HT429-PCI-T1R2 使用手册

Ver 1.2

西安方解石信息技术有限责任公司

Revision

时间	原因	版本	作者	备注
2019-02-20	V1.0 版本生成	V1.0	冯江宏	建立文档
2019-05-07	驱动安装更改	V1.1	冯江宏	更改驱动安装方式
2019-09-30	修改部分内容	V1.2	冯江宏	文档格式，部分描述修改

1. 概述

1.1. 产品描述

HT429-PCI-T1R2 是一款包含 1 路发送 2 路接收配置的 ARINC429 通讯板卡，其功能能够满足用户的工业测量和自动化控制需求，良好的兼容性适用于各类系统配置。

1.2. 特性

- 33MHz/32Bit PCI 接口；
- 最大 4 路发送通道， 每路带 （16M-1）×32 位 FIFO；
- 最大 8 路接收通道，每路带（4M-1）×32 位 FIFO；
- 发送和接收 FIFO 复位；
- 具有内外同步时钟触发功能；
- 在内外触发时随时更新数据；
- 触发具有可设的消息间隔、字间隔和发送帧的预定数量；
- 添加时间标签功能；
- 接收标号过滤；
- 接收 FIFO 触发深度可设功能；
- 波特率 100Kbps、 50Kbps、 48Kbps、 12.5Kbps 可设置；
- 标准 ARINC429 字格式转换与否可选择；
- 支持包模式发送与接收功能。

1.3. 详细描述

1.3.1. 发送通道

- 字模式下，发送缓冲区可存放（16M-1）个 32 位数据；包模式下，发送缓冲区可存放 4095 个数据包，每个数据包最大 4095 个 32 位数据；
- 可支持外部触发、内部触发、软件触发三个触发源；
- 支持高电平触发、低电平触发、上升沿触发、下降沿触发、双沿触发；
- 触发延迟可设置，分辨率为 1us；
- 触发数据个数可设置；
- 发送数据间隔可设置；
- 包模式下，可设置包与包之间的间隔，每个发送包独立设置，分辨率为 1us；
- 发送波特率 100Kbps、 50Kbps、 48Kbps、 12.5Kbps 可设置；
- 标准 ARINC429 字格式转换与否可选择。

1.3.2. 接收通道

- 字模式下每路接收 FIFO 大小（4M-1）×32 位；包模式下，能缓存的数据包最大个数为 1023，每个数据包能够能够存取 4095 个 32 位；
- 包模式下，包间隔判定可设，分辨率为 1us；
- 接收使能和接收 FIFO 复位功能；
- 接收标号过滤功能；
- 接收添加时标功能；
- 接收波特率 100Kbps、 50Kbps、 48Kbps、 12.5Kbps 可设置；
- 标准 ARINC429 字格式转换与否可选择。

1.4. 一般规格

- 物理尺寸：长×宽： 174.63×106.68mm
- 连 接 器： SCSI-68CN
- 工作电源： 5V
- 相对湿度： 5～95%，无凝结

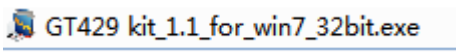
1.5. 产品安装

1.5.1. 硬件安装

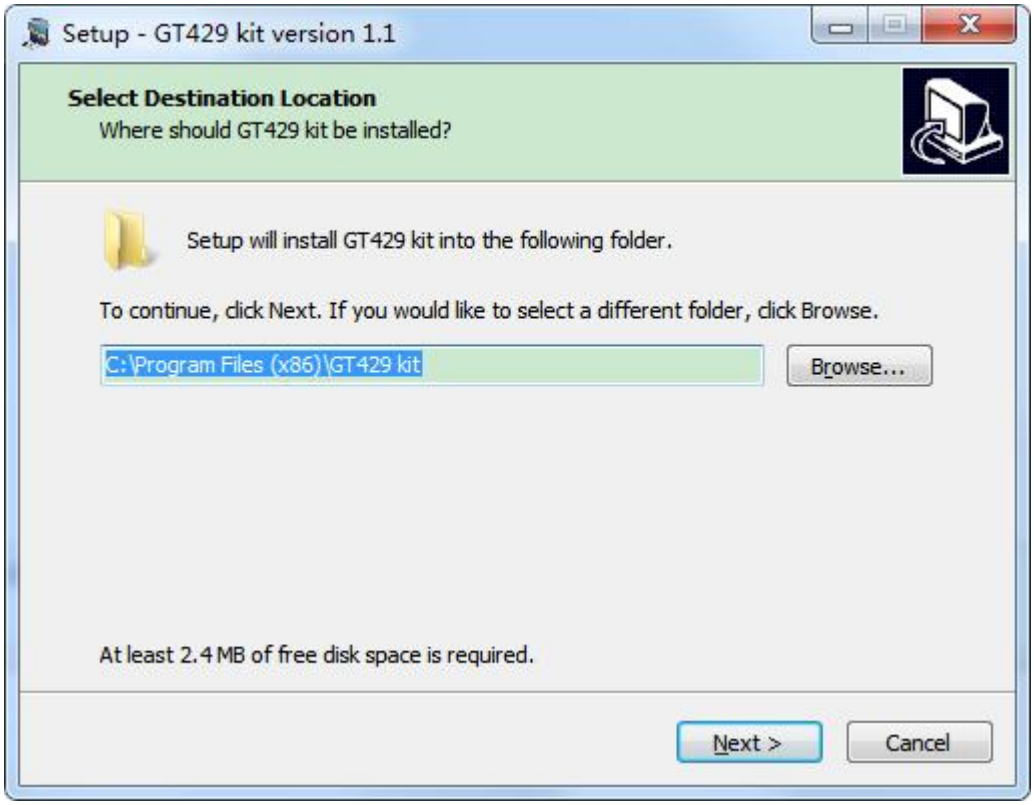
第一步： 打开板卡的防静电包装袋，取出板卡。
第二步： 关闭计算机设备的电源，将板卡安装到您的计算机机箱内。
第三步： 将配套的连接器或连接电缆插到板卡的连接器接口上。
关于连接电缆的制作请参照节 2.3 信号定义的内容。
开启计算机，系统提示发现新硬件，然后安装产品的驱动

