HT1553B-PCI-C4 使用手册

Ver 1.3

西安方解石信息技术有限责任公司

029-81151932

Revision

时间	原因	版本	作者	备注
2019-01-22	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
2019-06-11	修改循环缓冲区长度接口描述	V1.2	冯江宏	RT 循环缓冲区长度接口描述更改
2019-09-30	修改部分描述	V1.3	冯江宏	更改部分描述

1. 概述

MIL-STD-1553 是美国国防部颁布的一个军用标准，定义了一个串行数据总线的机械，电子和功能特性。它最初设计用于军用航空电子总线，也慢慢在军用和民用航天器板级数据处理（OBDH）子系统中广泛应用。它实现了多冗余（通常为双冗余）平衡线路物理层，一个（差分）网络接口，时分复用，半双工命令/响应协议，且可以处理多达 30 个 RT（Remote Terminals）设备（注：1）。

MIL-STD-1553 在 1973 年第一次作为美国空军标准颁布，首次用于 F16 战隼战斗机。其它战斗机设计也很快跟进，包括 F-18 大黄蜂，AH-64 阿帕奇， P-3C 猎户座，F-15 鹰式战斗机和 F-20 虎鲨。它现在广泛用于美军和 NASA 的所有分支。在美国之外，它被 NATO 采用为 STANAG 3838 AVS。STANAG 3838， 以 UK MoD Def-Stan 00-18 Part 2 的形式用于狂风战斗机，鹰式教练机；并且扩展的与 STANAG3910-"EFABus"，用于欧洲台风战斗机。萨博 JAS39 使用 MIL-STD-1553B。俄罗斯制造 MiG-35 也使用 MIL-STD-1553。

MIL-STD-1553B 在 1978 年颁布，用于代替 1975 年的 MIL-STD-1553A 标准。1553A 和 1553B 的基本差异在于后者中 option 字段被定义了而不是留给用户依据需求来定义。如果标准没有定义一个事务，则在它的使用上没有任何协调配合，硬件和软件需要重新设计以适应新的应用。1553B 的主要目的是提供提供灵活性而无需为新用户创建新的设计。通过明确指定电气接口，这样不同制造商的设计间的电气接口可

以兼容。

1553B 总线使用一对 70~85 欧姆@1MHz 阻抗线组成。总线使用圆形连接器，它的中央管脚用于曼彻斯特双相位信号的高信号（正信号）。发送器和接收器通过隔离变压器耦合到总线，且实用一对隔离电阻或是耦合变压器与桩连接。传输位速率 1Mbit/s。

一个 MIL-STD-1553 多重数据总线系统通常由一个 BC（总线控制器）和多个 RT（远程终端）组成。BC 通过一路数据总线控制 RT，此数据总线为 BC 和所有连接的 RT 提供了一个单独的数据通道。总线上也可能存在一个或多个 BM（总线监视器），但是，BM 不能参与数据传输，只能采集或记录数据用于分析。在冗余总线实现中，数条数据总线用于提供多于一条数据路径，例如，双数据冗余总线。总线上的所有传输对于 BC 和 RT 是可访问的。消息由一个或多个 16 位字组成（命令，数据或是状态）。16 位字使用曼彻斯特编码传输，每位传输为 0.5us 高和 0.5us 低（逻辑 1）或是低-高顺序（逻辑 0）。每个字之前由一个 3us 的同步脉冲先导（1.5us 低加上 1.5us 高对应数据字，或是相反的顺序对应命令或状态字，这些先导不会出现在曼彻斯特编码中），字之后跟随一个奇校验位。实际上，每个字可以被当作一个位字：3 位同步，16 位载荷和 1 位奇校验控制。消息内的字连续传输，消息间有 4us 的间隙。但是，在一些老的 BC 上消息间的间隙可以且通常会大于 4us，甚至到 1ms。设备需要在 4-12us 内发送一个有效命令的响应消息，如果没有在 14us 内发送，则认为没有收到命令或是消息。

总线上的通信受 BC 控制，BC 使用 BC 到 RT 的命令字接收或者传输。

在一次传输中，所有的通讯由 BC 启动，一个 RT 设备不能自己启动数据传输。在 RT 到 RT 传输中，次序如下：RT1 后的子系统中的应用或是功能写数据（待传输到一个特定子地址）。写数据的时间不需要与传输的时间相关，但是接口需要确保只有部分更新的数据不会被传输。BC 命令 RT2（数据目的地）在特定数据子地址接收数据，然后命令 RT1 从命令中指定的传输子地址开始传输。RT1 传输一个状态字，指示它当前的状态和数据。BC 接收 RT1 的状态字，且查看传输命令是否已被无误接收且执行。RT2 在共享数据总线上接收数据，将其写入指定的接收子地址，并且发送其状态字。其后，RT2 后的子系统应用或功能可以访问数据，同样，读此数据的时序不需要与传输关联。BC 接收 RT2 的状态字，且查看接收命令和数据是否被无误接收且执行。

虽然如此，如果任一 RT 发送状态字或是数据失败或是通过在状态字中设定错误位指示错误，BC 可以重启传输。数个选项可用于这样的重传，包括立即重试（在冗余数据总线上）和（在传输序列中）后期重试（在同一总线上）。

