRapidIO_Open

函数功能

当系统中有多张板卡时，需要根据总线号和设备号连接到指定板卡，并返回句柄以供其他函数使用。调用此函数同时对模块进行软件复位。

模块总线号和设备号可以通过计算机设备管理器查询。打开设备管理器，在模块属性中可以看到模块信息，如下图所示，“PCI Slot 3 (PCI 总线 3、设备 4、功能 0)”，此时的总线号是 3，设备号是 4。



此函数只需要调用一次即可，在调用其他函数时，需要使用此函数获得的板卡句柄。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_Open (ViUInt32 mode
                                   ViUInt32 busNum,
                                   ViUInt32 deviceNum,
                                   ViUInt32 *cardnum);
```

参数说明

mode

[in]模式

1 = 使用默认参数打开设备 0 = 使用指定的总线号设备号打开设备

busNum

[in] 模块总线号。

deviceNum

[in] 模块设备号。

cardnum

[out] 模块句柄，成功获取后供其他函数使用。

返回值

API_SUCCESS	（函数调用成功）
API_FAILED	（函数调用失败）
……	（其他错误）

使用例程

```
ViStatus hr = 0;
ViUInt32 busNum = 3;
ViUInt32 deviceNum = 4;

hr = EphRapidIO_Open (1, busNum, deviceNum, &cardnum);
if((hr==0) && (cardnum>0))
    printf("connect success")
```

EphRapidIO_Close

函数功能

此函数用于关闭连接，并释放连接模块时申请的系统资源。该函数在程序退出或关闭模块功能的时候使用。

函数原型

ViStatus _VI_FUNC EphRapidIO_Close(ViUInt32　cardnum);

参数说明

cardnum

[in]　模块句柄，由连接函数获得。

返回值

API_SUCCESS	（函数调用成功）
API_FAILED	（函数调用失败）
……	（其他错误）

使用例程

```
ViStatus hr=0;
ViUInt32 cardnum;


if (cardnum==0) return -1;
hr = EphRapidIO_Close (cardnum);
if(hr != 0)
    MessagePopup("err!","模块关闭失败！");
```

EphRapidIO_Reset

函数功能

此函数用于软件复位。在连接板卡函数中默认已调用一次此函数。

函数原型

ViStatus _VI_FUNC EphRapidIO_Reset(ViUInt32 cardnum);

参数说明

cardnum

[in] 模块句柄，由连接函数获得。

返回值

API_SUCCESS	（函数调用成功）
API_FAILED	（函数调用失败）
……	（其他错误）

使用例程

```
ViStatus hr=0;
ViUInt32 cardnum;

if (cardnum==0) return -1;
hr = EphRapidIO_Reset(cardnum);
if(hr != 0)
    MessagePopup("err!","模块初始化失败！");
```

EphRapidIO_GetManuID

函数功能

此函数获取厂商 ID，恩菲特厂商 ID 固定为 0x41F8。

函数原型

ViStatus _VI_FUNC EphRapidIO_GetManuID(

ViUInt32 cardnum,

ViUInt32 *manuID);

参数说明

cardnum

[in] 模块句柄，由连接函数获得。

manuID

[out] 厂商 ID。默认厂商 ID:0x41F8。

返回值

API_SUCCESS （函数调用成功）

API_FAILED （函数调用失败）

使用例程

```
ViStatus hr=0;
ViUInt32 manuID=0;
ViUInt32 cardnum;

 if (cardnum==0) return -1;
 hr = EphRapidIO_GetManuID(cardnum,&manuID);
if(hr!=0)
    MessagePopup("err!","函数调用失败！");
```

EphRapidIO_GetDevID

函数功能

此函数获取模块 ID。

函数原型

ViStatus _VI_FUNC EphRapidIO_GetDevID(

ViUInt32 cardnum,

ViUInt32 *devID);

参数说明

cardnum

[in] 模块句柄，由连接函数获得。

devID

[out] 返回模块 ID，如 0x6295 表示 EP-X6295，PCIe 接口的 RAPIDIO 模块。

返回值

API_SUCCESS （函数调用成功）

API_FAILED （函数调用失败）

使用例程