1.5.2. 驱动安装

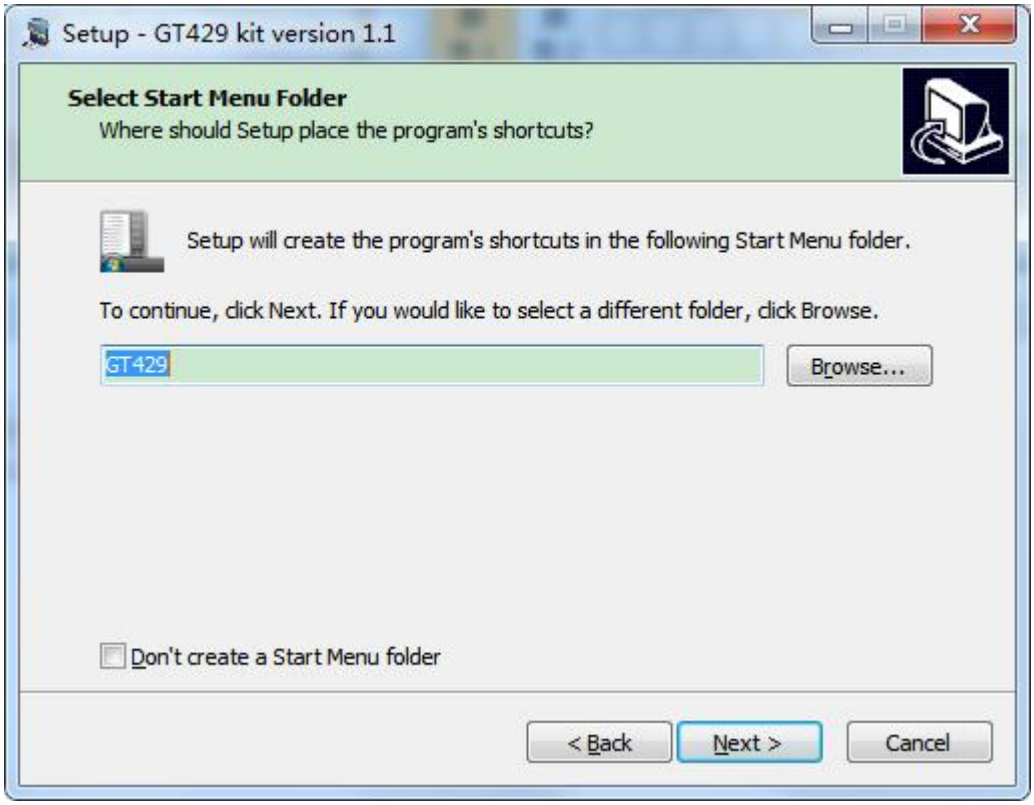
驱动程序包含下列文件（注：驱动程序版本以最新安装版本为准，此处仅作示例）：



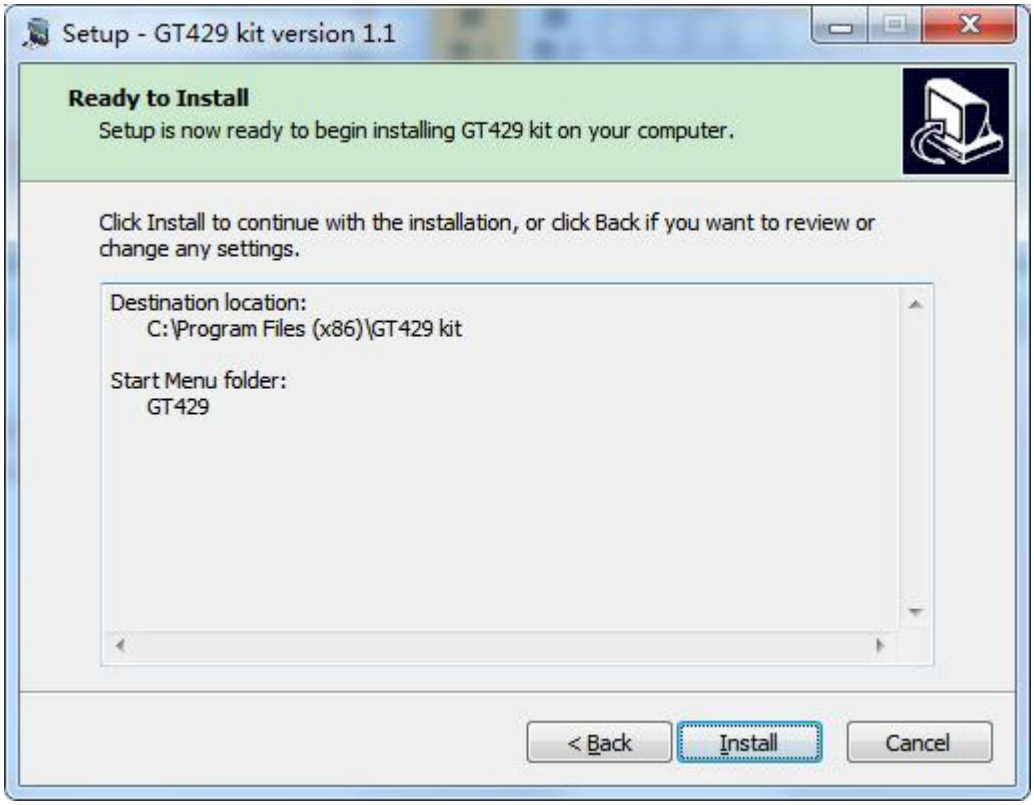
驱动安装步骤如下：
(1) 安装 429 通信卡。双击安装程序：



(2) 点击“Next”



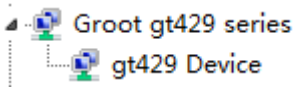
(3) 点击“Next”



(4) 点击“Install”，安装驱动程序。安装过程中若出现以下显示：

(5) 选择“始终安装此驱动程序软件”，等待安装程序完成安装。

(6) 安装驱动，完成后可以在设备管理器中看到如下图所示。



(7) 整体驱动安装完成，设备可以正常使用。

1.5.3. SDK 文件

驱动程序安装完成后，可以在安装目录的 lib 目录下得到编程使用的库文件，在安装目录的 inc 目录下可以得到编程使用的头文件， help 目录下可以看到使用说明。