序列确保 RT 是运行的且能够接收数据。状态字位于数据传输的最后确保数据被接收且数据传输的结果是可接受的。此序列化给予 MIL-STD-1553 高完整性。

1553B 标准不为任何特定传输指定任何特定时序，它依赖于系统

设计。通常情况下（大多数军用飞行器都这样做），BC 具有一个覆盖大多数传输的时间表，称为主周期（注 2）（主周期通常会被分成多个子周期）。这样一个周期执行调度结构，每个子周期（注 3）的传输以高速率产生，通常为 50 Hz，存在于其他子周期中的传输，如有两个组（假设：group2.1 和 group2.2），以次高速率发生，如 25 Hz。同样的，如果有四个组（3.1， 3.2， 3.3， 和 3.4），为 12.5 Hz。因此，如果使用了此调度结构，则传输都在谐波相关的频率，如：50，25， 12.5， 6.25， 3.125 和 1.5625（对于一个由 32 个子周期组成的主帧（@50Hz）来说）。RT 不能直接自己启动传输，标准包含一个方法用于当 RT 需要传输没有被 BC 自动调度的数据的情况。这些传输通常被称为非周期（循环）传输，这是因为它们存在于周期执行架构之外。如其次第，RT 在它状态字中的"Service Request"位请求传输。通常，这将导致 BC 发送一个向量模式字传输命令。但是在 RT 只有一个可能的发送周期的情况下，BC 可以跳过此部分。向量字被 RT 作为一个单独的 16 位字发送。此向量字的格式没有在标准中定义，所以系统设计必须指定来自 RT 的什么值意味着 BC 要做什么反应。这可能会立即调度一个非周期发送或者在当前子周期结束之后调度。这意味着 BC 需要轮询所有连接到数据总线上的 RT，通常在至少一个 major 周期。具有高优先级功能的 RT（例如那些操作飞行器控制界面的部分）会被轮询的更频繁。低优先级的功能被轮询的频率更低。

1553B 协议实现了 10 种类型的传输：

- BC 到 RT 传输（注 4）

- RT 到 BC 传输（注 5）
- RT 到 RT 传输（注 6）
- 无数据字的模式命令（注 7）
- 有数据字的发送模式命令（注 8）
- 有数据字的接收模式命令（注 9）
- BC 到 RT 广播（注 10）
- RT 到 RT 广播（注 11）
- 无数据字的模式命令广播（注 12）
- 有数据字的模式命令广播（注 13）

西安方解石信息技术有限责任公司自研生产的 1553B 总线协议卡系列至多可支持单卡 4 路总线（每路支持双冗余配置），可同时模拟 BC，RT 和 BM 功能，功能特性详见第二部分。板卡见下图，：

同时提供 Windows7 32 位操作系统驱动及 SDK 二次开发包（包含

开发所需的头文件和库文件，支持 VC，QT 等 C/C++平台开发），同时安装包附带 util 程序及相关源代码可以用于用户前期测试及开发参考。

如您需要其它系统平台的驱动及 SDK 支持，或者定制化需求，请联系西安方解石信息技术有限责任公司进行询问以获取支持， 联系电话：029-81151932。

2. 功能特性

- 33MHz/32Bit PCI 接口；
- 1~4 通道双冗余 1553B 仿真测试板卡；
- 遵循 MIL-STD-1553B /GJB289A-97 协议规范；
- MIL-STD-1553B 通讯速率： 1Mbps；
- 每通道为双冗余的 A、 B 通道；
- 每通道 1 个 BC,0~31 个 RT,1 个 MT，三种工作模式可同时工作；
- BC 支持重试；
- 支持单条消息的定时发送，分辨率 1ms；
- 支持时标模式；
- RT 方式下可设置非法命令表；
- MT 方式下支持过滤功能；
- 大容量的数据存储： 32M x 16bit。

3. 详细描述

a) BC 模式（总线控制器）

- 自动 BC 重试， 重试通道可选，重试次数可设 1~2 次；
- 消息间隔时间可设，分辨率 1us；
- 支持单条消息的定时发送，分辨率 1ms；
- 在发送过程中，可随机暂停或启动指定的消息；
- 在发送过程中，可动态更新 BC→RT 类型消息的数据区；
- 采用链表的方式来管理消息，可以动态插入消息；
- 根据要求，可设置硬件上自动过滤接收指定类型的消息。

b) RT 模式（远程终端）

- 0~31 个 RT；
- 非法命令表功能；
- 单缓冲和循环冲缓数据发送方式，每个子 RT 的数据缓存可到 8192 ×16bit；
- 循环缓冲数据接收方式，缓冲区大小： 128K ×16bit；
- 可以缓存 2K 条最新接收到的发送数据的消息；
- 可以缓存 2K 条最新接收到的接收数据的消息；
- 运行中可随时读取 RT 下的某个子 RT 接收到最新数据。

