429 系列板卡

用户手册

版　　本：　9.1.0

发布日期：2017-07-01

西安正弦波测控技术有限公司

声明

目 录

KMHT429 系列板卡 0
1. 概述 1
1.1. 功能 1
1.2. 基本工作环境 2
1.3. 板卡功能结构图 2
2. 硬件使用说明 3
2.1. 通讯接口定义 3
2.2. 硬件安装 4
3. 驱动接口说明 4
3.1. 驱动运行环境 4
3.2. 驱动安装说明 4
3.3. 驱动程序使用说明 4
3.3.1. 驱动程序引用文件 4
3.3.2. 驱动程序函数接口说明 4
3.3.2.1. ARINC429_Open 4
3.3.2.2. ARINC429_Close 5
3.3.2.3. ARINC429_Reset 5
3.3.2.4. ARINC429_StartTimeTag 5
3.3.2.5. ARINC429_SetTimeTag 6
3.3.2.6. ARINC429_GetTimeTag 6
3.3.2.7. ARINC429_RX_Reset 6
3.3.2.8. ARINC429_RX_Configure 7
3.3.2.9. ARINC429_RX_TriggerDepth 8
3.3.2.10. ARINC429_RX_TimeTagMode 8
3.3.2.11. ARINC429_RX_RecvEnable 8
3.3.2.12. ARINC429_RX_LableFilterEnable 9
3.3.2.13. ARINC429_RX_LableFilterTable 9
3.3.2.14. ARINC429_RX_BufferClr 9
3.3.2.15. ARINC429_RX_BufferStatus 9
3.3.2.16. ARINC429_RX_BufferData 10
3.3.2.17. ARINC429_TX_Reset 10
3.3.2.18. ARINC429_TX_Configure 11
3.3.2.19. ARINC429_TX_TriggerDepth 11
3.3.2.20. ARINC429_TX_TimerIntervalWord 12
3.3.2.21. ARINC429_TX_TimerIntervalFrame 12
3.3.2.22. ARINC429_TX_BufferClr 12
3.3.2.23. ARINC429_TX_BufferStatus 13

3.3.2.24. ARINC429_TX_BufferData 13
3.3.2.25. ARINC429_TX_BufferAreaSet 13
3.3.2.26. ARINC429_TX_BufferAreaGet 14
3.3.2.27. ARINC429_TX_BufferUpdate 14
3.3.2.28. ARINC429_TX_Start 14
3.3.2.29. ARINC429_TX_Sop 14
3.3.2.30. ARINC429_INT_Thread 15
3.3.2.31. ARINC429_INT_Init 15
3.3.2.32. ARINC429_INT_EnableGlobal 15
3.3.2.33. ARINC429_INT_EnableBit 15
3.3.2.34. ARINC429_INT_StatusClear 16
3.3.2.35. ARINC429_INT_Status 17
3.4. 驱动函数调用步骤 18
3.4.1. 驱动函数调用步骤 18
4. Demo 应用程序说明 21
4.1. Demo 运行环境 21
4.1.1. 硬件环境 21
4.1.2. 软件环境 21
4.1.3. 开发工具 21
4.2. Demo 简介 21
4.2.1. 板卡号选择窗口 21
4.2.2. 应用程序主窗口 22
4.2.3. 板卡设置窗口 23
4.3. 菜单功能 24
4.4. 操作建议 24
4.4.1. 开启应用程序 24
4.4.2. 板卡号选择 24
4.4.3. 设置板卡 25
4.4.4. 数据通讯操作 25
4.4.5. 退出应用程序 25
4.5. 使用说明 26
4.5.1. 硬件设置 26
4.5.1.1. 配置所有通道 26
4.5.1.2. 配置参数设置 26
4.5.1.3. 接收单元参数设置 27
4.5.1.4. 发送单元参数设置 28
4.5.1.5. 设置接收标号过滤 29
4.5.1.6. 配置的保存与加载（暂不支持） 30
4.5.2. 数据的接收与发送 31
4.5.2.1. 接收数据 31

4.5.2.2. 发送数据 34
4.6. 编程举例 35
附录 A：429 数据格式转换 1

手册内容简介

本手册内容共分为四个章节，分别为“概述”、“硬件使用说明”、“驱动接口说明”和“应用程序说明”。

第一章“概述”，是对硬件功能与使用方法的整体概述，更多的细节在第二章—硬件使用说明、第三章“驱动接口说明”和第四章“应用程序说明”中进行详细描述。

第二章“硬件使用说明”，描述了硬件的使用环境、配置安装方法、硬件结构及用户在实际使用中可能用到的硬件接口。

第三章“驱动接口说明”，该部分内容是本手册的重点内容，它详细描述了接口函数的功能及使用方法，通过这一章节的了解，用户可以进行产品的应用程序开发。

第四章“应用程序说明”，介绍了提供给用户的应用程序的使用方法，通过这个应用程序，用户可以不需编程而直接进行一些基本的通讯操作，达到快速应用的目的。另外，在该章中，也提供了驱动接口使用例程，为用户开发适合个人要求的应用程序的提供参考。

1. 概述

1.1. 功能

■ 基本特性：

- ■ 429 系列板卡, 完全满足美国航空电子工程委员会的 ARINC429 总线协议
- ■ 根据需求：

  接收可选择 1~16 路（计算机接口为 PCI、CPCI、PXI、PC104、PC104-Plus）

  接收可选择 1~8 路（计算机接口为 USB）
- ■ 根据需求:

  发送可选择 1~16 路（计算机接口为 PCI、CPCI、PXI、PC104、PC104-Plus）

  发送可选择 1~ 8 路（计算机接口为 USB）
- ■ 每路的波特率、奇偶校验等参数可单独设置
- ■ 每一路中的接收单元、发送单元的波特率、奇偶校验等参数可单独设置
- ■ 波特率除了包括 100K、50K、12.5K 标准速率，还可以设置 1K~100K 之间的任何值
- ■ 校验方式包括：无校验、奇校验、偶校验
- ■ 接收可选择：普通模式、时标模式
- ■ 接收可选择：正常接收、自环接收
- ■ 接收带有标号过滤功能
- ■ 接收每路带硬件缓存 65535x32-Bit FIFO
- ■ 接收可选择：编码接收、总线接收
- ■ 发送可设置：字间隔、帧间隔
- ■ 发送每路带硬件缓存 256x32-Bit FIFO，一帧最多发送 256x32-Bit

说明：我公司 ARINC429 系列板卡中，接收和发送数据格式有两种：

① 编码模式：

这种模式下，“总线上的数据格式”和“计算机系统软件的数据格式”并不一致。因为在硬件中，发送按既定的编码方式发送，接收按相同方式解码，具体格式转换方式见本文结尾附录 A。