```
ViStatus hr=0;
ViUInt32 devID=0;
ViUInt32 cardnum=0;

if (cardnum==0) return -1;
hr = EphRapidIO_GetDevID(cardnum,&devID);
if(hr != 0)
    MessagePopup(“err!”,”函数调用失败！”);
```

EphRapidIO_GetVersion

函数功能

获取模块版本号，API 版本号。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_GetVersion (
                                    ViUInt32  cardnum,
                                    ViUInt32 *Version,
                                    ViUInt32 *sVersion);
```

参数说明

cardnum

［in］ 模块句柄。

Version

［out］返回模块版本号，为 0x0100，表示 V1.00。

sVersion

［out］返回 API 版本号，为 0x0100，表示 V1.00。

返回值

0：表示成功 非 0：表示错误

使用例程

```
ViStatus hr=0;
ViUInt32 Version=0, sVersion=0;
ViUInt32 cardnum;

if (cardnum==0) return -1;
hr = EphRapidIO_GetVersion (cardnum,&Version, &sVersion);
if(hr != 0)
    MessagePopup("err!","函数调用失败！");
```

EphRapidIO_StatusToString

函数功能

将其他 API 函数的返回值转换为字符串。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_StatusToString (
                                    ViStatus hr,
                                    char *pString);
```

参数说明

hr

[in] 其它 API 函数返回值。

pString

[in] 返回值信息，建议 256 字节。

返回值

固定返回 0。

EphRapidIO_IntInstall

函数功能

注册/注销中断，注册时创建一个中断服务线程。

程序中注册/注销过程必须成对出现。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_IntInstall (
                                    ViUInt32  cardnum,
                                    API_INT_FIFO * sIntFIFO,
                                    ViUInt32 flag);
```

参数说明

cardnum

[in] 模块句柄。

sIntFIFO

[in] 中断服务结构体，详细见附录 1 的 API_RAPIDIO_INT_FIFO。

flag

[in] 0：注销中断，1：注册中断。注册/注销过程必须成对出现。

返回值

0：表示成功 非 0：表示错误。

EphRapidIO_SetNodeDeviceId

函数功能

设置本设备 ID，本设备作为接收端时，会检测接收数据中的 ID 标识是否与本设备匹配，不匹配时丢弃数据。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_SetNodeDeviceId (
                                    ViUInt32 cardnum,
                                    ViUInt32 ch,
                                    ViUInt32 NodeDeviceId);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

NodeDeviceId

[in] 设备 ID，根据硬件情况，当前 8 位有效。

返回值

0：表示成功 非 0：表示错误

使用例程

```
ViStatus hr=0;
ViUInt32 NodeDeviceId =32;
ViUInt32 cardnum;

if (cardnum==0) return -1;
hr = EphRapidIO_SetNodeDeviceId (cardnum,NodeDeviceId);
if(hr != 0)
    MessagePopup("err!","函数调用失败！");
```

EphRapidIO_GetNodeDeviceId

函数功能

获取本设备 ID，本设备作为接收端时，会检测接收数据中的 ID 标识是否与本设备匹配，不匹配时丢弃数据。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_GetNodeDeviceId (
                                    ViUInt32 cardnum,
                                    ViUInt32 ch,
                                    ViUInt32 *NodeDeviceId);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

NodeDeviceId

[out] 设备 ID，根据硬件情况，当前 8 位有效。

返回值

0：表示成功 非 0：表示错误

使用例程

```
ViStatus hr=0;
ViUInt32 NodeDeviceId =0;
ViUInt32 cardnum;

if (cardnum==0) return -1;
hr = EphRapidIO_GetNodeDeviceId (cardnum,&NodeDeviceId);
if(hr != 0)
    MessagePopup("err!","函数调用失败！");
```

EphRapidIO_SetWorkMode

函数功能

保留，实现 4X 及 1X 模式切换。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_SetWorkMode (
                                    ViUInt32 cardnum,
                                    ViUInt32 ch,
                                    ViUInt32 WorkMode);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

WorkMode

［in］ 工作模式，保留。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_GetWorkMode

函数功能

获取本设备的工作模式：4X 或 1X 模式。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_SetWorkMode (
                                    ViUInt32 cardnum,
                                    ViUInt32 ch,
                                    ViUInt32 *WorkMode);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。保留参数