c) MT 模式（总线监视器）

- 命令字过滤功能；
- 采用循环缓冲接收数据；
- 8M × 16bit 的数据区；

可缓存 128K 条最近监控到的消息。

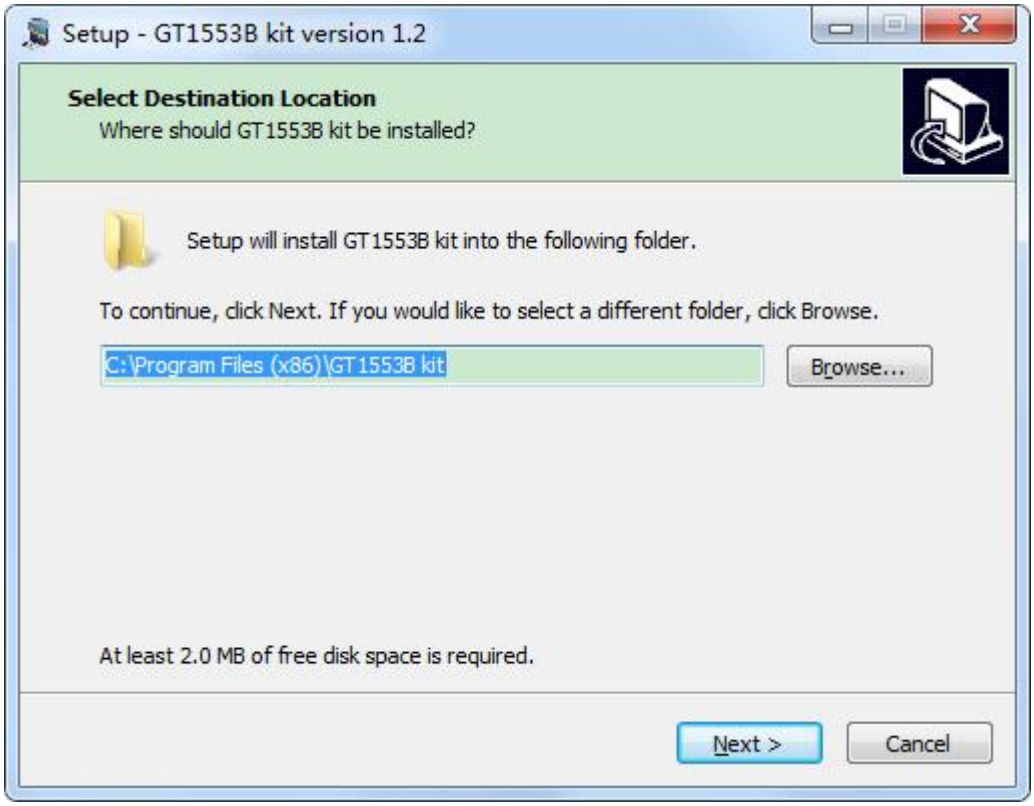
4. 驱动程序

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）：

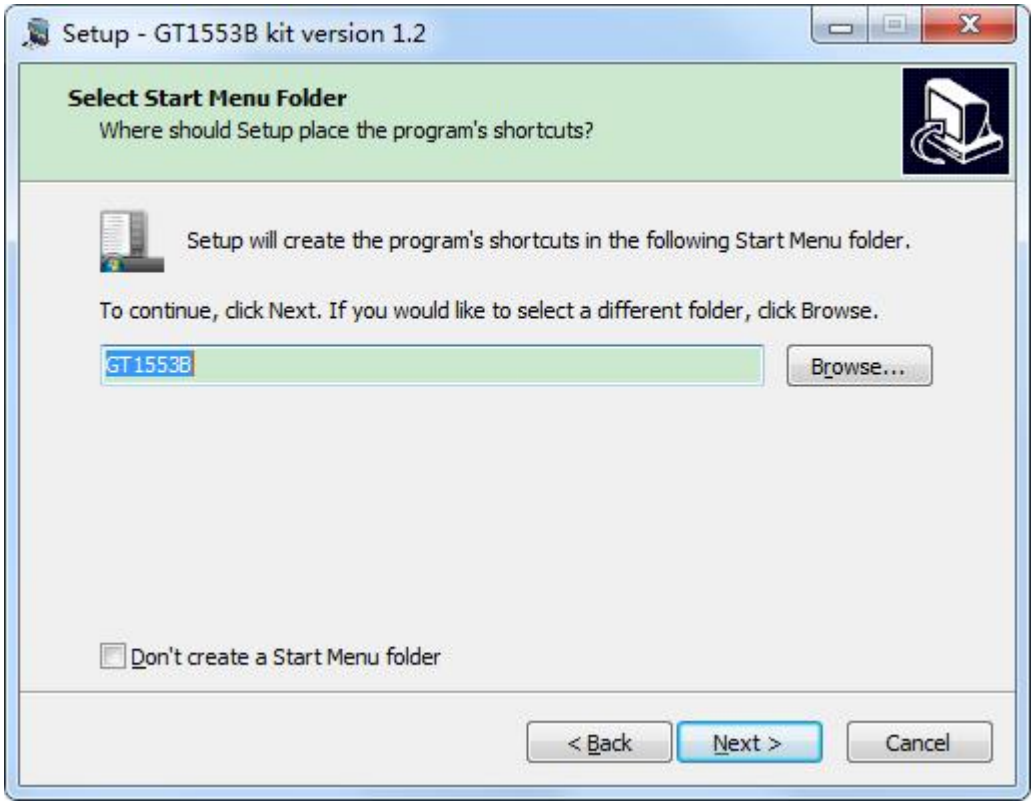
GT1553B kit_1.2.exe

驱动安装步骤如下：

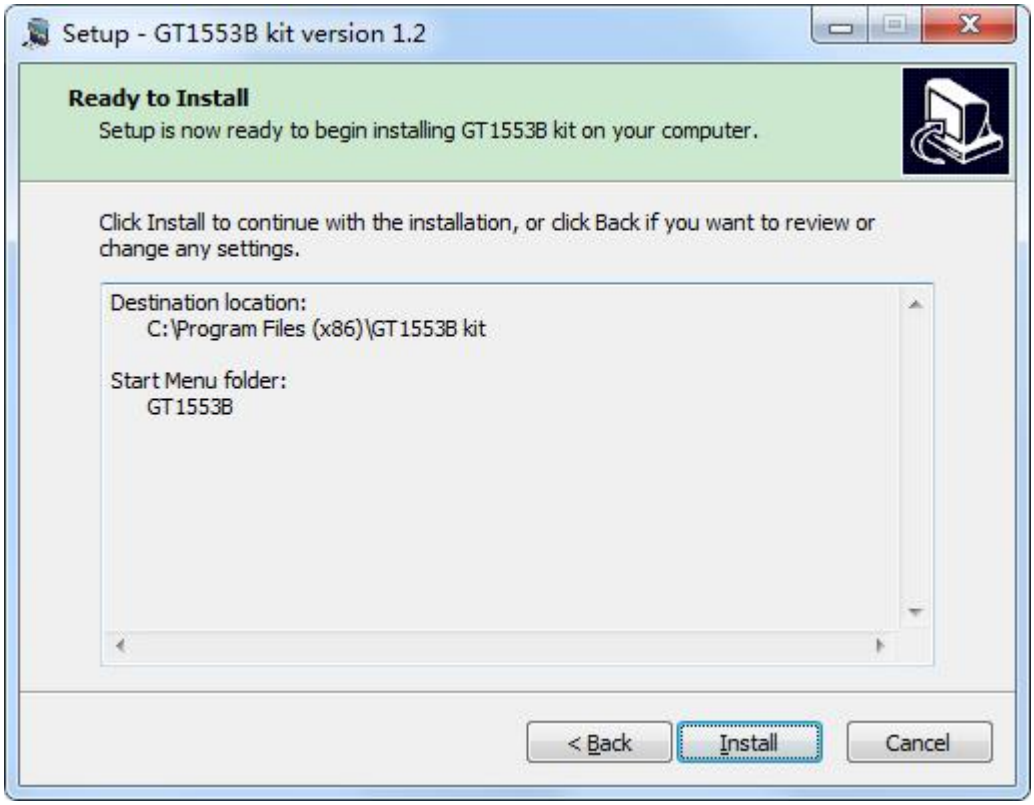
1、双击安装程序。



2、点击“Next”