② 总线模式。

这种模式下，“总线上的数据格式”和“计算机系统软件的数据格式”一致。因为在硬件中，接收和发送，严格按总线位序收发，不经过编码转换。

1.2. 基本工作环境

操作系统：WinXP/Win7
工作温度：-40～+85℃
相对湿度：0～95%

1.3. 板卡功能结构图

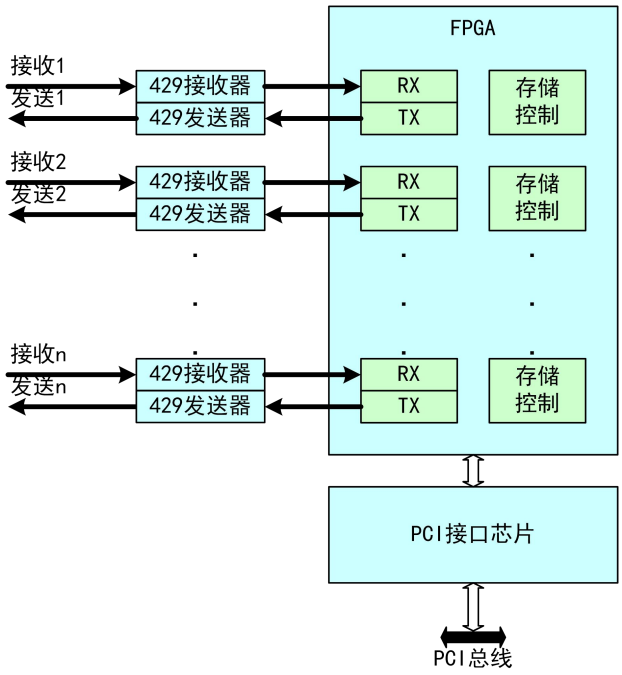


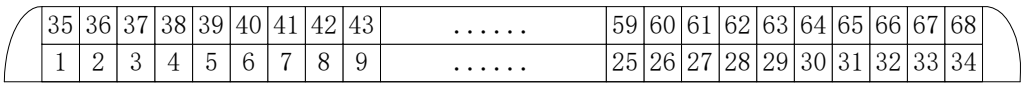
图 1-1 板卡功能结构图

2. 硬件使用说明

2.1. 通讯接口定义

表 2-1 标准 SCSI 68 连接器接头信号表

管脚	名称	说明	管脚	名称	说明
1	CH00_RX+	1 路接收+	35	CH00_TX+	1 路发送+
2	CH00_RX-	1 路接收-	36	CH00_TX-	1 路发送-
3	CH01_RX+	2 路接收+	37	CH01_TX+	2 路发送+
4	CH01_RX-	2 路接收-	38	CH01_TX-	2 路发送-
5	CH02_RX+	3 路接收+	39	CH02_TX+	3 路发送+
6	CH02_RX-	3 路接收-	40	CH02_TX-	3 路发送-
7	CH03_RX+	4 路接收+	41	CH03_TX+	4 路发送+
8	CH03_RX-	4 路接收-	42	CH03_TX-	4 路发送-
9	CH04_RX+	5 路接收+	43	CH04_TX+	5 路发送+
10	CH04_RX-	5 路接收-	44	CH04_TX-	5 路发送-
11	CH05_RX+	6 路接收+	45	CH05_TX+	6 路发送+
12	CH05_RX-	6 路接收-	46	CH05_TX-	6 路发送-
13	CH06_RX+	7 路接收+	47	CH06_TX+	7 路发送+
14	CH06_RX-	7 路接收-	48	CH06_TX-	7 路发送-
15	CH07_RX+	8 路接收+	49	CH07_TX+	8 路发送+
16	CH07_RX-	8 路接收-	50	CH07_TX-	8 路发送-
17	CH08_RX+	9 路接收+	51	CH08_TX+	9 路发送+
18	CH08_RX-	9 路接收-	52	CH08_TX-	9 路发送-
19	CH09_RX+	10 路接收+	53	CH09_TX+	10 路发送+
20	CH09_RX-	10 路接收-	54	CH09_TX-	10 路发送-
21	CH10_RX+	11 路接收+	55	CH10_TX+	11 路发送+
22	CH10_RX-	11 路接收-	56	CH10_TX-	11 路发送-
23	CH11_RX+	12 路接收+	57	CH11_TX+	12 路发送+
24	CH11_RX-	12 路接收-	58	CH11_TX-	12 路发送-
25	CH12_RX+	13 路接收+	59	CH12_TX+	13 路发送+
26	CH12_RX-	13 路接收-	60	CH12_TX-	13 路发送-
27	CH13_RX+	14 路接收+	61	CH13_TX+	14 路发送+
28	CH13_RX-	14 路接收-	62	CH13_TX-	14 路发送-
29	CH14_RX+	15 路接收+	63	CH14_TX+	15 路发送+
30	CH14_RX-	15 路接收-	64	CH14_TX-	15 路发送-
31	CH15_RX+	16 路接收+	65	CH15_TX+	16 路发送+
32	CH15_RX-	16 路接收-	66	CH15_TX-	16 路发送-
33			67		
34			68		



板卡上SCSI 68接头点号示意图

2.2. 硬件安装

a） 板卡安装：
   ① PCI 接口卡。将板卡插入主机箱的 PCI 插槽，然后用螺丝将板卡挡板和机箱固定在一起；
   ② CPCI 或 PXI 接口卡。将板卡沿导轨槽插入主机箱的插槽，直至板卡完全插入，并用螺丝固定板卡和机箱；
   ③ USB 接口卡。使用 USB（支持 USB2.0 以上）线缆将板卡连接到主机上。
b） 用板卡配套线缆将板卡接入 429 网络，确认接触良好后，到此硬件安装结束；
c） 开机后，提示“发现新硬件”，驱动程序安装见第三章驱动接口说明。

3. 驱动接口说明

3.1. 驱动运行环境

Windows 2000/ XP/7 操作系统

3.2. 驱动安装说明

a） 将板卡按规定步骤插入机箱
b） 开机后系统提示找到新硬件，需要安装它的驱动程序
c） 点击硬件安装的下一步，若 PC 机是 Windows 2000/98 的操作系统请指向光盘中“驱动安装程序\Win2k”目录；若 PC 机是 Windows XP 操作系统，请指向光盘中“驱动安装程序\WinXP”目录
d） 点击下一步，直到驱动程序安装成功
e） 安装完毕，在设备管理器会看到” KMHT429-PCI/CPCI”设备

3.3. 驱动程序使用说明

3.3.1. 驱动程序引用文件

库文件：ARINC429.dll 和 ARINC429.lib
函数库头文件：ARINC429_dll.h

3.3.2. 驱动程序函数接口说明

3.3.2.1. ARINC429_Open

函数原型：int ARINC429_Open (DevHandle *phARINC429, DWORD CardId);

函数功能：打开板卡，并分配板卡资源。
参数说明：phARINC429：板卡句柄的指针。
CardId：板卡编号，取值 0~255。若 PC 机中同时插 256 块同型号板卡，板卡的编号按板卡所在的插槽离 CPU 的距离由近到远以次编号 0、1…255。若 PC 机中同时只插一块板，板卡编号为 0。
返回值： 操作成功，返回值为 0；操作失败，返回值为-1；参数错误，返回值为-2。

3.3.2.2. ARINC429_Close

函数原型：int ARINC429_Close (DevDevHandle hARINC429);
函数功能：关闭板卡，释放板卡资源。
参数说明：hARINC429：板卡的句柄。
返回值： 操作成功，返回值为 0；操作失败，返回值为-1；参数错误，返回值为-2。

3.3.2.3. ARINC429_Reset

函数原型：int ARINC429_Reset (DevDevHandle hARINC429);
函数功能：板卡复位函数，全局复位整个板卡。
参数说明：hARINC429：板卡的句柄。
返回值： 操作成功，返回值为 0；操作失败，返回值为-1。

说明：板卡复位 1~16 通道，执行的操作为：
接收：波特率 100K，无校验，编码方式接收，标号过滤不使能，自环测试不使能
发送：波特率 100K，无校验，编码方式发送

3.3.2.4. ARINC429_StartTimeTag

函数原型： int **ARINC429_StartTimeTag** (DevDevHandle hARINC429, DWORD Enable, LPSYSTEMTIME CurSysTime);
函数功能：启动或停止时标模式。
参数说明：hARINC429：板卡的句柄。
Enable：时标使能位。1: 使能；0: 不使能。
CurSysTime：系统当前时间。

```
typedef struct _SYSTEMTIME
{
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
```

```
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds;
} SYSTEMTIME, *LPSYSTEMTIME;
```

wYear：年；
WMonth：月; January = 1, February =2，以此类推；
wDayOfWeek：星期; Sunday = 0, Monday = 1，以此类推；
wDay：日；
wHour：小时；
wMinute：分；
wSecond：秒；
wMilliseconds：毫秒。

返回值： 返回值为 0。

3.3.2.5. ARINC429_SetTimeTag

函数原型：int ARINC429_SetTimeTag (DevDevHandle hARINC429, DWORD TimeTag);
函数功能：设置 32 位时标起始值，分辨率 50us。只在时标模式使能的情况下，设置才有效。
参数说明：HARINC429：板卡的句柄。
TimeTag：时标起始值。
返回值： 返回值为 0。

3.3.2.6. ARINC429_GetTimeTag

函数原型：int ARINC429_GetTimeTag (DevDevHandle hARINC429, DWORD *TimeTag);
函数功能：获取当前时标值，分辨率 50us。时标模式不使能时，获取的时标始终为常值 0。
参数说明：hARINC429：板卡的句柄。
TimeTag：当前时标值。
返回值： 返回值为 0。

3.3.2.7. ARINC429_RX_Reset

函数原型：int ARINC429_RX_Reset (DevHandle hARINC429, UCHAR ch);
函数功能：复位指定通道接收单元函数。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
返回值： 返回值为 0。

说明：板卡复位指定通道接收单元，执行的操作为：

接收：波特率 100K，无校验，编码方式接收，标号过滤不使能，自环测试不使能

3.3.2.8. ARINC429_RX_Configure

函数原型： int ARINC429_RX_Configure (DevHandle hARINC429, UCHAR ch, RX_CFG_STRUCT *Cfg);