2. 硬件说明

本章描述了 HT429-PCI-T1R2 板卡硬件信息，包括硬件设置及信号定义等。

2.1. 功能结构框图

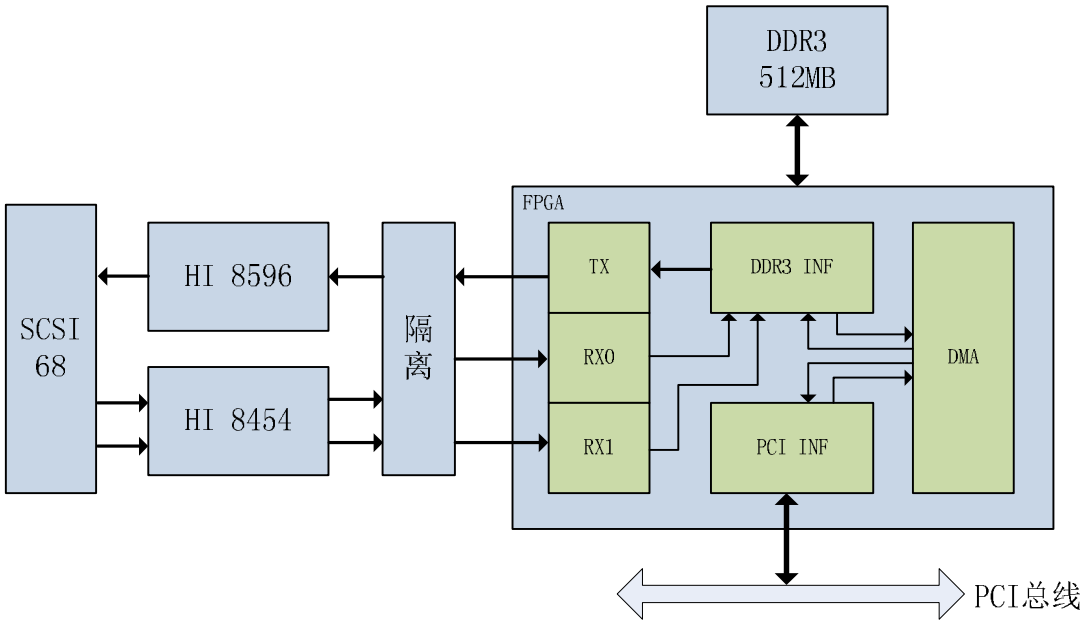


图 2-1 HT429-PCI-T1R2 功能结构框图

2.2. 印制板示意图

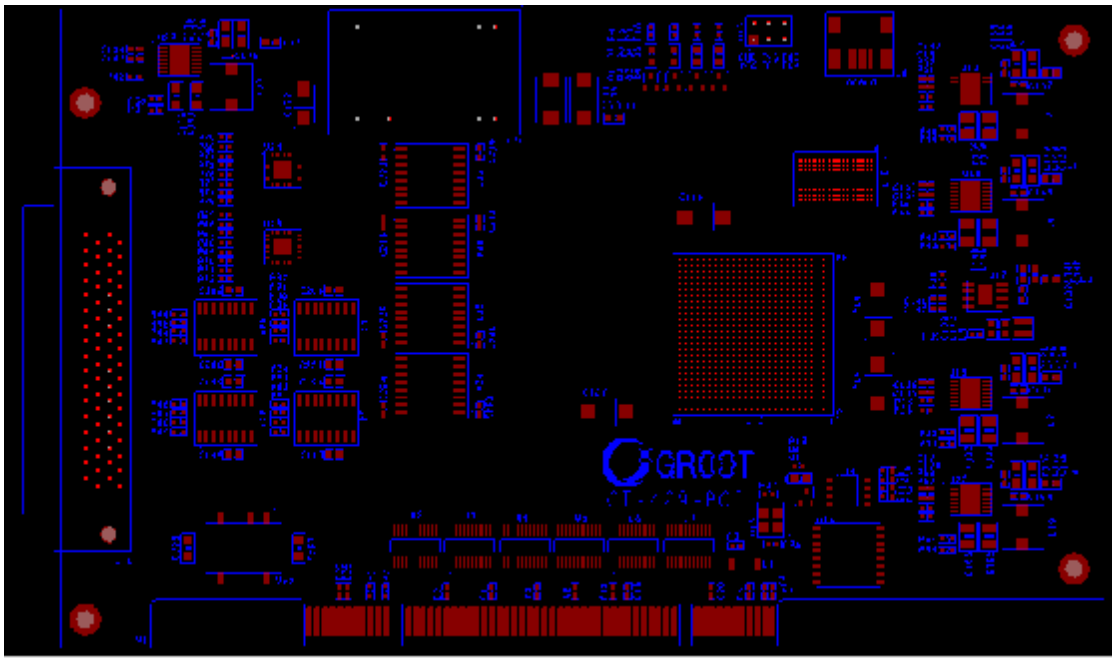


图 2-2 HT429-PCI-T1R2 印制板示意图

图 2-3 HT429-PCI-T1R2 实物图

2.3. 连接器和信号定义

连接器为标准 SCSI-68CN 型接头，如图 2-4 所示。SCSI-68CN 型接头信号列表如表 2-1 所示。

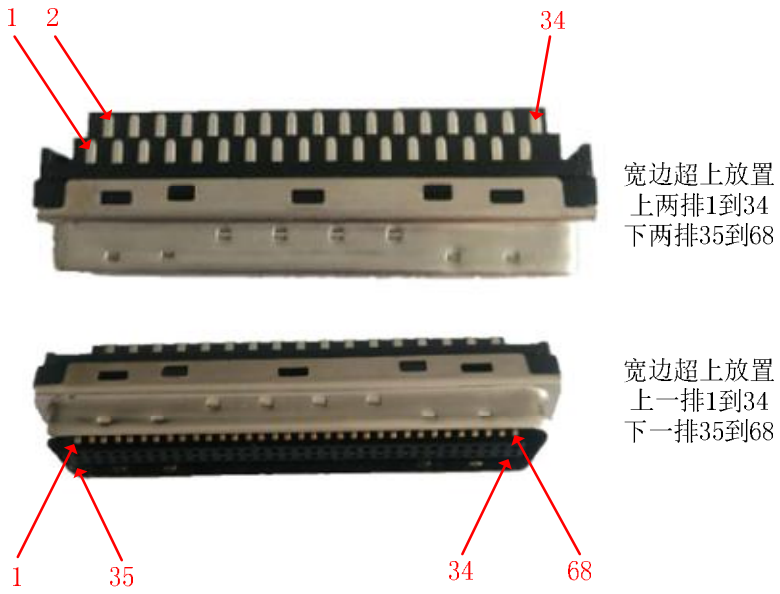


图 2-4 SCSI-68 连接器引脚标号示意图

表 2-1 SCSI-68 信号定义表

引脚	信号名	信号说明	引脚	信号名	信号说明
1	TRI_IN_0	外部触发 0	35	TRI_IN_2	外部触发 2
2	TRI_IN_1	外部触发 1	36	TRI_IN_3	外部触发 3
3	GND_TRI	触发参考地	37	GND_TRI	触发参考地
4	GND_TRI	触发参考地	38	GND_TRI	触发参考地
5			39		
6			40		
7			41		
8			42		
9	GND_429	429 参考地	43	GND_429	429 参考地
10	GND_429	429 参考地	44	GND_429	429 参考地
11	429_TXB_0	429 CH0 发送负	45		
12	429_TXA_0	429 CH0 发送正	46		
13	GND_429	429 参考地	47	GND_429	429 参考地
14	GND_429	429 参考地	48	GND_429	429 参考地
15			49		
16			50		
17	GND_429	429 参考地	51	GND_429	429 参考地
18	GND_429	429 参考地	52	GND_429	429 参考地
19	429_RXB_0	429 CH0 接收负	53	429_RXB_1	429 CH1 接收负
20	429_RXA_0	429 CH0 接收正	54	429_RXA_1	429 CH1 接收正
21	GND_429	429 参考地	55	GND_429	429 参考地
22	GND_429	429 参考地	56	GND_429	429 参考地
23			57		
24			58		

25	GND_429	429 参考地	59	GND_429	429 参考地
26	GND_429	429 参考地	60	GND_429	429 参考地
27			61		
28			62		
29	GND_429	429 参考地	63	GND_429	429 参考地
30	GND_429	429 参考地	64	GND_429	429 参考地
31			65		
32			66		
33	GND_429	429 参考地	67	GND_429	429 参考地
34	GND_429	429 参考地	68	GND_429	429 参考地

3. 宏定义

3.1. 包数据最大长度

```
#define MAX_PKT_DATA_LENGTH      4000
```

3.2. 函数返回值

Api 函数返回值统一使用 GT429_STATUS 枚举值定义。
返回值定义如下：
GT429_SUCCESS
成功
GT429_OS_ERROR
系统错误
GT429_LOW_MEMORY
内存不足
GT429_INVALID_MSG_ID
无效的消息 ID
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_BAD_PARAMETER_4
GT429_BAD_PARAMETER_5
GT429_BAD_PARAMETER_6
GT429_BAD_PARAMETER_7
GT429_BAD_PARAMETER_8
GT429_BAD_PARAMETER_9
无效的参数 1...9
GT429_NULL_DESCRIPTOR
空描述符
GT429_NO_BOARD
板卡打开失败
GT429_NOT_OPEN
板卡未打开
GT429_READ_TIMEOUT
读操作超时
GT429_WRITE_TIMEOUT
写操作超时
GT429_SIGNALED
收到系统信号