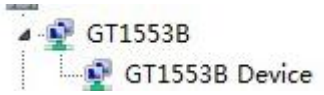
3、点击“Next”



4、点击“Install”安装驱动程序。安装过程中若出现以下显示：

选择“始终安装此驱动程序软件”，等待安装程序完成安装。

5、安装驱动，完成后可以在设备管理器中看到如下图所示。



6、整体驱动安装完成，设备可以正常使用。

5. 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。

6. 宏定义

6.1. RT 命令字数目定义

```
#define CMD_NUM                 (32)
#define BC_MSG_MAX_WORD_COUNT           (33)
```

7. 函数返回值

Api 函数返回值统一使用 GT1553B_STATUS 枚举值定义。
返回值定义如下：
GT1553B_SUCCESS
成功
GT1553B_OS_ERROR
系统错误
GT1553B_LOW_MEMORY
内存不足
GT1553B_INVALID_MSG_ID
无效的消息 ID
GT1553B_BAD_PARAMETER_1
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_BAD_PARAMETER_4
GT1553B_BAD_PARAMETER_5
GT1553B_BAD_PARAMETER_6
GT1553B_BAD_PARAMETER_7
GT1553B_BAD_PARAMETER_8
GT1553B_BAD_PARAMETER_9
无效的参数 1...9
GT1553B_NULL_DESCRIPTOR
空描述符
GT1553B_NO_BOARD
板卡打开失败
GT1553B_NOT_OPEN
板卡未打开

8. 结构体

8.1. BC 控制消息结构

```
typedef struct bc_msg_frame_st {
    UINT8 retry_enable;
    UINT8 channel_select;
    UINT8 run;
    UINT8 msg_format;
    UINT32 inter_msg_gap_time;
    UINT32 period;
    UINT16 msg_block[BC_MSG_MAX_WORD_COUNT];
    UINT16 reserved00;
    UINT16 cond_count_value;
    UINT16 cond_branch_msg_id;
    UINT16 cond_data_location;
    UINT16 cond_data_mask;
    UINT16 cond_data_expectation;
    UINT16 reserved01;
}ST_BC_MSG_FRAME, *ST_BC_MSG_FRAME_PTR;
```

成员定义

retry_enable：消息是否重试。0-不重试，1-重试。

channel_select：通道选择。0-消息经由 A 通道发出， 1-消息经由 B 通道发出。

run：消息执行位。0-消息不被执行， 1-消息被执行。

msg_format：取值 EN_MSG_FORMAT，参见 GT1553B_BCSetRcvMsgType 函数。

inter_msg_gap_time：消息间隔时间，单位 us。

period：消息发送周期，单位 ms。

msg_block：消息内容。

reserved00：保留。

cond_count_value：保留。

cond_branch_msg_id：保留。

cond_data_location：保留。

cond_data_mask：保留。

cond_data_expectation：保留。

reserved01：保留。

8.2. BC 接收消息结构

8.2.1. ST_BC_RCV_MSG_FRAME 结构体

```
//msg frame 定义， BC 接收帧
typedef struct bc_rcv_msg_frame_st {
    struct {
        struct {
            GT1553B_UINT32 tag;
            GT1553B_UINT16 count;
            union {
                GT1553B_UINT16 u16_value;
                struct {
                    unsigned short cmd1_error : 1;
                    unsigned short cmd2_error : 1;
                    unsigned short sync_error : 1;
                    unsigned short word_error : 1;
                    unsigned short reserved0 : 4;
                    unsigned short msg_format : 4;
                    unsigned short channel : 1;
                    unsigned short reserved1 : 3;
                }bw;
            }st;
        }bc;
        struct {
            GT1553B_UINT32 tag;
            GT1553B_UINT16 count;
            union {
                GT1553B_UINT16 u16_value;
                struct {
                    unsigned short reserved0 : 16;
                }bw;
            }st;
        }rt1;
        struct {
            GT1553B_UINT32 tag;
            GT1553B_UINT16 count;
            union {
                GT1553B_UINT16 u16_value;
                struct {
                    unsigned short reserved0 : 16;
                }bw;
            }st;
```

```
        }rt2;
    }header;
    GT1553B_UINT16 buf[BC_RCV_MSG_MAX_WORD_COUNT];
}ST_BC_RCV_MSG_FRAME, *ST_BC_RCV_MSG_FRAME_PTR;
```

成员定义

header：消息头部。

header.bc：bc 头部。

header.bc.tag：bc 时标。

header.bc.count：bc 消息字数。

header.bc.st：bc 消息状态。

header.bc.st.bw.cmd1_error：命令 1 错误。

header.bc.st.bw.cmd2_error：命令 2 错误。

header.bc.st.bw.word_error：字错误。

header.bc.st.bw.reserved0：保留。

header.bc.st.bw.msg_format：消息格式。

header.bc.st.bw.channel：消息通道。

header.bc.st.bw.reserved1：保留。

header.rt1：rt1 头部。

header.rt1.tag：rt1 时标。

header.rt1.count：rt1 消息字数。

header.rt1.st：rt1 消息状态。

header.rt1.st.bw.reserved0：保留。

header.rt2：rt2 头部。

header.rt2.tag：rt2 时标。

header.rt2.count：rt2 消息字数。

header.rt2.st：rt2 消息状态。

header.rt2.st.bw.reserved0：保留。

buf：消息内容。

8.3. RT 非法命令表结构

8.3.1. ST_ILLEGAL_CMD 结构体

```
typedef struct rt_illegal_cmd {
    UINT32 rx;
    UINT32 tx;
}ST_ILLEGAL_CMD, *ST_ILLEGAL_CMD_PTR;
```