WorkMode

［out］工作模式。1 = 1X 模式，4 = 4X 模式

返回值

0：表示成功 非 0：表示错误

EphRapidIO_MemRead

函数功能

读取通道缓存数据。

通道缓存：板卡为每个通道分配的数据存储空间。此缓存空间允许其他设备访问。如 EphRapidIO_NWrite()、EphRapidIO_NWriteR()、EphRapidIO_NWriteEx()或者 EphRapidIO_NRead()函数。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_MemRead (
                                ViUInt32  cardnum,
                                ViUInt32  ch,
                                ViUInt32  addr,
                                ViUInt32  len,
                                ViUInt32 *data);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

addr

[in] 设备通道缓存地址，现支持 64 字节对齐，1X 模式下 0～FFFFFC0 寻址范围，4X 模式下 0～1FFFFFC0 寻址范围。

len

[in] 读取的字节长度，单位“字节”，1X 模式下 4B～256MB 有效，4X 模式下 4B～512MB 有效。建议使用 4 的整数倍，否则实际读取数据长度可能小于设置长度。

data

[out] 读取的数据。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_MemWrite

函数功能

向通道缓存写入数据。

通道缓存：板卡为每个通道分配的数据存储空间。此缓存空间允许其他设备访问。如 EphRapidIO_NWrite()、EphRapidIO_NWriteR()、EphRapidIO_NWriteEx()或者 EphRapidIO_NRead()函数。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_MemWrite (
                                ViUInt32 cardnum,
                                ViUInt32 ch,
                                ViUInt64 addr,
                                ViUInt32 len,
                                ViUInt32 *data);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

addr

[in] 设备通道缓存地址，现支持 64 字节对齐，1X 模式下 0～0xFFFFFC0 寻址范围，4X 模式下 0～0x1FFFFFC0 寻址范围。

len

[in] 写入字节长度，单位“字节”，1X 模式下 64B～256MB 有效，4X 模式下 64B～512MB 有效。长度必须是 64 的整数倍。

data

[in] 写入的数据。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_WriteMaintPkt

函数功能

保留，实现 maintaince 读写功能，支持读本地维护模式数据。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_WriteMaintPkt (
                                    ViUInt32 cardnum,
                                    ViUInt32 ch,
                                    ViUInt32 remote,
                                    ViUInt32 addr,
                                    ViUInt32 data);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，保留。

addr

［in］ 写入地址，从 0 开始。

data

［in］ 写入的数据，32bit 有效。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_ReadMaintPkt

函数功能

保留，实现 maintaince 读写功能，支持写本地维护模式数据。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_ReadMaintPkt (
                                    ViUInt32  cardnum,
                                    ViUInt32  ch,
                                    ViUInt32  remote,
                                    ViUInt32  addr,
                                    ViUInt32 *data);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，保留。

addr

［in］ 读取地址，从 0 开始。

data

［out］读取的数据，32bit 有效。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_NWrite

函数功能

向远端发送块数据，支持 8B~64MB，数据发送完成后，不等待对方响应，直接返回，数据存放在远端设备的通道缓存中。

远端发送数据超时时间由 EphRapidIO_SetTimeOut()函数设置。超时时间结束时，设备可能不会立即停止数据发送或接收。即，可能存在：发送超时，但数据却完成发送的情况。

三种方式实现远端数据写入，对比如下：

EphRapidIO_NWrite	所有包均是无响应包	执行速度最快	函数执行完成时，不确定接收端是否完成接收
EphRapidIO_NWriteEx	最后一包为有响应包，其余为无响应包	执行速度较快	函数执行完成时，可确定接收端已完成接收
EphRapidIO_NWriteR	所有包均为有响应包	执行速度最慢	

函数原型

```
ViStatus _VI_FUNC EphRapidIO_NWrite (
                                    ViUInt32  cardnum,
                                    ViUInt32  ch,
                                    ViUInt32  remote,
                                    ViUInt32  srcTID,
                                    ViUInt64  addr,
                                    ViUInt32  len,
                                    ViUInt32 *data);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，0～65535 有效。

srcTID

［in］ 事务 ID，0～255 有效。

addr

［in］ 远端内存地址，要求为 8 的整数倍，其他要求需要根据对方设备地址空间分配而填写。如，EP-X6295 要求地址必须为 64 的整数倍。

len

［in］ 写入长度，单位字节，要求为 8 的整数倍，支持 8B～64MB。

data

［in］ 写入的数据。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_NWriteR

函数功能

向远端发送块数据，支持 8B~64MB。每包数据均等待对方响应后，再发送下一包数据，直到所有数据发送完成才返回。数据存放在远端设备的通道缓存中。

远端发送数据超时时间由 EphRapidIO_SetTimeOut()函数设置。超时时间结束时，设备可能不会立即停止数据发送或接收。即，可能存在：发送超时，但数据却完成发送的情况。

三种方式实现远端数据写入，对比如下：

EphRapidIO_NWrite	所有包均是无响应包	执行速度最快	函数执行完成时，不确定接收端是否完成接收
EphRapidIO_NWriteEx	最后一包为有响应包，其余为无响应包	执行速度较快	函数执行完成时，可确定接收端已完成接收
EphRapidIO_NWriteR	所有包均为有响应包	执行速度最慢	

函数原型