4. 枚举

4.1. 波特率枚举定义

```
typedef enum {
    EN_BAUDRATE_100KBPS = 0,
    EN_BAUDRATE_50KBPS,
    EN_BAUDRATE_48KBPS,
    EN_BAUDRATE_12P5KBPS,
    //以上为有效波特率
    EN_BAUDRATE_SEPARATOR,
    EN_BAUDRATE_BOTTOM,
    EN_BAUDRATE_UNKNOW = 0xFF,
}EN_BAUDRATE;
```

成员定义

EN_BAUDRATE_100KBPS：100 kbps

EN_BAUDRATE_50KBPS：50 kbps

EN_BAUDRATE_48KBPS：48 kbps

EN_BAUDRATE_12P5KBPS：12.5 kbps

4.2. 位长度枚举定义

```
typedef enum {
    EN_BITLEN_32 = 0,
    EN_BITLEN_25,
    EN_BITLEN_BOTTOM,
}EN_BITLEN
```

成员定义

EN_BITLEN_32：数据字使用 32 位宽

EN_BITLEN_25：数据字使用 25 位宽

4.3. 校验选择枚举定义

```
typedef enum {
    EN_PARITY_ODD = 0,
    EN_PARITY_EVEN,
    EN_PARITY_BOTTOM,
}EN_PARITY
```

成员定义

EN_PARITY_ODD：奇校验

EN_PARITY_EVEN：偶校验

4.4. 触发源枚举定义

```
typedef enum {
    EN_TRIGGER_SRC_SOFTWARE,
    EN_TRIGGER_SRC_INTERNALTIMER,
    EN_TRIGGER_SRC_EXTERNAL_0,
    EN_TRIGGER_SRC_EXTERNAL_1,
    EN_TRIGGER_SRC_EXTERNAL_2,
    EN_TRIGGER_SRC_EXTERNAL_3,
    EN_TRIGGER_SRC_BOTTOM,
}EN_TRIGGER_SRC
```

成员定义

EN_TRIGGER_SRC_SOFTWARE：软件触发，立即触发

EN_TRIGGER_SRC_INTERNALTIMER：内部定时器触发

EN_TRIGGER_SRC_EXTERNAL_0：外部触发 0

EN_TRIGGER_SRC_EXTERNAL_1：外部触发 1

EN_TRIGGER_SRC_EXTERNAL_2：外部触发 2

EN_TRIGGER_SRC_EXTERNAL_3：外部触发 3

4.5. 触发模式枚举定义

触发模式对于软件触发没有影响。

```
typedef enum {
    EN_TRIGGER_MODE_HIGHLEVEL,
    EN_TRIGGER_MODE_LOWLEVEL,
    EN_TRIGGER_MODE_POSEDGE,
    EN_TRIGGER_MODE_NEGEDGE,
    EN_TRIGGER_MODE_BOTHEDGE,
    EN_TRIGGER_MODE_BOTTOM,
}EN_TRIGGER_MODE
```

成员定义

EN_TRIGGER_MODE_HIGHLEVEL：高电平触发

EN_TRIGGER_MODE_LOWLEVEL：低电平触发

EN_TRIGGER_MODE_POSEDGE：上升沿触发

EN_TRIGGER_MODE_NEGEDGE：下降沿触发

EN_TRIGGER_MODE_BOTHEDGE：双沿触发

4.6. 缓冲区模式枚举定义

```
typedef enum {
    EN_WORD_MODE,
    EN_PKT_MODE,
    EN_BUFFER_MODE_BOTTOM,
}EN_BUFFER_MODE
```

成员定义

EN_WORD_MODE：数据字模式（默认模式）

EN_PKT_MODE：数据包模式，高级模式，需要硬件参数匹配

5. 结构体

5.1. 数据包结构

```
typedef struct pkt_struct {
    UN_PKT_HEADER header;
    GT429_UINT32 buf[MAX_PKT_DATA_LENGTH];
}ST_PKT, * ST_PKT_PTR;
```

成员定义

header：数据包头。

buf：数据包内容。

5.2. 数据包头结构

```
typedef union pkt_header {
    struct {
        GT429_UINT32 pkt_interval;
        GT429_UINT16 data_length;
        GT429_BOOL end_flag;
    }wh;
    struct {
        GT429_UINT32 pad;
        GT429_UINT16 data_length;
    }rh;
}UN_PKT_HEADER, * UN_PKT_HEADER_PTR;
```

成员定义

wh：写数据包头部。

wh.pkt_interval：下个数据包间隔，发送此数据包后在经过此时间发送下个数据包， 单位 us。

wh.data_length：数据包中的数据字个数。

wh.end_flag：此数据包是否是一个完整数据包中的最后一包。

rh：读数据包头部。

rh.pad：填充，保留。

rh.data_length：数据包中的数据字个数。

6. API 说明

6.1. 通用接口

6.1.1. GT429_OpenCard

原型

```
STDGT429CALL GT429_OpenCard(GT429HANDLE  * ph, GT429_UINT8 CardId)
```

功能

打开板卡，并分配板卡资源。

参数说明

输入参数

ph：板卡句柄指针
CardId：板卡编号，取值为 1~255。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_LOW_MEMORY
GT429_NOT_OPEN

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    GT429HANDLE ph = NULL;
    status = GT429_OpenCard(&ph, 1);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_OpenCard failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.2. GT429_CloseCard

原型

```
STDGT429CALL GT429_CloseCard(GT429HANDLE * ph)
```

功能

关闭板卡，并释放打开板卡时申请的资源。

参数说明

输入参数

ph：输入参数，板卡句柄指针

输出参数

无

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    status = GT429_CloseCard(ph);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_CloseCard failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.3. GT429_ResetCard

原型

```
STDGT429CALL GT429_ResetCard(GT429HANDLE ph)
```

功能

关闭板卡，并释放打开板卡时申请的资源。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
```

```
    status = GT429_ResetCard(ph);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_ResetCard failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.4. GT429_SetTimeTagEnable

原型

```
STDGT429CALL GT429_SetTimeTagEnable(GT429HANDLE ph, GT429_BOOL enable)
```

功能

设置是否使能时标。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
enable：GT429_TRUE -使能时标，GT429_FALSE-不使能时标

输出参数

无

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    status = GT429_SetTimeTagEnable(ph, GT429_TRUE);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_SetTimeTagEnable failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.5. GT429_GetTimeTagEnable

原型

```
STDGT429CALL GT429_GetTimeTagEnable(GT429HANDLE ph, GT429_BOOL * enable)
```

功能

获取当前是否使能了时标。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

enable：GT429_TRUE -当前时标使能，GT429_FALSE-当前时标不使能。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    GT429_BOOL enable;
    status = GT429_GetTimeTagEnable(ph, &enable);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_GetTimeTagEnable failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.6. GT429_SetTimeTagValue

原型

```
STDGT429CALL GT429_SetTimeTagValue(GT429HANDLE ph, GT429_UINT32 value)
```

功能

设置当前时标值。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

value：时标值。

输出参数

无

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    status = GT429_SetTimeTagValue(ph, 40);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_SetTimeTagValue failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.7. GT429_GetTimeTagValue

原型

```
STDGT429CALL GT429_GetTimeTagValue(GT429HANDLE ph, GT429_UINT32 *value)
```