成员定义

rx：接收非法命令表。

tx：发送非法命令表。

rx 和 tx 的每一位代表一个非法命令字，定义见下表：

位	说明
31	1：数据量为 31 的命令字非法； 0：合法
30	1：数据量为 30 的命令字非法； 0：合法
29	1：数据量为 29 的命令字非法； 0：合法
28	1：数据量为 28 的命令字非法； 0：合法
27	1：数据量为 27 的命令字非法； 0：合法
26	1：数据量为 26 的命令字非法； 0：合法
25	1：数据量为 25 的命令字非法； 0：合法
24	1：数据量为 24 的命令字非法； 0：合法
23	1：数据量为 23 的命令字非法； 0：合法
22	1：数据量为 22 的命令字非法； 0：合法
21	1：数据量为 21 的命令字非法； 0：合法
20	1：数据量为 20 的命令字非法； 0：合法
19	1：数据量为 19 的命令字非法； 0：合法
18	1：数据量为 18 的命令字非法； 0：合法
17	1：数据量为 17 的命令字非法； 0：合法
16	1：数据量为 16 的命令字非法； 0：合法
15	1：数据量为 15 的命令字非法； 0：合法
14	1：数据量为 14 的命令字非法； 0：合法
13	1：数据量为 13 的命令字非法； 0：合法
12	1：数据量为 12 的命令字非法； 0：合法
11	1：数据量为 11 的命令字非法； 0：合法
10	1：数据量为 10 的命令字非法； 0：合法
9	1：数据量为 9 的命令字非法； 0：合法
8	1：数据量为 8 的命令字非法； 0：合法
7	1：数据量为 7 的命令字非法； 0：合法
6	1：数据量为 6 的命令字非法； 0：合法
5	1：数据量为 5 的命令字非法； 0：合法
4	1：数据量为 4 的命令字非法； 0：合法
3	1：数据量为 3 的命令字非法； 0：合法
2	1：数据量为 2 的命令字非法； 0：合法
1	1：数据量为 1 的命令字非法； 0：合法
0	1：数据量为 32 的命令字非法； 0：合法

8.3.2. ST_ILLEGAL_CMD_RT 结构体

```
typedef struct illegal_cmd_rt {
    ST_ILLEGAL_CMD cmd[CMD_NUM];
}ST_ILLEGAL_CMD_RT, *ST_ILLEGAL_CMD_RT_PTR;
```

成员定义

md：所有命令字的非法命令表。

9. API 说明

9.1. 公共接口

9.1.1. GT1553B_OpenCard

原型

```
STDGT1553BCALL GT1553B_OpenCard(GT1553BHANDLE  * ph, GT1553B_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 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_LOW_MEMORY

GT1553B_NOT_OPEN

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553BHANDLE ph = NULL;
    status = GT1553B_OpenCard(&ph, 1);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_OpenCard failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.2. GT1553B_CloseCard

原型

```
STDGT1553BCALL GT1553B_CloseCard(GT1553BHANDLE * 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 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_CloseCard(ph);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_CloseCard failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.3. GT1553B_ResetCard

原型

```
STDGT1553BCALL GT1553B_ResetCard(GT1553BHANDLE 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 GT1553B_OpenCard 得到。

输出参数

无。

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

函数也可能返回其它错误码

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_ResetCard(ph);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_ResetCard failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.4. GT1553B_SetTimeTagEnable

原型

```
STDGT1553BCALL GT1553B_SetTimeTagEnable(GT1553BHANDLE ph, GT1553B_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 GT1553B_OpenCard 得到。
enable：GT1553B_TRUE -使能时标，GT1553B_FALSE-不使能时标

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_1
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_SetTimeTagEnable(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_SetTimeTagEnable failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.5. GT1553B_GetTimeTagEnable

原型

```
STDGT1553BCALL GT1553B_GetTimeTagEnable(GT1553BHANDLE ph, GT1553B_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 GT1553B_OpenCard 得到。

输出参数

enable：GT1553B_TRUE -当前时标使能，GT1553B_FALSE-当前时标不使能。

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
    status = GT1553B_GetTimeTagEnable(ph, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_GetTimeTagEnable failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.6. GT1553B_SetTimeTagValue

原型

```
STDGT1553BCALL GT1553B_SetTimeTagValue(GT1553BHANDLE ph, GT1553B_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 GT1553B_OpenCard 得到。

value：时标值。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_SetTimeTagValue(ph, 40);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_SetTimeTagValue failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.7. GT1553B_GetTimeTagValue

原型

```
STDGT1553BCALL GT1553B_GetTimeTagValue(GT1553BHANDLE ph, GT1553B_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 GT1553B_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 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 time_us;
    status = GT1553B_GetTimeTagValue(ph, &time_us);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_GetTimeTagValue failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.8. GT1553B_SetResponseTimeout

原型

```
STDGT1553BCALL  GT1553B_SetResponseTimeout(GT1553BHANDLE  ph,  GT1553B_UINT16 timeout)
```

功能

设置响应超时值，单位 0.5us。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
timeout：超时值，单位 0.5us。如：timeout = 20，则设置的超时时间为 10us。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_1
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_SetResponseTimeout(ph, 40);          //设置响应超时为 20us
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_SetResponseTimeout failed. Return value = %d\r\n", status);
        return status;
    }
```

9.1.9. GT1553B_GetResponseTimeout

原型

```
STDGT1553BCALL  GT1553B_GetResponseTimeout(GT1553BHANDLE  ph,  GT1553B_UINT16 *timeout)
```

功能

获取响应超时值，单位为 0.5us。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

timeout：获取到的当前响应超时值，如果获取到的值为 20，则当前响应超时时间为 10us。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 timeout;
    status = GT1553B_GetResponseTimeout(ph, &timeout);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_GetResponseTimeout failed. Return value = %d\r\n", status);
        return status;
    }
```

9.2. BC 操作接口

9.2.1. GT1553B_BCInit

原型

```
STDGT1553BCALL GT1553B_BCInit(GT1553BHANDLE ph)
```

功能

Bus ctroller 初始化。

初始化操作包括：

Bus ctroller 停止

BC 发送消息数目设置为 0

BC 设置为接收所有类型的消息

BC 重试次数设置为 1

BC 两次重试均设置为不改变通道

BC 设置为不在消息出错时重试

BC 设置为不在状态出错时重试

BC 设置为不在消息出错时停止发送

BC 设置为不在状态出错时停止发送

BC 清接收缓冲区

BC 使能

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

无

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

函数也可能返回其它错误码。

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCInit(ph);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCInit failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.2. GT1553B_BCSetEnable

原型