函数功能：接收单元参数配置。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

Cfg：接收单元配置参数结构

```
typedef struct
{
    DWORD Cfg_BaudCounter;
    DWORD Cfg_ParityEnable;
    DWORD Cfg_ParityMode;
    DWORD Cfg_BusOrderEnable;
    DWORD Cfg_SDDecoderEnable;
    DWORD Cfg_SDDecoderData;
    DWORD Cfg_SelfTestEnable;
}RX_CFG_STRUCT;
```

Cfg_BaudCounter：接收波特率设置：

计算公式如下：如果需要设置的波特率为 X，则这里设置的值为 CNT = 48000000 / (32 * X)。例如，设置的波特率为 100K，则 CNT = 48000000 / (32 * 100000) = 15 = 0x00001111。

Cfg_ParityEnable：接收奇偶校验使能。

1：校验使能，校验位为校验值；

0：校验不使能，校验位可当数据位使用。

Cfg_ParityMode：奇偶校验模式选择。

1：奇校验；

0：偶校验。

Cfg_BusOrderEnable：接收通道选择“总线顺位接收”/“编码顺位接收”。

1：按总线上位顺序接收，接收不需译码；

0：按既定解码方式接收。

Cfg_SDDecoderEnable：接收通道 S/D 码检测使能。

1：使能 CHx 接收通道，源/目标译码器；

0：关闭 CHx 接收通道，源/目标译码器。

Cfg_SDDecoderData：接收通道 S/D 码。

如果 RX_SD_EN =1，第 1 路接收通道将收到 ARINC429 数据的 S/D 与 B10B9 进行比较，相同则接收，不相同则丢弃。其中， B10 与串行数据

bit10 比较， B9 与串行数据 bit9 比较。

Cfg_SelfTestEnable：接收通道自环测试使能。

1：自环使能；

0：自环不使能。

返回值： 返回值为 0。

3.3.2.9. ARINC429_RX_TriggerDepth

函数原型： int ARINC429_RX_TriggerDepth (DevHandle hARINC429, UCHAR ch, DWORD TriggerDepth);

函数功能：设置接收 FIFO 触发深度，在接收采用中断方式下有效，取值 0~65534。

如触发深度为 n，表示硬件接收到 n+1 个数据后硬件会产生中断。硬件初始值为 0，表示触发深度为 1，即硬件接收到 1 个数据后硬件会产生中断。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

TriggerDepth：FIFO 触发深度，取值 0~65534。

返回值： 返回值为 0。

3.3.2.10. ARINC429_RX_TimeTagMode

函数原型： int ARINC429_RX_TimeTagMode (DevHandle hARINC429, UCHAR ch, DWORD Enable);

函数功能：设置接收时标模式。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

Enable：时标使能。1：使能该功能；0：不使能。注意：必须在执行函数 ARINC429_StartTimeTag 使能总时标之后，此通道时标设置为 1 才有意义。

返回值： 返回值为 0。

3.3.2.11. ARINC429_RX_RecvEnable

函数原型： int ARINC429_RX_RecvEnable (DevHandle hARINC429, UCHAR ch, DWORD Enable);

函数功能：接收单元接收使能。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

Enable：时标使能位。1：接收使能；0：接收不使能。

返回值： 返回值为 0。

3.3.2.12. ARINC429_RX_LableFilterEnable

函数原型：int ARINC429_RX_LableFilterEnable (DevHandle hARINC429, UCHAR ch, DWORD Enable);
函数功能：标号过滤表使能。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
Enable：标号过滤表使能位。1：标号过滤使能；0：不使能。
返回值： 返回值为 0。

3.3.2.13. ARINC429_RX_LableFilterTable

函数原型：int ARINC429_RX_LableFilterTable (DevHandle hARINC429, UCHAR ch, LABEL_TABLE_STRUCT FilterTable);
函数功能：接收标号过滤表配置。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
FilterTable：接收标号过滤表结构。

```
typedef struct
{
    DWORD LabelTable[4][256];
}LABEL_TABLE_STRUCT;
```

LabelTable[4][256]：接收波特率设置：
数组的行坐标值代表 S/D 号，列坐标值代表标号；若数组对应元素的值不为零，则 S/D 号对应的标号参与过滤。例：LabelTable[1][2]=1：表示 S/D 为 1 且 Label 为 2 的 429 数据参与过滤，即被接收。
返回值： 返回值为 0。

3.3.2.14. ARINC429_RX_BufferClr

函数原型：int ARINC429_RX_BufferClr (DevHandle hARINC429, UCHAR ch);
函数功能：清空接收缓存。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
返回值： 返回值为 0。

3.3.2.15. ARINC429_RX_BufferStatus

函数原型：int ARINC429_RX_BufferStatus (DevHandle hARINC429, UCHAR ch, DWORD *Status);

函数功能：获取接收 FIFO 状态。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
Status：接收 FIFO 状态：
状态字为 32bit，其中各位的含义如下表：

名称	位	说明
-	31:18	Reserve
RX_FIFO_FULL	17	接收 FIFO 满标志
RX_FIFO_EMPTY	16	接收 FIFO 空标志
RX_FIFO_CNT[15：0]	15::0	接收 FIFO 中的数据 CNT

返回值： 返回值为 0。

3.3.2.16. ARINC429_RX_BufferData

函数原型： Int ARINC429_RX_BufferData (DevHandle hARINC429, UCHAR ch, DWORD TimeTagMode, DWORD *Length, DWORD *DataBuf);
函数功能：接收数据。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
TimeTagMode：时标使能位。1：使能该功能；0：不使能。
Length：接收的有效数据数量。
DataBuf：存放接收的数据。
返回值： 读出数据，返回值为 1；未读出消息，返回值为 2。

注意： 在正常接收模式下，Buf 中存储的全是接收的 32Bit 的 429 数据；而在“时标模式”下，Buf 中，429 数据（32Bit）和时标（32Bit）交替存放的，表示一个 429 数据跟随一个相应的时间值。

3.3.2.17. ARINC429_TX_Reset

函数原型：int ARINC429_TX_Reset (DevHandle hARINC429, UCHAR ch);
函数功能：复位指定通道发送单元函数。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
返回值： 返回值为 0。

说明：板卡复位指定通道发送单元，执行的操作为：
发送：波特率 100K，无校验，编码方式发送。

3.3.2.18. ARINC429_TX_Configure

函数原型： int ARINC429_TX_Configure (DevHandle hARINC429, UCHAR ch, TX_CFG_STRUCT *Cfg);

函数功能：接收单元参数配置。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

Cfg：接收单元配置参数结构

```
typedef struct
{
    DWORD Cfg_BaudCounter;
    DWORD Cfg_ParityEnable;
    DWORD Cfg_ParityMode;
    DWORD Cfg_BusOrderEnable;
}RX_CFG_STRUCT;
```

Cfg_BaudCounter：发送波特率设置：

计算公式如下：如果需要设置的波特率为 X，则这里设置的值为 CNT = 48000000 / (32 * X)。例如，设置的波特率为 100K，则 CNT = 48000000 / (32 * 100000) = 15 = 0x00001111。

Cfg_ParityEnable：发送奇偶校验使能。

1：校验使能，校验位为校验值；

0：校验不使能，校验位可当数据位使用。

Cfg_ParityMode：奇偶校验模式选择。

1：奇校验；

0：偶校验。

Cfg_BusOrderEnable：发送通道选择“总线上按数据原始顺位发送”、“总线上按数据编码顺位发送”。

1：总线上按数据原始顺位发送。

0：总线上按数据编码顺位发送。

返回值： 返回值为 0。

3.3.2.19. ARINC429_TX_TriggerDepth

函数原型： int ARINC429_TX_TriggerDepth (DevHandle hARINC429, UCHAR ch, DWORD TriggerDepth);

函数功能：设置发送 FIFO 触发深度，在发送采用中断方式下有效，取值 0~255。（由于发送暂不支持采用中断方式，此值暂无意义）

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

TriggerDepth：FIFO 触发深度，取值 0~255。

返回值： 返回值为 0。

3.3.2.20. ARINC429_TX_TimerIntervalWord

函数原型：Int ARINC429_TX_TimerIntervalWord (DevHandle hARINC429, UCHAR ch, DWORD Time);

函数功能：发送字定时发送时间设置。这里的字定时，是指前 DWORD 字发送的第一个 Bit 位，到下一个 DWORD 字发送的第一个 Bit 位之间的时间，分辨率 50us。

注意，字定时设置的值必须合理，如 100K 速率下，一个字发送需要 320us。如设定的值小于 320us，则设置无效。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