功能

设置当前时标值。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

value：获取到的当前时标值，单位 us。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    GT429_UINT32 time_us;
    status = GT429_GetTimeTagValue(ph, &time_us);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_GetTimeTagValue failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.8. GT429_TxSetConfigure

原型

```
STDGT429CALL  GT429_TxSetConfigure(GT429HANDLE  ph,  EN_BAUDRATE  baudrate,
EN_BITLEN bitlen, GT429_BOOL parity, EN_PARITY paritysel)
```

功能

TX 通道配置。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
baudrate：波特率。
bitlen：数据位宽。
parity：是否校验。
paritysel：校验选择，奇校验或是偶校验。

输出参数

无

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_3
GT429_BAD_PARAMETER_5
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    status = GT429_TxSetConfigure(ph, EN_BAUDRATE_100KBPS, EN_BITLEN_32, GT429_TRUE, EN_PARITY_ODD);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_TxSetConfigure failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.9. GT429_TxGetConfigure

原型

```
STDGT429CALL  GT429_TxGetConfigure(GT429HANDLE  ph,  EN_BAUDRATE  *baudrate,
EN_BITLEN *bitlen, GT429_BOOL *parity, EN_PARITY *paritysel)
```

功能

获取 TX 通道配置。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

`baudrate`：TX 通道波特率。
`bitlen`：TX 通道数据位宽。
`parity`：是否校验。
`paritysel`：校验方式。

返回值:
成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_BAD_PARAMETER_4
GT429_BAD_PARAMETER_5
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    EN_BAUDRATE baudrate;
    EN_BITLEN bitlen;
    GT429_BOOL parity;
    EN_PARITY paritysel;
    status = GT429_TxGetConfigure(ph, &baudrate, &bitlen, &parity, &paritysel);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_TxGetConfigure failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.10. GT429_TxSetWordConvertEnable

原型

```
STDGT429CALL GT429_TxSetWordConvertEnable(GT429HANDLE ph, GT429_BOOL enable)
```

功能

设置 429 发送通道字格式转换是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
enable：GT429_TRUE -使能字格式转换，GT429_FALSE-不使能字格式转换。

输出参数

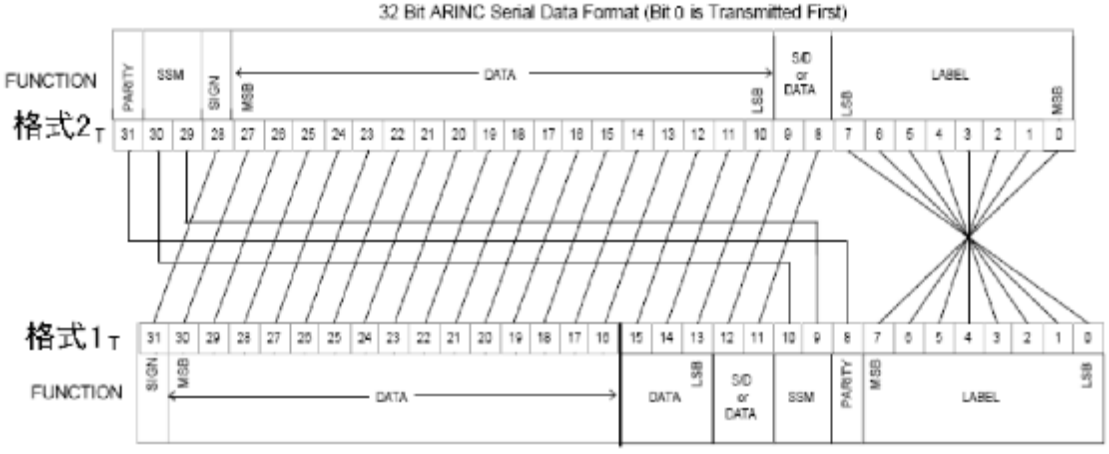
无

返回值:

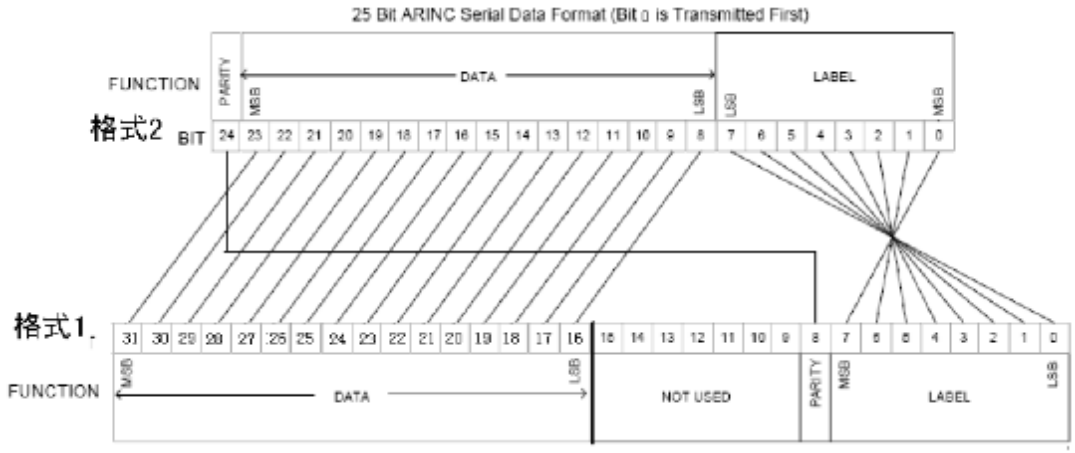
成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

备注

	转换	不转换
32 位发送	PC 数据与 429 总线数据都为格式 2。串行数据最高位可以设置成校验位或数据位。	PC 数据格式 1 429 总线数据格式 2
32 位接收	PC 数据与 429 总线数据都为格式 2。接收最高位是奇校验位，‘0’：表示校验正确。	
25 位发送	PC 数据低 25 位与 429 总线数据都为格式 2。串行数据最高位可以设置成校验位或数据位。	PC 数据格式 1 429 总线数据格式 2
25 位接收	PC 数据低 25 位与 429 总线数据都为格式 2。接收 bit24 是奇校验位，‘0’：表示校验正确。	



32 位转换



25 位转换

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    status = GT429_TxSetWordConvertEnable(ph, enable);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_TxSetWordConvertEnable failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

6.1.11. GT429_TxGetWordConvertEnable

原型

```
STDGT429CALL GT429_TxGetWordConvertEnable(GT429HANDLE ph, GT429_BOOL * enable)
```

功能

获取 429 发送通道字格式转换是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

enable：GT429_TRUE - 使能字格式转换， GT429_FALSE - 不使能字格式转换。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
```

```
    GT429_BOOL enable;
    status = GT429_TxGetWordConvertEnable(ph, &enable);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_TxGetWordConvertEnable, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

6.1.12. GT429_RxSetConfigure

原型

```
STDGT429CALL GT429_RxSetConfigure(GT429HANDLE ph, GT429_UINT8 channel, EN_BAUDRATE baudrate, EN_BITLEN bitlen)
```

功能

TX 通道配置。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
baudrate：波特率。
bitlen：数据位宽。

输出参数

无

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_BAD_PARAMETER_4
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_RxSetConfigure(ph, 0, EN_BAUDRATE_100KBPS, EN_BITLEN_32);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxSetConfigure failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.13. GT429_RxGetConfigure