```
ViStatus _VI_FUNC EphRapidIO_NWriteR (
                                ViUInt32  cardnum,
                                ViUInt32  ch,
                                ViUInt32  remote,
                                ViUInt32  srcTID,
                                ViUInt64  addr,
                                ViUInt32  len,
                                ViUInt32 *data);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，0～65535 有效。

srcTID

［in］ 事务 ID，0～255 有效。

addr

［in］ 远端内存地址，至少为 8 的整数倍，其他要求需要根据对方设备地址空间分配而填写。如，EP-X6295 要求地址必须为 64 的整数倍。

len

［in］ 写入长度，单位字节，要求为 8 的整数倍，支持 8B～64MB。

data

［in］ 写入的数据。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_NWriteEx

函数功能

向远端发送块数据，支持 8B~64MB，仅等待对方响应最后一包时返回。数据存放在远端设备的通道缓存中。

远端发送数据超时时间由 EphRapidIO_SetTimeOut()函数设置。超时时间结束时，设备可能不会立即停止数据发送或接收。即，可能存在：发送超时，但数据却完成发送的情况。

三种方式实现远端数据写入，对比如下：

EphRapidIO_NWrite	所有包均是无响应包	执行速度最快	函数执行完成时，不确定接收端是否完成接收
EphRapidIO_NWriteEx	最后一包为有响应包，其余为无响应包	执行速度较快	函数执行完成时，可确定接收端已完成接收
EphRapidIO_NWriteR	所有包均为有响应包	执行速度最慢	

函数原型

```
ViStatus _VI_FUNC EphRapidIO_NWriteR (
                                ViUInt32  cardnum,
                                ViUInt32  ch,
                                ViUInt32  remote,
                                ViUInt32  srcTID,
                                ViUInt64  addr,
                                ViUInt32  len,
                                ViUInt32 *data);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，0～65535 有效。

srcTID

［in］ 事务 ID，0～255 有效。

addr

［in］ 远端内存地址，至少为 8 的整数倍，其他要求需要根据对方设备地址空间分配而填写。如，EP-X6295 要求地址必须为 64 的整数倍。

len

［in］ 写入长度，单位字节，要求为 8 的整数倍，支持 8B～64MB。

data

［in］ 写入的数据。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_NRead

函数功能

从远端读取一定长度的数据，支持 8、16、32、64、128、256 字节。数据存放在远端设备的通道缓存中。

远端读取超时时间由 EphRapidIO_SetTimeOut()函数设置。超时时间结束时，设备可能不会立即停止数据发送或接收。即，可能存在：发送超时，但数据读取操作却完成的情况。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_NRead (
                                    ViUInt32  cardnum,
                                    ViUInt32  ch,
                                    ViUInt32  remote,
                                    ViUInt32  srcTID,
                                    ViUInt64  addr,
                                    ViUInt32  len,
                                    ViUInt32 *data,
                                    ViUInt32 *actBytes);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，0～65535 有效。

srcTID

［in］ 事务 ID，0～255 有效。

addr

［in］ 远端内存地址，至少为 8 的整数倍，其他要求需要根据对方设备地

址空间分配而填写。如，EP-X6295 要求地址必须为 64 的整数倍。

len

［in］ 读取长度，单位字节，最大支持 256 字节。

data

［out］读取数据，最大缓存 256 字节。

actBytes

［out］实际读取字节数。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_WriteMsg

函数功能

向远端设备发送一条消息，最大负载 4096 字节数据。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_WriteMsg (
                                ViUInt32  cardnum,
                                ViUInt32  ch,
                                ViUInt32  remote,
                                ViUInt32  mailbox,
                                ViUInt32  ltr,
                                ViUInt32  len,
                                ViUInt32 *data);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，0～65535 有效。

mailbox

［in］ 邮箱号，0～63 有效。

ltr

［in］ ltr 号，0～3 有效。

len

［in］ 读取长度，1 表示 1 个字节，1～4096 有效。

data

［in］ 读取的消息数据，最大 4096 个字节（1024 个 32bit）

返回值

0：表示成功 非 0：表示错误

EphRapidIO_ReadMsg

函数功能

读取一条已经接收到的消息。当板卡接收到一条消息后会产生一次中断，中断源为消息中断（0x10）。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_ReadMsg (
                                ViUInt32  cardnum,
                                ViUInt32  ch,
                                ViUInt32 *mailbox,
                                ViUInt32 *ltr,
                                ViUInt32 *len,
                                ViUInt32 *data);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

mailbox

[out] 邮箱号，0～63 有效。

ltr

[out] ltr 号，0～3 有效。

len

[out] 指示接收消息的长度，1 表示 1 个字节，1～4096 有效。

data

[out] 读取的消息数据，最大 4096 个字节（1024 个 32bit），建议分配 4096 字节缓存。

返回值

0：表示成功 非 0：表示错误。

EphRapidIO_WriteDoorbell

函数功能

发起一次门铃事件。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_WriteDoorbell (
                                    ViUInt32 cardnum,
                                    ViUInt32 ch,
                                    ViUInt32 remote,
                                    ViUInt32 srcTID,
                                    ViUInt32 data);
```