Time：设定的字定时时间值，取值 0~65535，分辨率 50us。

返回值： 返回值为 0。

3.3.2.21. ARINC429_TX_TimerIntervalFrame

函数原型：Int ARINC429_TX_TimerIntervalFrame (DevHandle hARINC429, UCHAR ch, DWORD Time);

函数功能：帧定时发送时间设置。这里的帧定时，是指前帧发送的第一个 DWORD 字，到下一个帧发送的第一个 DWORD 字之间的时间，分辨率 50us。

注意，设置的帧定时要考虑如下因素：

① 在字定时为 0 的情况下（字肩并肩发送），100K 速率下发送 10 个字，整帧发送需持续 3200us。那么设定的帧定时应不小于 3200us，否则，数据发送不完全。

② 在字定时为 500us 的情况下，100K 速率下发送 10 个字，整帧发送需持续 5000us，那么设定的帧定时应不小于 5000us，否则，数据发送不完全。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

Time：设定的帧定时时间值，取值 0~65535，分辨率 50us。

返回值： 返回值为 0。

3.3.2.22. ARINC429_TX_BufferClr

函数原型： int ARINC429_TX_BufferClr (DevHandle hARINC429, UCHAR ch, DWORD AreaBNA);

函数功能：清空发送缓存。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

AreaBNA：发送区选择。0：A 发送区；1：B 发送区。

返回值： 返回值为 0。

3.3.2.23. ARINC429_TX_BufferStatus

函数原型： int ARINC429_TX_BufferStatus (DevHandle hARINC429, UCHAR ch, DWORD AreaBNA, DWORD *Status);

函数功能：获取发送 FIFO 状态。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

AreaBNA：发送区选择。0：A 发送区；1：B 发送区。

Status：发送 FIFO 状态：

状态字为 32bit，其中各位的含义如下表：

名称	位	说明
-	31:18	Reserve
TX_FIFO_FULL	17	发送 FIFO 满标志
TX_FIFO_EMPTY	16	发送 FIFO 空标志
TX_FIFO_CNT[15：0]	15::0	发送 FIFO 中的数据 CNT

返回值： 返回值为 0。

3.3.2.24. ARINC429_TX_BufferData

函数原型： Int ARINC429_TX_BufferData (DevHandle hARINC429, UCHAR ch, DWORD AreaBNA, DWORD Length, DWORD *DataBuf);

函数功能：发送数据。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

AreaBNA：发送区选择。0：A 发送区；1：B 发送区。

Length：发送数据量，取值 1~256。

DataBuf：存放发送数据。

返回值： 发送正常，返回值为 1；发送异常，返回值为 2。

3.3.2.25. ARINC429_TX_BufferAreaSet

函数原型： int ARINC429_TX_BufferAreaSet (DevHandle hARINC429, UCHAR ch, DWORD AreaBNA);

函数功能：发送区设置（可设置为 A 区或 B 区）。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

AreaBNA：发送区选择。0：A 发送区；1：B 发送区。

返回值： 返回值为 0。

3.3.2.26. ARINC429_TX_BufferAreaGet

函数原型： Int ARINC429_TX_BufferAreaGet (DevHandle hARINC429, UCHAR ch, DWORD *AreaBNA);

函数功能：获取当前的发送区（A 区或 B 区）

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

AreaBNA：发送区标志。0：A 发送区；1：B 发送区。

返回值： 返回值为 0。

3.3.2.27. ARINC429_TX_BufferUpdate

函数原型： Int ARINC429_TX_BufferUpdate (DevHandle hARINC429, UCHAR ch, DWORD Length, DWORD *DataBuf);

函数功能：在帧重发的情况下，更新发送数据帧。

如果在帧重发方式下，且发送区工作在 A 区域，调用此函数后，将数据写入 B 区域，然后设置发送区工作在 B 区域。同理，如果在帧重发方式下，且发送区工作在 B 区域，调用此函数后，将数据写入 A 区域，然后设置发送区工作在 A 区域。

这种方式下更新发送数据，保证了数据帧周期发送不中断，动态更新。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

Length：发送数据量，取值 1~256。

DataBuf：存放更新的发送数据。

返回值： 更新成功，返回值为 1；更新失败，返回值为 2。

3.3.2.28. ARINC429_TX_Start

函数原型：int ARINC429_TX_Start (DevHandle hARINC429, UCHAR ch);

函数功能：发送开始。

参数说明：hARINC429：板卡的句柄。

ch：通道号，取值 0~15。

返回值： 返回值为 0。

3.3.2.29. ARINC429_TX_Sop

函数原型：int ARINC429_TX_Stop (DevHandle hARINC429, UCHAR ch);

函数功能：发送停止。
参数说明：hARINC429：板卡的句柄。
ch：通道号，取值 0~15。
返回值：　返回值为 0。

3.3.2.30. ARINC429_INT_Thread

函数原型：Int **INT_Thread** (DevHandle hARINC429);
函数功能：创建中断线程。
参数说明：hARINC429：板卡的句柄。
返回值：　操作成功，返回值为 0；操作失败，返回值为-1。

3.3.2.31. ARINC429_INT_Init

函数原型：Int **INT_Init** (DevHandle hARINC429, INT_HANDLER INT_Func);
函数功能：中断初始化，使能总中断。
参数说明：hARINC429：板卡的句柄。
INT_Func：中断处理函数。
返回值：　操作成功，返回值为 0；操作失败，返回值为-1。

3.3.2.32. ARINC429_INT_EnableGlobal

函数原型：int **INT_EnableGlobal** (DevHandle hARINC429, DWORD Enable);
函数功能：使能总中断。
参数说明：hARINC429：板卡的句柄。
Enable：使能位 1:开启总中断；0:关闭总中断。
返回值：　返回值为 0。

3.3.2.33. ARINC429_INT_EnableBit

函数原型：int **INT_EnableBit** (DevHandle hARINC429, DWORD EnableBitVector);
函数功能：使能分中断。
参数说明：hARINC429：板卡的句柄。
EnableBitVector：每位定义：1:中断使能；0:不使能。此 32 位字的每一位定义如下表 3-1《中断位使能表》

表 3-1　中断位使能表

BIT	说明
31:16	未使用
15	16 通道接收中断使能
14	15 通道接收中断使能

BIT	说明
13	14 通道接收中断使能
12	13 通道接收中断使能
11	12 通道接收中断使能
10	11 通道接收中断使能
9	10 通道接收中断使能
8	9 通道接收中断使能
7	8 通道接收中断使能
6	7 通道接收中断使能
5	6 通道接收中断使能
4	5 通道接收中断使能
3	4 通道接收中断使能
2	3 通道接收中断使能
1	2 通道接收中断使能
0	1 通道接收中断使能

返回值： 返回值为 0。

3.3.2.34. ARINC429_INT_StatusClear

函数原型： int **INT_StatusClear** (DevHandle hARINC429, DWORD StatusClearVector);
函数功能： 清中断，用来清除相应中断状态位。
参数说明： hARINC429：板卡的句柄。
StatusClearVector：每位定义：1:清此中断状态位；0:不清。此 32 位字的每一位定义如下表 3-2《清中断状态位》

表 3-2 清中断状态位

BIT	说明
31:16	未使用
15	清 16 通道接收中断状态位
14	清 15 通道接收中断状态位
13	清 14 通道接收中断状态位
12	清 13 通道接收中断状态位
11	清 12 通道接收中断状态位
10	清 11 通道接收中断状态位
9	清 10 通道接收中断状态位
8	清 9 通道接收中断状态位
7	清 8 通道接收中断状态位
6	清 7 通道接收中断状态位
5	清 6 通道接收中断状态位

BIT	说明
4	清 5 通道接收中断状态位
3	清 4 通道接收中断状态位
2	清 3 通道接收中断状态位
1	清 2 通道接收中断状态位
0	清 1 通道接收中断状态位

返回值： 返回值为 0。

3.3.2.35. ARINC429_INT_Status

函数原型： int **INT_Status**(DevHandle hARINC429, DWORD *Status);

函数功能： 读取中断状态，以确定如何响应中断。

参数说明： hARINC429：板卡的句柄。

Status：双字指针，指向中断状态寄存器空间。每位定义：1:中断状态有效；0:无效。此32位字的每一位定义如下表3-3《中断状态位》

表 3-3 中断状态位

BIT	说明
31:16	未使用
15	16 通道接收中断状态位
14	15 通道接收中断状态位
13	14 通道接收中断状态位
12	13 通道接收中断状态位
11	12 通道接收中断状态位
10	11 通道接收中断状态位
9	10 通道接收中断状态位
8	9 通道接收中断状态位
7	8 通道接收中断状态位
6	7 通道接收中断状态位
5	6 通道接收中断状态位
4	5 通道接收中断状态位
3	4 通道接收中断状态位
2	3 通道接收中断状态位
1	2 通道接收中断状态位
0	1 通道接收中断状态位