原型

```
STDGT429CALL GT429_RxGetConfigure(GT429HANDLE ph, GT429_UINT8 channel, EN_BAUDRATE *baudrate, EN_BITLEN *bitlen)
```

功能

获取 RX 通道配置。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。

输出参数

`baudrate`：TX 通道波特率。
`bitlen`：TX 通道数据位宽。

返回值:

成功时返回 `GT429_SUCCESS`。
失败时可能返回的返回值列表：
`GT429_BAD_PARAMETER_1`
`GT429_BAD_PARAMETER_2`
`GT429_BAD_PARAMETER_3`
`GT429_BAD_PARAMETER_4`
`GT429_NULL_DESCRIPTOR`
`GT429_NO_BOARD`
`GT429_OS_ERROR`

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    EN_BAUDRATE baudrate;
    EN_BITLEN bitlen;
    status = GT429_RxGetConfigure(ph, 0, &baudrate, &bitlen);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_RxGetConfigure failed. Return value = %d\r\n", status);
        return status;
    }
```

6.1.14. GT429_RxSetWordConvertEnable

原型

```
STDGT429CALL GT429_RxSetWordConvertEnable(GT429HANDLE ph, GT429_UINT8 channel, GT429_BOOL enable)
```

功能

设置 429 接收通道字格式转换是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
enable：GT429_TRUE -使能字格式转换，GT429_FALSE-不使能字格式转换。

输出参数

无

返回值:
成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    status = GT429_RxSetWordConvertEnable(ph, 0, GT429_FALSE);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_RxSetWordConvertEnable failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

6.1.15. GT429_RxGetWordConvertEnable

原型

```
STDGT429CALL GT429_RxGetWordConvertEnable(GT429HANDLE ph, GT429_UINT8 channel,
GT429_BOOL * enable)
```

功能

获取 429 接收通道字格式转换是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。

输出参数

enable：GT429_TRUE - 使能字格式转换， GT429_FALSE - 不使能字格式转换。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_BAD_PARAMETER_3

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
    GT429_STATUS status = GT429_SUCCESS;
    GT429_BOOL enable;
    status = GT429_RxGetWordConvertEnable(ph, 0, &enable);
    if (status != GT429_SUCCESS) {
        debug_printf("GT429_RxGetWordConvertEnable, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

6.1.16. GT429_TxSetEnable

原型

```
STDGT429CALL GT429_TxSetEnable(GT429HANDLE ph, GT429_BOOL enable)
```

功能

设置 429 发送通道使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

enable：GT429_TRUE -使能，GT429_FALSE-不使能。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxSetEnable(ph, GT429_TRUE);
if (status != GT429_SUCCESS) {
```

```
    debug_printf("GT429_TxSetEnable failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.17. GT429_TxGetEnable

原型

```
STDGT429CALL GT429_TxGetEnable(GT429HANDLE ph, GT429_BOOL * enable)
```

功能

获取 429 发送通道是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

enable：GT429_TRUE - 使能， GT429_FALSE - 不使能。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_BOOL enable;
status = GT429_TxGetEnable(ph, &enable);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetEnable, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.18. GT429_TxStart

原型

```
STDGT429CALL GT429_TxStart(GT429HANDLE ph)
```

功能

429 发送通道启动，此函数需要在 TX 配置完成后调用。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxStart(ph);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxStart failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.19. GT429_TxReset

原型

```
STDGT429CALL GT429_TxReset(GT429HANDLE ph)
```

功能

429 发送通道复位。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxReset(ph);
```

```
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxReset failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.20. GT429_TxSetTriggerConfigure

原型

```
STDGT429CALL GT429_TxSetTriggerConfigure(GT429HANDLE ph, EN_TRIGGER_SRC src,
EN_TRIGGER_MODE mode, GT429_UINT16 count, GT429_UINT32 depth)
```

功能

429 发送通道触发配置。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
src：触发源。
mode：触发模式。
count：触发次数。
depth：触发深度。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxSetTriggerConfigure(ph, EN_TRIGGER_SRC_SOFTWARE, EN_TRIGGER_MODE_HIGHLEVEL, 1, 0);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxSetTriggerConfigure failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.21. GT429_TxGetTriggerConfigure

原型

```
STDGT429CALL GT429_TxGetTriggerConfigure(GT429HANDLE ph, EN_TRIGGER_SRC * src,
EN_TRIGGER_MODE * mode, GT429_UINT16 * count, GT429_UINT32 * depth)
```

功能

获取 429 通道触发配置。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

src：触发源。
mode：触发模式。
count：触发次数。
depth：触发深度。
无。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_BAD_PARAMETER_4
GT429_BAD_PARAMETER_5
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
EN_TRIGGER_SRC src;
EN_TRIGGER_MODE mode;
GT429_UINT16 count;
GT429_UINT32 depth;
status = GT429_TxGetTriggerConfigure(ph, &src, &mode, &count, &depth);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetTriggerConfigure failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.22. GT429_TxSetBufferMode

原型

```
STDGT429CALL GT429_TxSetBufferMode(GT429HANDLE ph, EN_BUFFER_MODE mode)
```

功能

429 发送通道设置缓冲区模式。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
mode：缓冲区模式。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxSetBufferMode(ph, EN_WORD_MODE);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxSetBufferMode failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.23. GT429_TxGetBufferMode

原型

```
STDGT429CALL GT429_TxGetBufferMode(GT429HANDLE ph, EN_BUFFER_MODE * mode)
```

功能

获取 429 发送通道缓冲区模式。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

mode：缓冲区模式。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
EN_BUFFER_MODE mode;
status = GT429_TxGetBufferMode(ph, &mode);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetBufferMode failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.24. GT429_TxSetInternalTriggerTimeEnable

原型

```
STDGT429CALL GT429_TxSetInternalTriggerTimeEnable(GT429HANDLE ph, GT429_BOOL enable)
```

功能

设置 429 发送通道使能内部触发定时器。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

enable：GT429_TRUE -使能，GT429_FALSE-不使能。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
```

```
status = GT429_TxSetInternalTriggerTimeEnable(ph, GT429_TRUE);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxSetInternalTriggerTimeEnable failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.25. GT429_TxGetInternalTriggerTimeEnable

原型

```
STDGT429CALL GT429_TxGetInternalTriggerTimeEnable(GT429HANDLE ph, GT429_BOOL *enable)
```

功能

获取 429 发送通道内部触发定时器是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

enable：GT429_TRUE - 使能， GT429_FALSE - 不使能。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_BOOL enable;
status = GT429_TxGetInternalTriggerTimeEnable(ph, &enable);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetInternalTriggerTimeEnable, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.26. GT429_TxSetInternalTriggerTimePeriod

原型

```
STDGT429CALL GT429_TxSetInternalTriggerTimePeriod(GT429HANDLE ph, GT429_UINT32 period)
```

功能

设置 429 发送通道内部触发定时器周期。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
period：周期值，单位(1/100M)S。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxSetInternalTriggerTimePeriod(ph, 1000);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxSetInternalTriggerTimePeriod failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.27. GT429_TxGetInternalTriggerTimePeriod

原型

```
STDGT429CALL GT429_TxGetInternalTriggerTimePeriod(GT429HANDLE ph, GT429_UINT32 * period)
```

功能

获取 429 发送通道内部触发定时器周期。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

period：周期值，单位(1/100M)S。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 period;
status = GT429_TxGetInternalTriggerTimePeriod(ph, &period);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetInternalTriggerTimePeriod, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.28. GT429_TxSetTriggerDelay

原型