参数说明

cardnum

［in］ 模块句柄。

ch

［in］ 通道号。

remote

［in］ 远端设备地址，0～65535 有效。

srcTID

［in］ 事务 ID，0～255 有效。

data

［in］ 门铃事件的数据，16bit 有效。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_ReadDoorbell

函数功能

读取门铃事件的数据。当板卡接收到门铃后会产生一次中断，中断源为消息中断（0x08）。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_ReadDoorbell (
                                    ViUInt32  cardnum,
                                    ViUInt32  ch,
                                    ViUInt32 *srcTID,
                                    ViUInt32 *data);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

srcTID

[out] 事务 ID，0～255 有效。

data

[out] 门铃事件的数据。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_SetTimeOut

函数功能

设置远端读写超时时间。设置超时时间会影响函数：EphRapidIO_NWrite()、EphRapidIO_NWriteR()、EphRapidIO_NWriteEx()、EphRapidIO_NRead()。

注意：

执行中断注册函数：EphRapidIO_IntInstall 时，会默认设置超时时间为中断超时时间。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_SetTimeOut (
                                ViUInt32  cardnum,
                                ViUInt32  timeout);
```

参数说明

cardnum

[in] 模块句柄。

timeout

[in] 超时时间,单位：毫秒。

返回值

0：表示成功 非 0：表示错误

EphRapidIO_SetMixMode

函数功能

设置通道接收收模式。

通道接收模式分为正常模式和混合模式。正常模式下，设备在接收数据时，仅接收数据的目标 ID 与设备的设备 ID 匹配的数据；混合模式下，设备则不判断 ID 是否匹配，会接收所有的数据。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_SetMixMode(
                                ViUInt32  cardnum,
                                ViUInt32 ch,
                                ViUInt32 flag);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

flag

[in] 收发模式，1:混合模式(接收所有数据) 0:正常模式(仅接收目的 ID 为自己的数据。

返回值

0: 表示成功 非 0: 表示错误

EphRapidIO_GetMixMode

函数功能

获取通道收发模式。

通道接收模式分为正常模式和混合模式。正常模式下，设备在接收数据时，仅接收数据的目标 ID 与设备的设备 ID 匹配的数据；混合模式下，设备则不判断 ID 是否匹配，会接收所有的数据。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_GetMixMode(
                                ViUInt32  cardnum,
                                ViUInt32 ch,
                                ViUInt32 *flag);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

flag

[out] 收发模式，1:混合模式(接收所有数据) 0:正常模式(仅接收目的 ID 为自己的数据。

返回值

0: 表示成功 非 0: 表示错误

EphRapidIO_GetDoorBellCnt

函数功能

获取通道门铃事件缓存个数。

注意：当使用中断时，中断结构体中也会返回该缓存个数，详细见附录 1 的 API_RAPIDIO_INT_FIFO。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_GetDoorBellCnt(
                                    ViUInt32 cardnum,
                                    ViUInt32 ch,
                                    ViUInt32 *count);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

count

[out] 缓存个数。范围 0~2048

返回值

0：表示成功 非 0：表示错误

EphRapidIO_GetRecvMsgFrameCnt

函数功能

获取通道消息缓存个数。

注意：当使用中断时，中断结构体中也会返回该缓存个数，详细见附录 1 的 API_RAPIDIO_INT_FIFO。

函数原型