返回值： 返回值为 0。

3.4. 驱动函数调用步骤

3.4.1. 驱动函数调用步骤

1. 打开板卡

   打开板卡（ARINC429_Open）

2. 复位板卡（ARINC429_Reset）

   全局复位板卡（ARINC429_Reset）

3. 中断初始化（中断方式）

   ① 创建中断线程（ARINC429_INT_Thread）

   ② 初始化中断（ARINC429_INT_Init）

   ③ 使能总中断（ARINC429_INT_EnableGlobal）

4. 时标模式

   启动或停止时间标签模式（ARINC429_StartTimeTag）

5. 接收（指定通道）

   （1）接收通道复位（ARINC429_RX_Reset）

   （2）接收通道参数设置

   ① 配置波特率、奇偶校验、接收编码方式等参数（ARINC429_RX_Configure）

   ② 设置接收触发深度(ARINC429_RX_TriggerDepth) （中断方式）

   ③ 设置接收数据是否带时标（ARINC429_RX_TimeTagMode）

   ④ 设置接收标号过滤

   a. 设置标号过滤使能（ARINC429_RX_LableFilterEnable）

   b. 设置标号过滤表（ARINC429_RX_LableFilterTable）

   ⑤ 接收前，清空硬件缓存（ARINC429_RX_BufferClr）

   ⑥ 接收使能（ARINC429_RX_RecvEnable）

   （3）接收数据

   查询方式接收（查询方式）：

   ① 读取数据（ARINC429_RX_BufferData）

   如返回值为 2，则表示没有接收到有效的数据，再次执行此步骤。如返回值为 1，进入步骤②

   ② Length 中表示接收到有效数据的长度。在“正常模式”下，接收的数据全是 429 数据，Length 表示 Buf 中全部 429 数据的数量；在“时标模式下”，Lenth 表示接收 429 数据和其附带时标的总数量，可以得知，实际接收有效的 429 数据数量为 Length/2。如继续接收，再次执行步骤①

   中断方式接收（中断方式）：

   ① 中断到来，读取中断状态（ARINC429_INT_Status），判断如所接收通道有消息中断，执行以下②③④操作

   ② 读取接收数据（ARINC429_RX_BufferData）

   ③ 清除相应的中断状态（ARINC429_INT_StatusClear）

④ 开启总中断（ARINC429_INT_EnableGlobal）(注意：驱动中接收到硬件中断后，第一时间关闭板卡总中断，然后通过事件通知上层软件)

（4）停止接收使能（ARINC429_RX_RecvEnable）

6. 发送（指定通道）

（1）发送通道复位（ARINC429_TX_Reset）

（2）发送通道参数设置

配置波特率、奇偶校验、接收编码方式等参数（ARINC429_TX_Configure）

（3）发送数据（默认发送区为 A 区）

方式 1，正常发送：

① 启动发送器使能(ARINC429_TX_Start)

② 发送数据（ARINC429_TX_BufferData）

如返回值为 2，表示将数据写入缓存不成功（可能由于参数 Length 太大，发送缓存没有足够的空间），需再次执行此步骤，直到返回值为 1 为止，表示数据写入缓存成功

③ 重复步骤②，发送新数据

方式 2，字定时发送：

① 根据单字发送持续时间，合理设置字定时时间(ARINC429_TX_TimerIntervalWord)

单字发送时间 = 32 * 1bit 发送所需时间

② 发送数据（ARINC429_TX_BufferData）

如返回值为 2，表示将数据写入缓存不成功（可能由于参数 Length 太大，发送缓存没有足够的空间），需再次执行此步骤，直到返回值为 1 为止，表示数据写入缓存成功

③ 启动发送器使能(ARINC429_TX_Start)

④ 查询发送 FIFO 状态(ARINC429_TX_BufferStatus)，直到 FIFO 为空，然后执行步骤⑤

⑤ 关闭发送器使能(ARINC429_TX_Stop)

方式 3，帧定时发送：

① 根据单字发送持续时间，合理设置字定时时间(ARINC429_TX_TimerIntervalWord)（可选）

② 根据单帧发送持续时间，合理设置帧定时时间(ARINC429_TX_TimerIntervalFrame)

单帧发送时间 = 字定时时间 * 1 帧中数据量

③ 发送数据（ARINC429_TX_BufferData）

如返回值为 2，表示将数据写入缓存不成功（可能由于参数 Length 太大，发送缓存没有足够的空间），需再次执行此步骤，直到返回值为 1 为止，表示数据写入缓存成功

④ 启动发送器使能(ARINC429_TX_Start)
⑤ 发送过程中，可更新发送数据帧(ARINC429_TX_BufferUpdate)
⑥ 启动发送后，经过指定时间，需用户主动关闭发送器使能(ARINC429_TX_Stop)