```
STDGT1553BCALL GT1553B_BCSetEnable(GT1553BHANDLE ph, GT1553B_BOOL enable)
```

功能

设置 BC 使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

enable：GT1553B_TRUE - 使能 BC， GT1553B_FALSE - 不使能 BC。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetEnable(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetEnable failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.3. GT1553B_BCGetEnable

原型

```
STDGT1553BCALL GT1553B_BCGetEnable(GT1553BHANDLE ph, GT1553B_BOOL *enable)
```

功能

获取 BC 当前是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

enable：GT1553B_TRUE - 当前 BC 使能， GT1553B_FALSE - 当前 BC 不使能。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
    status = GT1553B_BCGetEnable(ph, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetEnable failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.4. GT1553B_BCSetRetryNum

原型

```
STDGT1553BCALL GT1553B_BCSetRetryNum(GT1553BHANDLE ph, GT1553B_UINT8 num)
```

功能

设置 BC 重试次数。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
num：0 - 重试 1 次， 1 - 重试 2 次。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_1
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT8 num = 0;
    status = GT1553B_BCSetRetryNum(ph, num);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetRetryNum failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.5. GT1553B_BCGetRetryNum

原型

```
STDGT1553BCALL GT1553B_BCGetRetryNum(GT1553BHANDLE ph, GT1553B_UINT8 *num)
```

功能

获取 BC 当前重试次数。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

num：0 - 重试 1 次， 1 - 重试 2 次。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT8 num;
    status = GT1553B_BCGetRetryNum(ph, &num);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetRetryNum failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.6. GT1553B_BCSetRetryOnMsgError

原型

```
STDGT1553BCALL GT1553B_BCSetRetryOnMsgError(GT1553BHANDLE ph, GT1553B_BOOL bme)
```

功能

设置 BC 在消息错误时是否重试。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

bme：GT1553B_TRUE - 重试， GT1553B_FALSE - 不重试。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetRetryOnMsgError(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
```

```
        debug_printf("GT1553B_BCSetRetryOnMsgError failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.7. GT1553B_BCGetRetryOnMsgError

原型

```
STDGT1553BCALL GT1553B_BCGetRetryOnMsgError(GT1553BHANDLE ph, GT1553B_BOOL *bme)
```

功能

获取 BC 当前是否在消息出错时重试。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

bme：GT1553B_TRUE - 重试， GT1553B_FALSE - 不重试。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL bme;
    status = GT1553B_BCGetRetryOnMsgError(ph, &bme);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetRetryOnMsgError failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.8. GT1553B_BCSetRetryOnStatusError

原型

```
STDGT1553BCALL GT1553B_BCSetRetryOnStatusError(GT1553BHANDLE ph, GT1553B_BOOL bme)
```

功能

设置 BC 在 RT 状态字错误时是否重试。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

bme：GT1553B_TRUE - 重试， GT1553B_FALSE - 不重试。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetRetryOnStatusError(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetRetryOnStatusError failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.9. GT1553B_BCGetRetryOnStatusError

原型

```
STDGT1553BCALL GT1553B_BCGetRetryOnStatusError(GT1553BHANDLE ph, GT1553B_BOOL *bme)
```

功能

获取 BC 当前是否在 RT 状态字出错时重试。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

bme：GT1553B_TRUE - 重试， GT1553B_FALSE - 不重试。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL bme;
    status = GT1553B_BCGetRetryOnStatusError(ph, &bme);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetRetryOnStatusError failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.10. GT1553B_BCSetRetryChannelAlter

原型

```
STDGT1553BCALL GT1553B_BCSetRetryChannelAlter(GT1553BHANDLE ph, GT1553B_BOOL isFirstAlter, GT1553B_BOOL isSecondAlter)
```

功能

设置 BC 重试时是否改变通道。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
isFirstAlter：GT1553B_TRUE - 第一次重试改变通道， GT1553B_FALSE - 第一次重试不改变通道。
isSecondAlter：GT1553B_TRUE - 第二次重试改变通道， GT1553B_FALSE - 第二次重试不改变通道。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_1
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetRetryChannelAlter(ph, GT1553B_TRUE, GT1553B_FALSE);
    if (status != GT1553B_SUCCESS) {
```

```
        debug_printf("GT1553B_BCSetRetryChannelAlter failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.11. GT1553B_BCGetRetryChannelAlter

原型

```
STDGT1553BCALL GT1553B_BCGetRetryChannelAlter(GT1553BHANDLE ph, GT1553B_BOOL * isFirstAlter, GT1553B_BOOL * isSecondAlter)
```

功能

获取 BC 重试时是否改变通道。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

isFirstAlter：GT1553B_TRUE - 第一次重试改变通道， GT1553B_FALSE - 第一次重试不改变通道。

isSecondAlter：GT1553B_TRUE - 第二次重试改变通道， GT1553B_FALSE - 第二次重试不改变通道。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_BAD_PARAMETER_3

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL isFirstAlter, isSecondAlter;
    status = GT1553B_BCGetRetryChannelAlter(ph, &isFirstAlter, &isSecondAlter);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetRetryChannelAlter failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.12. GT1553B_BCSetStopOnError

原型

```
STDGT1553BCALL GT1553B_BCSetStopOnError(GT1553BHANDLE ph, GT1553B_BOOL bse)
```

功能

设置 BC 在消息错误时是否停止。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

bse：GT1553B_TRUE - 停止， GT1553B_FALSE - 不停止。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetStopOnError(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetStopOnError failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.13. GT1553B_BCGetStopOnError

原型

```
STDGT1553BCALL GT1553B_BCGetStopOnError(GT1553BHANDLE ph, GT1553B_BOOL *bse)
```

功能

获取 BC 当前是否在消息出错时停止。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

bse：GT1553B_TRUE - 停止， GT1553B_FALSE - 不停止。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL bse;
    status = GT1553B_BCGetStopOnError(ph, &bse);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetStopOnError failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.14. GT1553B_BCSetStopOnStatus

原型