```
ViStatus _VI_FUNC EphRapidIO_GetRecvMsgFrameCnt (
                                ViUInt32 cardnum,
                                ViUInt32 ch,
                                ViUInt32 *count);
```

参数说明

cardnum

[in] 模块句柄。

ch

[in] 通道号。

count

[out] 缓存个数。范围 0~2048

返回值

0：表示成功 非 0：表示错误

附录 1 数据结构

API_ RAPIDIO_INT_FIFO

```
typedef struct api_rapidio_int_fifo
{
    ViInt32 (_stdcall *function)(ViUInt32 cardnum, struct api_rapidio_int_fifo
*pFIFO);
    int iPriority;
    ViUInt32 cardnum;
    ViInt32    head_index;
    ViInt32    tail_index;
    ViInt32 IntMask;
    int      nUser[8];
    void *pUser[8];
    struct RAPIDIO_INTR_FIFO
    {
        ViUInt32 chnum;
        ViInt32 frameCount;
        ViInt32 InterruptSource;
    }
    fifo[16];
    ViUInt32 timeOut;
    ViUInt32 intDiscardCount;
}
API_RAPIDIO_INT_FIFO;
```

这是中断传入传出的结构体类型，在这个类型定义了中断所需要的参数。

序号	类型	参数	描述

1	*	function	需要声明为 ViInt32 (_stdcall *function)(ViUInt32 cardnum, struct fcp1276C_api_int_fifo *pFIFO)
2	int	iPriority	线程优先级。
3	ViUInt32	cardnum	产生中断的子板句柄。
4	ViInt32	head_index	中断 fifo 数据（参数 9）的起始位置。
5	ViInt32	tail_index	中断 fifo 数据（参数 9）的有效深度。
6	int	nUser[8]	用户自定义数据。
7	void*	pUser[8]	用户自定义指针。
	ViInt32	IntMask	中断屏蔽，保留
8	内嵌 struct FCP1276C_INTR_FIFO{} fifo [16]; 用于缓存每一个中断到来时，通道接收数据状态。		
9	ViUInt32	timeOut	中断超时时间
10	ViUInt32	intDiscardCount	未及时处理的中断数量

附录 2 宏定义

```
// 函数返回值信息
#define API_SUCCESS             0     //函数返回成功
#define API_FAILED              -1     //函数返回失败
#define API_INVALID_CARDNUM     -2     //无效板卡句柄号
#define API_INVALID_CH          -3     //无效通道号
#define API_INVALID_LEN         -4     //无效长度值
#define API_PTR_ERR             -5     //指针为空
#define API_NODEVICE            -6     //没有找到板卡
#define API_INVALID_PARA        -7     //无效参数
#define API_NOMATCHING          -8     //不匹配错误
#define API_SEND_TIMEOUT        -9  //发送超时
#define API_RECV_TIMEOUT        -10 //接收超时
#define API_DEVICE_CLOSED        -11    //设备已关闭
#define API_CHECK_FAILED        -12 //校验失败
#define API_NOT_ENOUGH_SPACE    -13     /*存储空间不够*/
#define API_NO_NEW_DATA         -14     /*If no new data is available
for this port*/
#define API_XMTBUF_BUSY         -15    /*发送 buffer 忙*/
#define API_BADMODE             -16    /*方式不匹配*/
#define API_BAD_VLHANDEL        -17    /*无效 VL 句柄*/
#define API_BAD_PORTHANDEL      -18
#define API_NO_VL_ENTRY         -19    //没有足够的空间创建 VL
#define API_NO_PORT_ENTRY       -20   //没有可用的端口
#define API_CONNECT_FAILED      -21   //连接失败
#define API_NO_SUPPORT          -22   //不支持功能
```

附录 3 程序范例

范例 1：

本例程实现了以下功能：

1. 连接模块，获得句柄;
2. 通道 0 发送门铃事件;
3. 通道 0 发送消息;
4. 通道 0 块传输；
5. 退出模块，释放资源。