（4）停止发送器使能（ARINC429_TX_Stop）

7. 关闭板卡

（1）全局复位板卡（ARINC429_Reset）

（2）关闭板卡（ARINC429_Close）

4. Demo 应用程序说明

ARINC429 应用程序是本公司针对 KMHT429 系列板卡产品开发的应用程序，提供数据的创建、配置、通讯、关闭，以及保存等功能。

4.1. Demo 运行环境

4.1.1. 硬件环境

可支持 USB2.0 以上总线的计算机
至少 128M 内存
显示器分辨率至少可设为 800*600
KMHT429-PCI/CPCI/PXI 通讯板卡

4.1.2. 软件环境

Windows /2000/XP/Vista/7 操作系统
KMHT429-PCI/CPCI/PXI 硬件驱动程序

4.1.3. 开发工具

Microsoft Visual Studio 2010

4.2. Demo 简介

4.2.1. 板卡号选择窗口

图 4-1 板卡号选择窗口

4.2.2. 应用程序主窗口

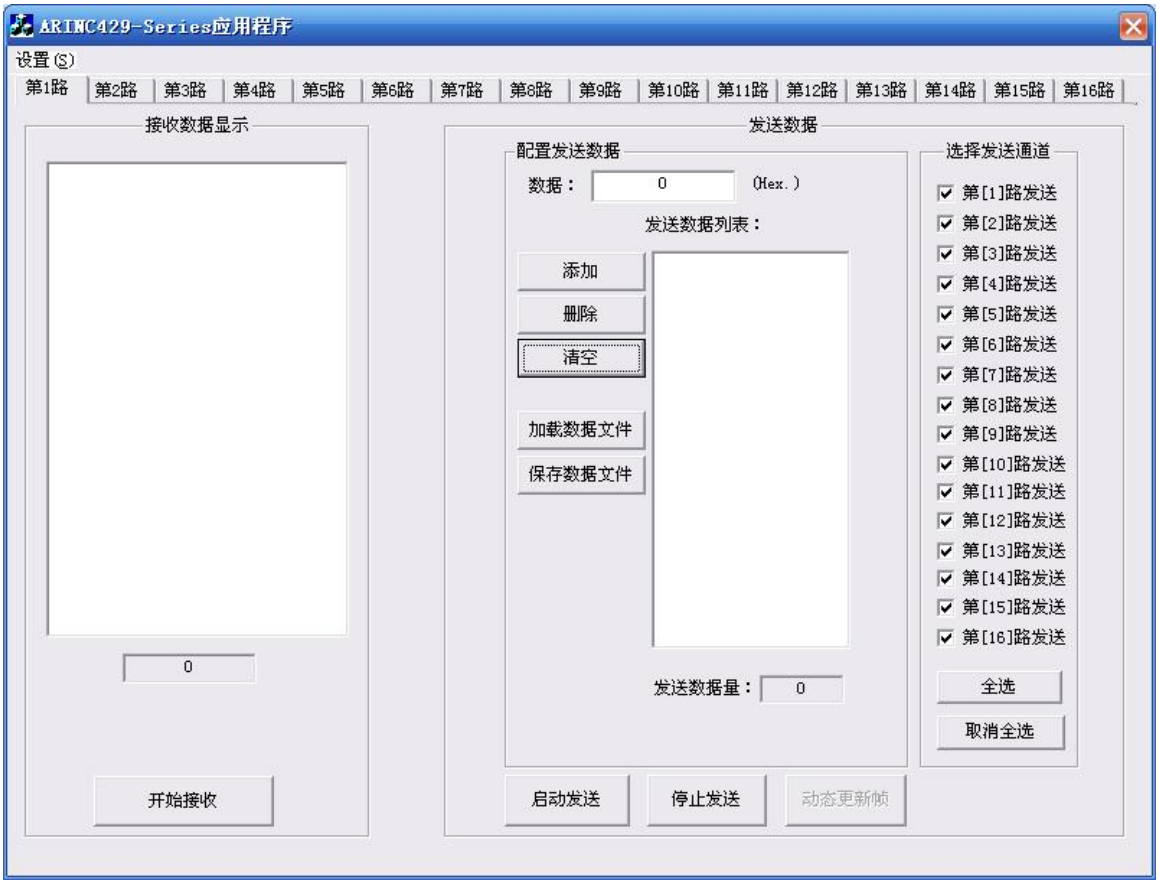


图 4-2 应用程序主窗口

4.2.3. 板卡设置窗口

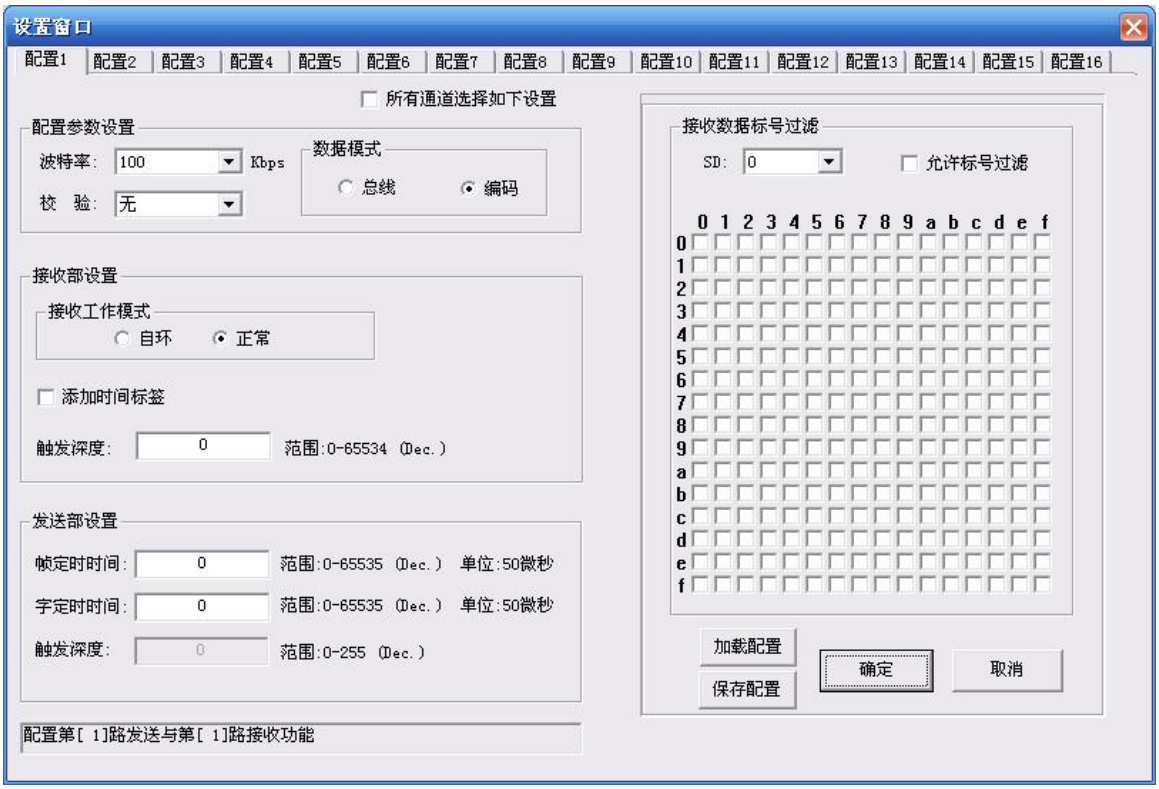


图 4-3 板卡设置窗口

4.3. 菜单功能

“设置”菜单按钮，显示板卡设置窗口（图 4-3 所示）。

4.4. 操作建议

4.4.1. 开启应用程序

在当前路径中找到“…\应用程序\ ARINC429_APP \Debug”或“…\应用程序\ ARINC429_APP \Release”中的 ARINC429.exe，直接双击运行程序。

4.4.2. 板卡号选择

图 4-4 板卡号选择窗口

应用程序启动后，需要做的第一步就是选择板卡号、选择接收路数、选择发送路数。

“板卡号”就用来指定用户当前操作的是哪块板卡，它的输入范围是从 0 到 255 的十进制数，有下列两种情况：

① 对于 PCI 接口系列板卡（包括 PCI、CPCI、PXI），板卡号按板卡所在的插槽离 CPU 的距离由近到远以次编号 0,1, …255；若 PC 机中同时只插一块板，板卡编号为 0。

② 对于 USB 接口系列板卡，板卡号与对应板卡的顺序选择关系由如下规则决定：

(1)只插入 1 块板卡 Card_A 时，板卡的板卡号为 0；

(2)再插入 1 块板卡 Card_B 时，板卡 Card_B 的板卡号为 0，板卡 Card_A 的板卡号升级为 1；

(3)再插入 1 块板卡 Card_C 时，板卡 Card_C 的板卡号为 0，板卡 Card_B 的板卡

号升级为 1，板卡 Card_A 的板卡号升级为 2；

“接收路数”和**“发送路数”**选择由待打开的硬件板卡型号决定。如型号为“ARINC429-USB-84”板卡含有 8 路接收和 4 路发送，所以“接收路数”选择 8，“发送路数”选择 4。

当用户完成板卡号的选择后，执行“确定”命令便可提交设置给应用程序，程序会自动通过该信息查找板卡，如果找到板卡且复位成功软件会进入到 ARINC429 应用程序的主窗口，等待用户执行新的操作。执行“关闭”或“取消”命令，程序会立刻退出。如果指定序号的板卡存在，则直接进入主窗口（如图 4-2）。

4.4.3. 设置板卡

在进入主窗口后，程序会自动对板卡进行默认配置，该配置结果允许用户可以进行最基本的数据发送与接收操作，默认配置内容为：① 对于接收：波特率 100K，无校验，编码方式接收，时标不使能，标号过滤不使能，自环测试不使能；② 对于发送：波特率 100K，无校验，编码方式发送，无字定时，无帧定时。

执行菜单“设置”，程序会显示设置窗口（如图 4-2 所示），这个窗口中包含了硬件所有可被设置的属性。完成设置后，单击“确定”按钮向板卡提交设置，也可以单击“取消”结束本次操作。

4.4.4. 数据通讯操作

应用程序提供了简单的数据通讯操作，如数据的接收与发送。程序会将收到的数据显示在接收数据列表中。用户可以通过窗口界面配置发送数据，还可以从文件中加载发送数据，并通过软件将数据写入发送缓冲区，再由硬件发送出去。

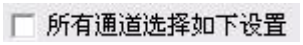
4.4.5. 退出应用程序

单击“主窗口”右上角的“X”按钮，便可退出应用程序，即出程序时有提示，确认后即可退出。

4.5. 使用说明

4.5.1. 硬件设置

4.5.1.1. 配置所有通道



将所有通道参数设置与当前通道相同。由于每一路（通道）的参数都可以单独设置，如在多路相同设置的情况下，用户需要逐一设置，比较繁琐。在这种情况下，用户可以在设置完当前页面的情况下，勾选此项，然后点击“确定”，即可将其它通道的参数设置和此通道相同。

4.5.1.2. 配置参数设置

配置参数项目如下表所示：

参数	说明
波特率	当前路设置的数据波特率（默认 100K）。包括： ① 100K； ② 50K； ③ 12.5K。
校验	奇偶校验设置（默认无校验）。包括： ① 无校验； ② 奇校验； ③ 偶校验。
数据模式	数据在总线上发送方式（默认采用编码模式）。包括： ① 编码模式； ② 总线模式。