```
STDGT1553BCALL GT1553B_BCSetStopOnStatus(GT1553BHANDLE ph, GT1553B_BOOL bss)
```

功能

设置 BC 在 RT 状态字错误时是否停止。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

bss：GT1553B_TRUE - 停止， GT1553B_FALSE - 不停止。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetStopOnStatus(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetStopOnStatus failed, return status = 0x%08x!!!\r\n", status);
```

```
        return status;
    }
```

9.2.15. GT1553B_BCGetStopOnStatus

原型

```
STDGT1553BCALL GT1553B_BCGetStopOnStatus(GT1553BHANDLE ph, GT1553B_BOOL *bss)
```

功能

获取 BC 当前是否在 RT 状态字出错时停止。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

bss：GT1553B_TRUE - 停止， GT1553B_FALSE - 不停止。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL bss;
    status = GT1553B_BCGetStopOnStatus(ph, &bss);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetStopOnStatus failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.16. GT1553B_BCSetRcvMsgType

原型

```
STDGT1553BCALL GT1553B_BCSetRcvMsgType(GT1553BHANDLE ph, GT1553B_UINT32 type)
```

功能

设置 BC 接收数据类型。数据类型枚举定义如下：

```
typedef enum {
    EN_MSG_BC_2_RT,
```

```
    EN_MSG_BC_2_RT_BROADCAST,
    EN_MSG_MODE_BC_2_RT_WITH_DATA,
    EN_MSG_MODE_BC_2_RT_BROADCAST_WITH_DATA,
    EN_MSG_RT_2_RT,
    EN_MSG_RT_2_RT_BROADCAST,
    EN_MSG_MODE_BC_2_RT,
    EN_MSG_MODE_BC_2_RT_BROADCAST,
    EN_MSG_RT_2_BC,
    EN_MSG_MODE_RT_2_BC,
    EN_MSG_BOTTOM
}EN_MSG_FORMAT, * EN_MSG_FORMAT_PTR;
```

type 参数的每一位代表一种类型。Bit0 代表 EN_MSG_BC_2_RT，Bit1 代表 EN_MSG_BC_2_RT_BROADCAST，依次类推。位为 1 表示接收此类型的消息，位为 0 表示不接收此类型的消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
type：bit0~9 每位代表一个消息类型，为 1 表示接收此类型的消息，为 0 表示不接收此类型的消息。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_1
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetRcvMsgType(ph, 0x7); // 接收 EN_MSG_BC_2_RT 、 EN_MSG_BC_2_RT_BROADCAST 、 EN_MSG_MODE_BC_2_RT_WITH_DATA 类型的消息，不接收其它类型的消息
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetRcvMsgType failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.17. GT1553B_BCGetRcvMsgType

原型

```
STDGT1553BCALL GT1553B_BCGetRcvMsgType(GT1553BHANDLE ph, GT1553B_UINT32 *type)
```

功能

获取 BC 当前允许接收的消息类型。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

type：bit0~9 每位代表一个消息类型，为 1 表示接收此类型的消息，为 0 表示不接收此类型的消息。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_1

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 type;
    status = GT1553B_BCGetRcvMsgType(ph, &type);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetRcvMsgType failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.18. GT1553B_BCWriteMsg

原型

```
STDGT1553BCALL GT1553B_BCWriteMsg(GT1553BHANDLE ph, GT1553B_UINT16 id,
ST_BC_MSG_FRAME_PTR msg);
```

功能

配置 BC 消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：消息的编号，0-510。消息顺序执行，消息编号必须连续。如：BC 需要发送 3 条消息，则 3 条消息编号为 0，1，2。然后调用 GT1553B_SetSendMsgCount 设置待处理的消息个数为 3。
msg：消息结构。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_3
GT1553B_INVALID_MSG_ID
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_BC_MSG_FRAME msg;
    msg.retry_enable = 0;           //0 不重试， 1 重试
    msg.channel_select = 0;         //0 A 通道， 1 B 通道
    msg.run = 1;
    msg.msg_format = EN_MSG_BC_2_RT;      //EN_MSG_BC_2_RT
    msg.inter_msg_gap_time = 1000;        //ms
    msg.period = 0;
    msg.cond_count_value = 0xAAAA;        //reserved
    msg.cond_branch_msg_id = 0x5555;      //reserved
    msg.cond_data_location = 0x5A5A;      //reserved
    msg.cond_data_mask = 0xA5A5;          //reserved
    msg.cond_data_expectation = 0x7EE7;   //reserved
    msg.msg_block[0] = GT1553B_GENERATE_COMMAND_WORD(0, 0, 1, 0);
    memset(&msg.msg_block[1], 0xAA, sizeof(GT1553B_UINT16)* (i % 32 == 0 ? 32 : i % 32));
    status = GT1553B_BCWriteMsg(ph, 0, &msg);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCWriteMsg failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
    status = GT1553B_BCSetSendMsgCount(handle, 1);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetSendMsgCount failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.2.19. GT1553B_BCSetSendMsgCount

原型

```
STDGT1553BCALL  GT1553B_BCSetSendMsgCount(GT1553BHANDLE  ph,  GT1553B_UINT32
count);
```

功能

设置 BC 发送消息个数。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
count：消息个数。

输出参数

无

返回值

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

见 GT1553B_BCWriteMsg。

9.2.20. GT1553B_BCGetSendMsgCount

原型

```
STDGT1553BCALL  GT1553B_BCGetSendMsgCount(GT1553BHANDLE  ph,  GT1553B_UINT32
*count)
```

功能

获取 BC 发送表中的消息个数。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

count：消息个数。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 count;
    status = GT1553B_BCGetSendMsgCount(ph, &count);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetSendMsgCount failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.21. GT1553B_BCSetMsgRunFlag

原型

```
STDGT1553BCALL GT1553B_BCSetMsgRunFlag(GT1553BHANDLE ph, GT1553B_UINT16 id, GT1553B_BOOL run)
```