```
void main()
{
ViStatus hr           = 0;
ViUInt32 busnum       = 3;
ViUInt32 deviceNum    = 4;
ViUInt32 DoorBellData = 0x5a5a; // 门铃数据
ViUInt32 ch           = 0;      // 通道 0
ViUInt32 remote       = 32;     // 远端地址为 32
ViUInt32 thisDevID    = 16;     // 本端地址为 16
ViUInt32 mailbox      = 2;      // 邮箱为 2
ViUInt32 ltr          = 1;      // ltr 为 1
ViUInt32 msgData[256] = {0};    // 消息数据
ViUInt32 msgLen       = 256*4;  // 消息数据长度
ViUInt32 srcTID       = 5;      // 事务 ID
ViUInt64 remoteDataAddr = 0;     // 远端数据地址
ViUInt32 cardnum;  // 板卡句柄
```

```
//连接板卡
hr = EphRapidIO_Open (1,busnum, deviceNum, &cardnum);
// 配置本端地址
hr = EphRapidIO_SetNodeDeviceId(cardnum, ch, thisDevID);
// 通道 0 发送门铃事件
hr = EphRapidIO_WriteDoorbell (cardnum, ch, remote, srcTID, DoorBellData);
// 通道 0 发送消息
hr = EphRapidIO_WriteMsg(cardnum, ch, remote, mailbox, ltr, msgLen, msgData);
// 通道 0 传输块数据，借用消息的数据
hr = EphRapidIO_NWriteR (cardnum, ch, remote, srcTID, remoteDataAddr, msgLen, msgData);

ch = 1;
// 通道 1 接收门铃数据
hr = EphRapidIO_ReadDoorbell (cardnum, ch, remote, &srcTID, &DoorBellData);
// 通道 1 接收消息
hr = EphRapidIO_ReadMsg (cardnum, ch, &mailbox, &ltr, &msgLen, &msgData);.
//退出模块
hr = EphRapidIO_Close (cardnum);
}
```

范例 2：

本例程实现了以下功能：

1. 连接模块，获得句柄;
2. 注册中断；
3. 中断处理门铃事件;

4. 中断接收消息;
5. 注销中断;
6. 退出模块，释放资源。

```
API_INT_FIFO sIntFIFO1 = {0};
// 中断回调函数
ViInt32 _stdcall demo_Msg_watch_function (ViUInt32 cardnum, ViUInt16 chnum, struct api_int_fifo *sIntFIFO);
// 中断回调函数，过程中不能快速响应读写面板控件，应减少面板操作
ViInt32 _stdcall demo_Msg_watch_function (ViUInt32 cardnum, ViUInt16 ch, struct api_int_fifo *sIntFIFO)
{
    // 声明参数，……
    ViStatus hr           = 0;
    ViUInt32 mailbox      = 2;      // 邮箱为 2
    ViUInt32 ltr          = 1;      // ltr 为 1
    ViUInt32 msgData[256] = {0};    // 消息数据
    ViUInt32 msgLen       = 256*4;  // 消息数据长度
    // 遍历不同事务，现处理 DoorBell 与 MSG
    if (sIntFIFO->mode == 0x8)  //  DoorBell 中断
    {
    // 接收门铃
    }
    else if (sIntFIFO->mode == 0x10)   // 接收到一帧消息 MSG
    {
    // 接收消息
    }
    else
    {    // 其它中断
    }
}
```

```
void main()
{
//连接板卡
hr = EphRapidIO_Open (TRUE,busnum, deviceNum, &cardnum);
// 注册中断
memset (&sIntFIFO1, 0, sizeof (API_INT_FIFO));
sIntFIFO1.function = demo_Msg_watch_function;
sIntFIFO1.iPriority = THREAD_PRIORITY_ABOVE_NORMAL;
sIntFIFO1.cardnum = cardnum;
hr = EphRapidIO_IntInstall (cardnum, &sIntFIFO1, 1);
While(1)
{     // 延时，等待中断处理
}
// 注销中断
hr = EphRapidIO_IntInstall (cardnum, &sIntFIFO1, 0);
//退出模块
hr = EphRapidIO_Close (cardnum);
}
```