Demo 中为了使设置简单，同一路的这三个参数进行了统一设置（尽管同一路中，接收和发送也可以设置不同的波特率、校验方式、数据模式）。如在窗口中设置的三个参数为

100K、奇校验、总线模式，则此路的接收和发送都采用这种设置。Demo 默认所有路（接收、发送）的设置为 100K、无校验、编码模式。

4.5.1.3. 接收单元参数设置

设置参数项目如下表所示：

参数	说明
接收工作模式	接收工作模式选择（默认为正常接收模式）。包括： ① 自环； ② 正常。 默认为正常接收模式，接收的数据来自外部端口的输入。自环模式下，此路接收和此路发送在硬件内部连接，接收的数据全部来自对应路的发送单元，常用于自环测试。
时间标签	时标使能（默认时标不使能）；时间标签的作用是用来描述当前接收到的这个数据的到达时间。添加时标模式下，每接收 1 个数据字，附带一个 32 位硬件时标（分辨率 50us），这种情况下，接收数据和时标间隔存储。
接收触发深度	接收硬件缓冲区中断触发深度（默认为 0），取值 0~65534。在接收采用中断的模式下使用，如设置为 N，表示此路的接收缓冲区的数据量为 N+1 时，触发中断。上层软件可通过感知中断，然后从硬件缓存中读出新接收的数据。例如，在接收采用中断模式下，接收大量数据，防止不丢数（硬件缓存溢出），需设置合理的触发深度值。 需要注意的是，如果用户设置了时间标签功能，则时间标签也会被算为数据中的一个长度。

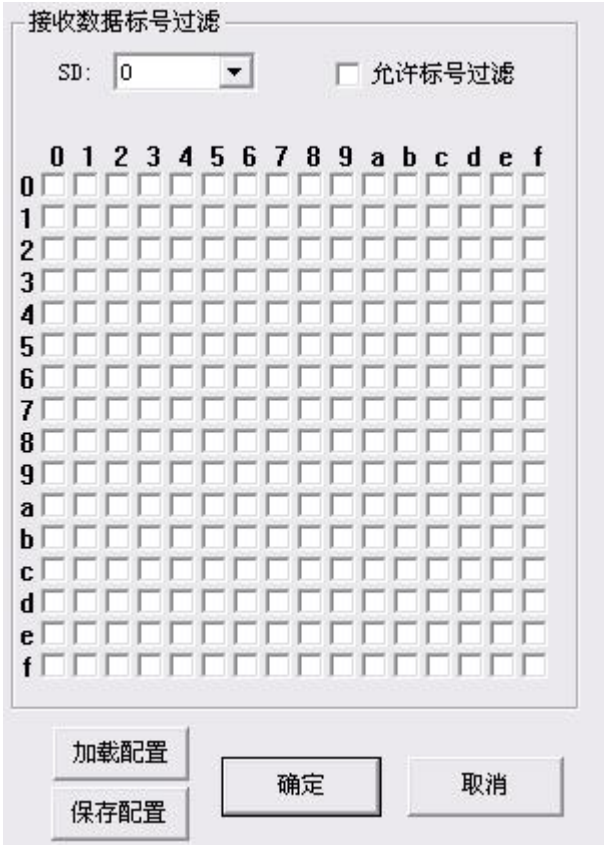
4.5.1.4. 发送单元参数设置

设置参数项目如下表所示：

参数	说明
帧定时时间	帧定时时间（分辨率 50us），取值 0~65535（默认为 0）。Dmeo 的发送按帧发送，所谓帧的概念，就是主窗口发送数据列表窗口中添加的这组数据，数量为 1~256。 如果帧定时时间设为 0，则启动发送后，此帧只发送一遍。 如果帧定时时间不为 0，则发送单元以此值为周期（设定值 * 50us），重复发送此帧。在帧重发的过程中，发送帧中的数据是固定不变的，中途不得通过软件修改。 注意，设置的帧定时时间要合理，必须不小于当前帧发送持续时间，否则数据发送不完全。如下列情况： ① 在字定时为 0 的情况下（字肩并肩发送），100K 速率下发送 10 个字，整帧发送需持续 3200us。那么设定的帧定时应不小于 3200us，否则，数据发送不完全。 ② 在字定时为 500us 的情况下，100K 速率下发送 10 个字，整帧发送需持续 5000us，那么设定的帧定时应不小于 5000us，否则，数据发送不完全。
字定时时间	字定时时间（分辨率 50us），取值 0~65535（默认为 0）。发送字定时发送时间设置。所谓字定时，是指在一帧中，前一个 DWORD 字发送的第一个 Bit 位，到下一个 DWORD 字发送的第一个 Bit 位之间的时间。 注意，字定时设置的值必须合理，如 100K 速率下，一个字发送需要 320us。如设定的值小于 320us，则设置无效。
发送触发深度	发送硬件缓冲区中断触发深度（默认为 0），取值 0~255。（由于发送暂不支持采用中断方式，此值暂无意义）在发送采用中断的模式下使用，如设置为 N，表示此路的发送缓冲区的数据量为 N+1 时，触发中断。上层软件可通过感知中断，然后发送新的数据到硬件缓冲区。例如，在发送采用中断模式下，发送大量数据，防止硬件缓存区溢出，需设置合理的触发深度

	值。

4.5.1.5. 设置接收标号过滤



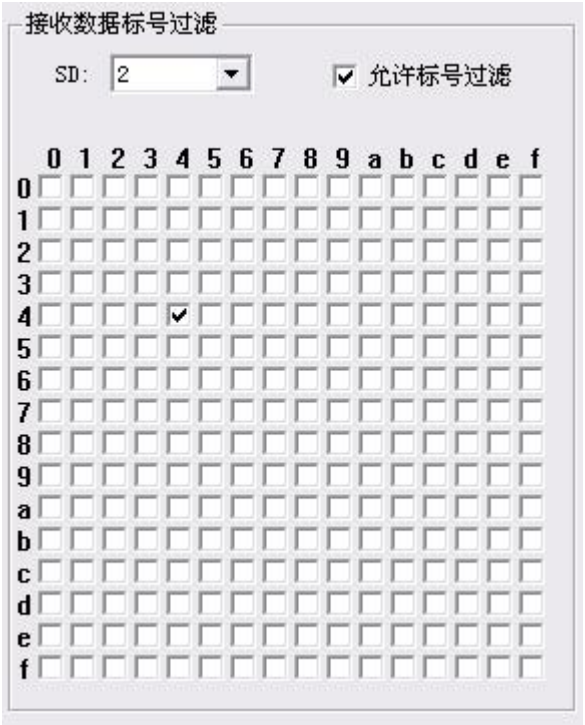
标号过滤实际上就是，通知硬件只接收那些用户想收到的数据。该功能的实现主要是依据 429 标准，通过识别数据的标号实现数据过滤功能，标号指的是 ARINC429 数据中的第 0 位到第 7 位表示的一个 8 位二进制数据。

当数据位设置为 32 位时，SD 值也参与过滤，SD 值指 ARINC429 数据中的第 11、12 两位，SD 下拉列表中的数据为 0 到 3，描述二进制数据的两个数据位。

当数据位设置为 25 位时，SD 值则不参与过滤，软件中每一个 SD 值都对应着 256 个标号。设置标号过滤时，与 32 位的有些区别，需要将 4 个 SD 值对应的 4 组标号均设置为相同的，而在 32 位数据通讯时，由于 SD 参与了过滤，则每个 SD 值都可以对应不同的标号设置。

在接收标号过滤设置中，共有 256 个标号可设置，分别是从 0 到 255。且每一个 SD 值

都有 256 个标号。标号里横向的是低 4 位，纵向的是高 4 位，如果用户要设置一个 SD 为 2 标号为 44H（十进制表示为 68）的数据过滤时，需要先将 SD 选择在 2 上，然后再在相应的标号上选中即可，操作结果如下所示：



在设置 256 个标号时当用户选中了其中的某一项，就表示对该标号的数据进行过滤，也就是会接收到该标号的数据。

单配置了标号与 SD 是不够的，只有将“允许标号过滤”选中，才可以便板卡具有标号过滤功能。

4.5.1.6. 配置的保存与加载（暂不支持）

在每次应用程序打开后，程序会自动按照默认值对板卡进行配置，实现基本的数据通讯能力，但往往用户都需要将板卡的功能配置成自己想要的功能，这样就需要经常对板卡进行配置操作，在设置窗口中的“加载配置”与“保存配置”可以提供些方便。当用户进入到配置窗口后，首次对板卡通讯进行自定义配置，配置完成后，可以执行“保存配置”，这时程序会将程序中当前的配置保存到配置文件中，以便下次重新启动应用程序后，用户