```
STDGT429CALL GT429_TxSetTriggerDelay(GT429HANDLE ph, GT429_UINT32 delay)
```

功能

设置 429 发送通道触发延时。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
delay：延迟值，单位 1 us。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxSetTriggerDelay(ph, 1000);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxSetTriggerDelay failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.29. GT429_TxGetTriggerDelay

原型

```
STDGT429CALL GT429_TxGetTriggerDelay(GT429HANDLE ph, GT429_UINT32 * delay)
```

功能

获取 429 发送通道内部触发延时。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

delay：延时值，单位 1 us。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 delay;
status = GT429_TxGetTriggerDelay(ph, &delay);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetTriggerDelay, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.30. GT429_TxInternalTriggerTimeReset

原型

```
STDGT429CALL GT429_TxInternalTriggerTimeReset(GT429HANDLE ph)
```

功能

复位 429 发送通道内部触发定时器。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxInternalTriggerTimeReset(ph);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxInternalTriggerTimeReset, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.31. GT429_RxReset

原型

```
STDGT429CALL GT429_RxReset(GT429HANDLE ph, GT429_UINT8 channel)
```

功能

429 接收通道复位。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

channel：RX 通道，取值 0 或 1。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_RxReset(ph, 0);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxReset failed. Return value = %d\r\n", status);
    return status;
```

```
}
```

6.1.32. GT429_RxSetEnable

原型

```
STDGT429CALL GT429_RxSetEnable(GT429HANDLE ph, GT429_UINT8 channel, GT429_BOOL enable)
```

功能

设置 429 接收通道是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
enable：GT429_TRUE -使能，GT429_FALSE-不使能。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_RxSetEnable(ph, 0, GT429_FALSE);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxSetEnable failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.33. GT429_RxGetEnable

原型

```
STDGT429CALL GT429_RxGetEnable(GT429HANDLE ph, GT429_UINT8 channel, GT429_BOOL * enable)
```

功能

获取 429 接收通道是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。

输出参数

enable：GT429_TRUE - 使能， GT429_FALSE - 不使能。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_BOOL enable;
status = GT429_RxGetEnable(ph, 0, &enable);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxGetEnable, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.34. GT429_RxSetBufferMode

原型

```
STDGT429CALL GT429_RxSetBufferMode(GT429HANDLE ph, GT429_UINT8 channel, EN_BUFFER_MODE mode)
```

功能

429 接收通道设置缓冲区模式。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
mode：缓冲区模式。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_RxSetBufferMode(ph, 0, EN_WORD_MODE);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxSetBufferMode failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.35. GT429_RxGetBufferMode

原型

```
STDGT429CALL GT429_RxGetBufferMode(GT429HANDLE ph, GT429_UINT8 channel, EN_BUFFER_MODE * mode)
```

功能

获取 429 接收通道缓冲区模式。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。

输出参数

mode：缓冲区模式。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
EN_BUFFER_MODE mode;
```

```
status = GT429_RxGetBufferMode(ph, 0, &mode);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxGetBufferMode failed. Return value = %d\r\n", status);
    return status;
}
```

6.1.36. GT429_RxSetFilterEnable

原型

```
STDGT429CALL GT429_RxSetFilterEnable(GT429HANDLE ph, GT429_UINT8 channel, GT429_BOOL enable)
```

功能

设置 429 接收通道是否使能过滤。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
enable：GT429_TRUE -使能，GT429_FALSE-不使能。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_RxSetFilterEnable(ph, 0, GT429_FALSE);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxSetFilterEnable failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.37. GT429_RxGetFilterEnable

原型

```
STDGT429CALL GT429_RxGetFilterEnable(GT429HANDLE ph, GT429_UINT8 channel,
```

```
GT429_BOOL * enable)
```

功能

获取 429 接收通道是否使能过滤。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。

输出参数

enable：GT429_TRUE - 使能， GT429_FALSE - 不使能。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_BOOL enable;
status = GT429_RxGetFilterEnable(ph, 0, &enable);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxGetFilterEnable, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.1.38. GT429_RxSetFilter

原型

```
STDGT429CALL GT429_RxSetFilter(GT429HANDLE ph, GT429_UINT8 channel, GT429_BOOL
filter[4][256])
```

功能

设置 429 接收通道过滤表。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
filter：标号过滤表，数组的行坐标代表 S/D 号，列坐标代表标号； 若数组对应的元素的值为 GT429_TRUE， 则 S/D 号对应的标号参与过滤。例如：filter[1][2] = GT429_TRUE 表

示 S/D 为 1 且 label 为 2 的 429 数据参与过滤，即被接收。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_BOOL filter[4][256]={GT429_TRUE};
status = GT429_RxSetFilter (ph, 0, filter);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxSetFilter failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.2. 字模式操作接口

6.2.1. GT429_TxWordClr

原型

```
STDGT429CALL GT429_TxWordClr(GT429HANDLE ph)
```

功能

429 发送通道清零字模式缓冲区。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxWordClr(ph);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxWordClr failed. Return value = %d\r\n", status);
    return status;
}
```

6.2.2. GT429_TxSetWordInterval

原型

```
STDGT429CALL GT429_TxSetWordInterval(GT429HANDLE ph, GT429_UINT32 interval)
```

功能

429 发送通道设置字发送间隔。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
interval：发送间隔，单位 us。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxSetWordInterval(ph, 100);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxSetWordInterval failed. Return value = %d\r\n", status);
    return status;
}
```

6.2.3. GT429_TxGetWordInterval

原型

```
STDGT429CALL GT429_TxGetWordInterval(GT429HANDLE ph, GT429_UINT32 * interval)
```

功能

获取 429 通道发送间隔。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

interval：发送间隔。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 interval;
status = GT429_TxGetWordInterval(ph, &interval);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetWordInterval failed. Return value = %d\r\n", status);
    return status;
}
```

6.2.4. GT429_TxGetRemainWordCount

原型

```
STDGT429CALL GT429_TxGetRemainWordCount(GT429HANDLE ph, GT429_UINT32 * count)
```

功能

获取 429 发送通道缓冲区还可写入的 32 位字个数。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

count：还可写入的 32 位字个数。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 count;
status = GT429_TxGetRemainWordCount(ph, &count);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetRemainWordCount, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.2.5. GT429_RxGetWordCount

原型

```
STDGT429CALL  GT429_RxGetWordCount(GT429HANDLE  ph,  GT429_UINT8  channel,
GT429_UINT32 * count)
```

功能

获取 429 接收通道中当前 32 位数据字个数。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

channel：RX 通道，取值 0 或 1。

输出参数

count：接收通道中当前 32 位数据字个数。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_BAD_PARAMETER_3

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 count;
status = GT429_RxGetWordCount(ph, 0, &count);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxGetWordCount failed. Return value = %d\r\n", status);
    return status;
}
```

6.2.6. GT429_SendWord

原型

```
STDGT429CALL GT429_SendWord(GT429HANDLE ph, GT429_UINT16 len, GT429_UINT32 *
buf)
```

功能

429 发送数据字。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
len：数据字个数。
buf：数据缓冲区。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 wbuf[4] = { 0xAAAAAAAA, 0x55555555, 0x7EE77EE7, 0xE77EE77E };
status = GT429_SendWord (ph, 4, wbuf);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_SendWord failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.2.7. GT429_ReceiveWord

原型

```
STDGT429CALL    GT429_ReceiveWord(GT429HANDLE    ph,    GT429_UINT8    channel,
GT429_UINT16 len, GT429_UINT16 * olen, GT429_UINT32 * buf, GT429_UINT32 timeout)
```

功能

429 接收数据字。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
len：数据字个数。
timeout：超时时间，单位 ms。

输出参数

buf：接收数据缓冲区。
Olen：实际读出的数据个数，每个为 32 位数据字

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_4
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
GT429_READ_TIMEOUT
GT429_SIGNALED
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 rbuf[4];
GT429_UINT16 olen;
status = GT429_ReceiveWord(handle, 0, 8, &olen, r0buf, 10000);
if (status != GT429_SUCCESS) {
    debug_printf ("GT429_ReceiveWord failed. Return value =  0x%08x!!!\r\n", status);
    return status;
}
```

6.2.8. GT429_ReceiveWordWithTs

原型

```
STDGT429CALL GT429_ReceiveWordWithTs(GT429HANDLE ph, GT429_UINT8 channel,
GT429_UINT16 len, GT429_UINT16 * olen, GT429_UINT32 * buf, GT429_UINT32 timeout)
```

功能

429 接收数据字和时标。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
len：数据字个数。
timeout：超时时间，单位 ms。

输出参数

buf：接收数据缓冲区。保存数据字和时标。顺序为| 32 位数据字 0 | 32 位数据字 0 时标 | 32 位数据字 1 | 32 位数据字 1 时标 | .....
Olen：实际读出的数据个数，每个为 32 位

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_4
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
GT429_READ_TIMEOUT
GT429_SIGNALED
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 rbuf[4];
GT429_UINT16 olen;
status = GT429_ReceiveWord(handle, 0, 8, &olen, r0buf, 10000);
if (status != GT429_SUCCESS) {
    debug_printf ("GT429_ReceiveWord failed. Return value = 0x%08x!!!\r\n", status);
    return status;
}
```

6.3. 包模式操作接口

6.3.1. GT429_TxPktClr

原型

```
STDGT429CALL GT429_TxPktClr(GT429HANDLE ph)
```

功能

包操作接口，429 发送通道清零包模式缓冲区。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_TxPktClr(ph);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxPktClr failed. Return value = %d\r\n", status);
    return status;
}
```

6.3.2. GT429_TxGetRemainPktCount

原型

```
STDGT429CALL GT429_TxGetRemainPktCount(GT429HANDLE ph, GT429_UINT32 * count)
```

功能

包操作接口，获取 429 发送通道缓冲区还可写入的包个数。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

输出参数

count：还可写入的包个数。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD

GT429_OS_ERROR

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 count;
status = GT429_TxGetRemainPktCount(ph, &count);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_TxGetRemainPktCount, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.3.3. GT429_RxSetPktIdleTime

原型

```
STDGT429CALL GT429_RxSetBufferMode(GT429HANDLE ph, GT429_UINT8 channel, GT429_UINT32 time)
```

功能

包操作接口，429 接收通道设置包分割时间。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。

channel：RX 通道，取值 0 或 1。

time：包分割时间，单位 us。

输出参数

无。

返回值

成功时返回 GT429_SUCCESS。

失败时可能返回的返回值列表：

GT429_BAD_PARAMETER_1

GT429_BAD_PARAMETER_2

GT429_BAD_PARAMETER_3

GT429_NULL_DESCRIPTOR

GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
status = GT429_RxSetPktIdleTime(ph, 0, 1000);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxSetPktIdleTime failed. Return value = %d\r\n", status);
    return status;
}
```

6.3.4. GT429_RxGetPktIdleTime

原型

```
STDGT429CALL GT429_RxGetPktIdleTime(GT429HANDLE ph, GT429_UINT8 channel, GT429_UINT32 * time)
```

功能

包操作接口，获取 429 接收通道包分割时间。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。

输出参数

time：包分割时间，单位 us。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 time;
status = GT429_RxGetPktIdleTime(ph, 0, &time);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxGetPktIdleTime failed. Return value = %d\r\n", status);
    return status;
}
```

6.3.5. GT429_RxGetPktCount

原型

```
STDGT429CALL GT429_RxGetPktCount(GT429HANDLE ph, GT429_UINT8 channel, GT429_UINT32 * count)
```

功能

包操作接口，获取 429 接收通道中当前包个数。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。

输出参数

count：包个数。

返回值

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 count;
status = GT429_RxGetPktCount(ph, 0, &count);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_RxGetPktCount failed. Return value = %d\r\n", status);
    return status;
}
```

6.3.6. GT429_SendPacket

原型

```
STDGT429CALL GT429_SendPacket(GT429HANDLE ph, ST_PKT_PTR ppkt)
```

功能

包操作接口，429 发送数据包。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
ppkt：用户包结构指针，由用户分配并初始化。

输出参数

无

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
GT429_UINT32 wbuf[4] = { 0xAAAAAAAA, 0x55555555, 0x7EE77EE7, 0xE77EE77E };
ST_PKT wpkt;
//设置包模式
status = GT429_TxSetBufferMode(handle, EN_PKT_MODE);
if (status != GT429_SUCCESS) {
    debug_printf ("GT429_TxSetBufferMode failed. Return value = %d\r\n", status);
    return status;
}
wpkt.header.wh.data_length = 4;
wpkt.header.wh.end_flag = GT429_TRUE;
wpkt.header.wh.pkt_interval = 10;
memcpy(wpkt.buf, wbuf, sizeof(wbuf));
status = GT429_SendPacket (ph, &wpkt);
if (status != GT429_SUCCESS) {
    debug_printf("GT429_SendPacket failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

6.3.7. GT429_ReceivePacket

原型

```
STDGT429CALL GT429_ReceivePacket(GT429HANDLE ph, GT429_UINT8 channel, ST_PKT_PTR ppkt, GT429_UINT32 timeout)
```

功能

包操作接口，429 发送数据包。

参数说明

输入参数

ph：板卡句柄，由函数 GT429_OpenCard 得到。
channel：RX 通道，取值 0 或 1。
timeout：超时时间，单位 ms。

输出参数

ppkt：用户接收包结构缓冲区，由用户分配。

返回值:

成功时返回 GT429_SUCCESS。
失败时可能返回的返回值列表：
GT429_BAD_PARAMETER_1
GT429_BAD_PARAMETER_2
GT429_BAD_PARAMETER_3
GT429_BAD_PARAMETER_4
GT429_NULL_DESCRIPTOR
GT429_NO_BOARD
GT429_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT429_STATUS status = GT429_SUCCESS;
ST_PKT rpkt;
//设置包模式
status = GT429_RxSetBufferMode(handle, 0, EN_PKT_MODE);
if (status != GT429_SUCCESS) {
    debug_printf ("GT429_TxSetBufferMode failed. Return value = 0x%08x!!!\r\n", status);
    return status;
}
status = GT429_ReceivePacket(handle, 0, &r0pkt, 10000);
if (status != GT429_SUCCESS) {
    debug_printf ("GT429_ReceivePacket failed. Return value = 0x%08x!!!\r\n", status);
    return status;
}
```