可以通过执行“加载配置”操作重新获取之前保存的配置。

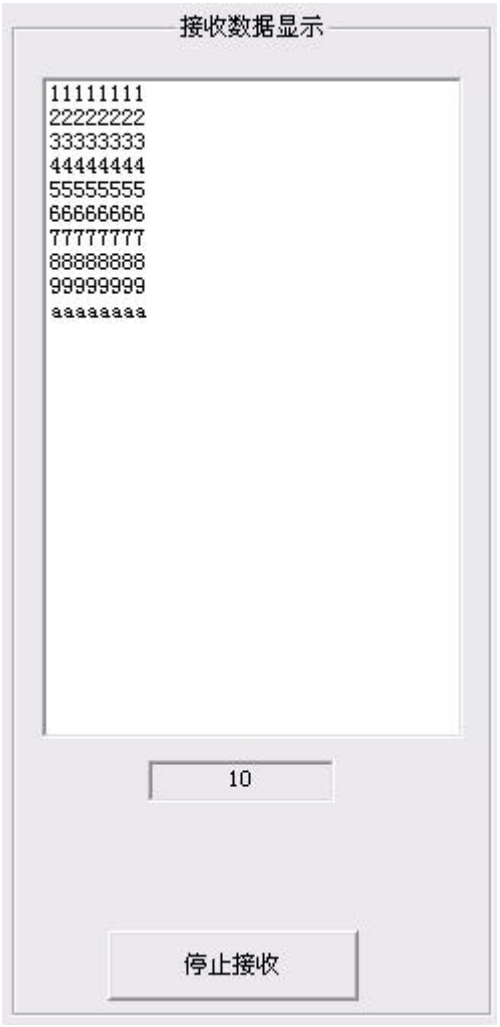
4.5.2. 数据的接收与发送

Demo 是通用型上层应用软件，适用于各种收发通道数（1~16 路）。板卡的接收通道数、发送通道数已经在“板卡号选择”窗口（如图 4-1 所示）中决定。主窗口用于实现数据通讯操作（如图 4-2 所示）。

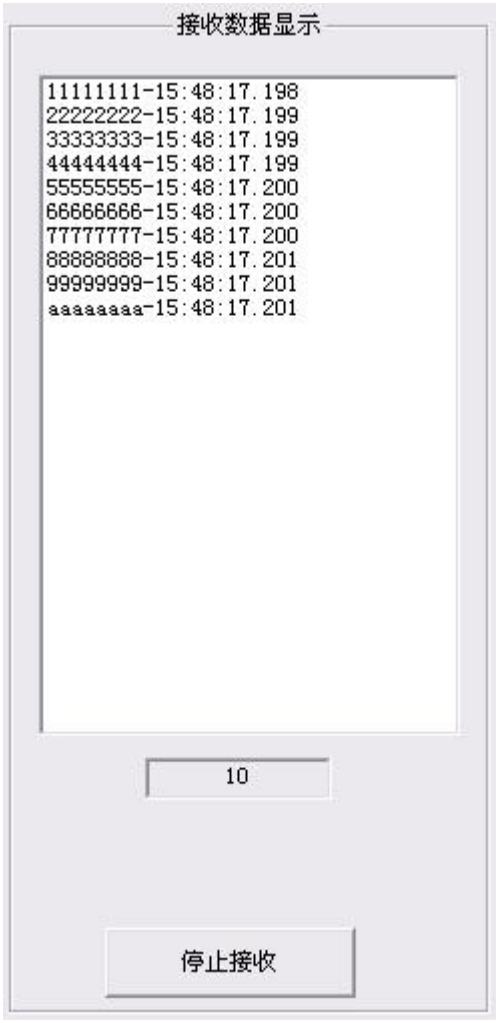
4.5.2.1. 接收数据

需要进行数据接收时，执行“开始接收”就可以了，执行后，该按钮会自动变为“停止接收”按钮，终止接收操作时，只需再次点击一下该按钮就可以了。

程序会将接收到的数据分别显示在列表窗口中，接收到的数据量会显示在列表窗口下方的文本窗口中。查看接收数据可以通过选择接收通道 Tab 卡实现。如下图所示：



接收数据时，如果添加了时标功能，在接收列表中的数据显示会有所变化。在每一个数据后面都会加上“时:分:秒.毫秒”的显示，用来表示当前数据接收到的时间。如下图所示：


接收数据显示
11111111-15:48:17.198
22222222-15:48:17.199
33333333-15:48:17.199
44444444-15:48:17.199
55555555-15:48:17.200
66666666-15:48:17.200
77777777-15:48:17.200
88888888-15:48:17.201
99999999-15:48:17.201
aaaaaaaa-15:48:17.201
10
停止接收

4.5.2.2. 发送数据

发送数据前，需要先组帧，即要保证发送数据列表不为空。添加发送数据到列表有两个方法，一个就是通过“添加”、“删除”、“清空”操作将数据文本窗口中的数据添加到列表，输入数据时要以十六进制的输入方式进行输入，列表中显示的数据也都是十六进制的。另一个方法（暂不支持）是通过“加载数据文件”和“保存数据文件”进行发送列表的配置，“加载数据文件”会指定一个数据文件，并将里面的数据以续加的方式添加到列表中，不会将列表中原有的数据清空。“保存数据文件”会将列表中的数据保存到文件中。两种配置发送数据列表的方法可以混合使用。注意，“删除”操作会将发送列表中被选中的数据删除掉，当列表中没有数据被选中时，将会从列表中的最后一个数据开始删除。

发送数据时可以通过执行“启动发送”进行数据发送，此后发送器一直处于发送状态，直到执行右边的“停止发送”按钮为止。在执行“启动发送”后，有以下两种情况：

① 如果帧定时时间为 0，则“发送数据列表”中的数据作为一帧，只发送一次。再次点击“启动发送”按钮，则“发送数据列表”中的数据再次发送一次。

② 如果帧定时时间不为 0，则“发送数据列表”中的数据作为一帧，按设置的周期（帧定时时间*50us）一直循环发送下去，直到通过执行“停止定时发送”停止发送为止。

在这种模式（帧周期发送）下，修改“发送数据列表”中的数据，然后点击“动态更新帧”，就会立即更新发送帧，且不影响原始发送周期。

动态更新帧的功能是在帧周期发送的模式下使用的。如果相应发送通道不在此模式（帧定时时间为 0）下，点击“动态更新帧”，会出现错误提示。

4.6. 编程举例

为方便开发个性化的应用程序，我公司提供了应用程序例程，该例程可从交付客户的光盘中获得。例程采用的是 VC 6.0 工具进行编写，所用的编程语言是 C 语言。这里我们仅是提供了一个简单的例子，但我们的动态链接库是可以被 Visual C++、Visual Basic、Delphi、LabView 以及 CVI 等开发工具调用的。

附录 A：429 数据格式转换

图上面的为 429 总线上的数据格式，而下面的为计算机系统软件的数据格式。即发送 FIFO 发送的数据与 429 串行数据总线上的数据进行下面的格式转换。其中 Word1 是系统数据低 16 位，而 Word2 是系统数据高 16 位。

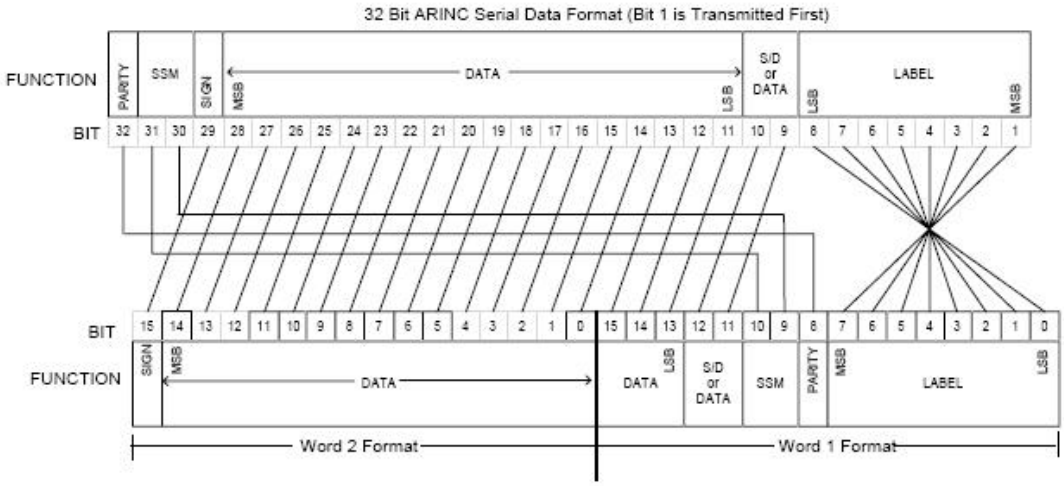


Figure 2: Mapping of Serial Data to/from Word 1 and Word 2 in 32 bit format.

例如：系统软件写数据 12345678 到发送 FIFO，奇偶校验设置位奇。则系统数据为 Word1=0X5678，Word2=0X1234。

429 串行数据格式为：0X62468A1E。