功能

BC 消息运行控制函数。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
run：消息运行控制位。
GT1553B_TRUE：当 BC 运行时，使得对应消息执行。对于事件消息（消息周期为 0）会发送一次。对于周期消息（消息周期不为 0），会导致消息（不间断）周期运行。
GT1553B_FALSE：当 BC 运行时，使得对应消息不被执行。

输出参数

无。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_INVALID_MSG_ID
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCSetMsgRunFlag(ph, 0, GT1553B_TRUE);          //设置 id 为 0 的消息被执行
    if (status != GT1553B_SUCCESS) {
```

```
        debug_printf("GT1553B_BCSetMsgRunFlag failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.22. GT1553B_BCGetMsgInfo

原型

```
STDGT1553BCALL GT1553B_BCGetMsgInfo(GT1553BHANDLE ph, GT1553B_UINT16 id, ST_BC_MSG_FRAME_PTR msg)
```

功能

获取 BC 消息表中的消息配置信息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：消息 id。

输出参数

msg：获取的消息。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_3
GT1553B_INVALID_MSG_ID
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_BC_MSG_FRAME msg;
    status = GT1553B_BCGetMsgInfo(ph, 0, &msg);      //获取 id 为 0 的消息信息
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetMsgInfo failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.23. GT1553B_BCSetMsgDataBlock

原型

```
STDGT1553BCALL GT1553B_BCSetMsgDataBlock(GT1553BHANDLE ph, GT1553B_UINT16 id,
```

```
GT1553B_UINT8 cnt, GT1553B_UINT16 * buf)
```

功能

修改 BC 消息表中消息的数据字。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：消息 ID。
cnt：要写入的数据字的数量。以字为单位。
buf：存放待写入的数据。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_INVALID_MSG_ID
GT1553B_INVALID_DATA_COUNT
GT1553B_BAD_PARAMETER_4
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 db[3] = {0xAAAA, 0x5555, 0x5A5A};
    status = GT1553B_BCSetMsgDataBlock(ph, 0, 3, db);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCSetMsgDataBlock failed, return status = 0x%08x!!!\r\n", status);
        return status;
    }
```

9.2.24. GT1553B_BCStart

原型

```
STDGT1553BCALL GT1553B_BCStart(GT1553BHANDLE ph)
```

功能

启动 BC，BC 开始处理消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表:

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
GT1553B_STATUS status = GT1553B_SUCCESS;
status = GT1553B_BCStart(ph);
if (status != GT1553B_SUCCESS) {
    debug_printf("GT1553B_BCStart failed. Return value = 0x%08x\r\n", status);
    return status;
}
```

9.2.25. GT1553B_BCStop

原型

```
STDGT1553BCALL GT1553B_BCStop(GT1553BHANDLE ph)
```

功能

停止 BC， 执行此函数，让 BC 由运行状态进入停止状态，立即停止处理所有消息。如执行此函数时，BC 正在处理某条消息，则等待处理完当前的消息，然后停止。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表:

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_BCStop(ph);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCStop failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.2.26. GT1553B_BCClearMsgBuffer

原型

```
STDGT1553BCALL GT1553B_BCClearMsgBuffer(GT1553BHANDLE ph)
```

功能

清空 BC 硬件接收消息缓冲区。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
GT1553B_STATUS status = GT1553B_SUCCESS;
status = GT1553B_BCClearMsgBuffer(ph);
if (status != GT1553B_SUCCESS) {
    debug_printf("GT1553B_BCClearMsgBuffer failed, return status = 0x%08x!!!\r\n", status);
    return status;
}
```

9.2.27. GT1553B_BCGetMsgNumNewly

原型

```
STDGT1553BCALL GT1553B_BCGetMsgNumNewly(GT1553BHANDLE ph, GT1553B_UINT32 * num)
```

功能

查询 BC 新消息数目（即用户还未来得及接收的新消息）。这里的新消息是函数 GT1553B_BCSetRcvMsgType 中指定接收类型的消息，GT1553B_BCSetRcvMsgType 中指定不接收的消息类型不在这里统计。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

num：BC 新消息数目。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
GT1553B_STATUS status = GT1553B_SUCCESS;
GT1553B_UINT32 num;
status = GT1553B_BCGetMsgNumNewly(ph, &num);
if (status != GT1553B_SUCCESS) {
    debug_printf("GT1553B_BCGetMsgNumNewly failed. Return value = 0x%08x\r\n", status);
    return status;
}
```

9.2.28. GT1553B_BCGetTotalMsgCount

原型

```
STDGT1553BCALL GT1553B_BCGetTotalMsgCount(GT1553BHANDLE ph, GT1553B_UINT32 *cnt)
```

功能

查询 BC 从板卡复位到目前处理的总的消息数（包括用户还未来得及接收的新消息，且不区分消息的类型）。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

count：BC 消息总数目。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
GT1553B_STATUS status = GT1553B_SUCCESS;
```

```
GT1553B_STATUS count;
status = GT1553B_BCGetTotalMsgCount(ph, &count);
if (status != GT1553B_SUCCESS) {
    debug_printf("GT1553B_BCGetTotalMsgCount failed. Return value = 0x%08x\r\n", status);
    return status;
}
```

9.2.29. GT1553B_BCReadMsg

原型

```
STDGT1553BCALL GT1553B_BCReadMsg(GT1553BHANDLE ph, ST_BC_RCV_MSG_FRAME_PTR *msg)
```

功能

用户按 BC 处理消息先后顺序读取消息。调用此函数前推荐调用 `GT1553B_BCGetMsgNumNewly` 函数来确保存在新消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

msg：BC 接收消息结构，存放读取到的消息。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_BC_RCV_MSG_FRAME msg;
    status = GT1553B_BCReadMsg(ph, &msg);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCReadMsg failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.2.30. GT1553B_BCReadLastMsg

原型

```
STDGT1553BCALL GT1553B_BCReadLastMsg(GT1553BHANDLE ph, ST_BC_RCV_MSG_FRAME_PTR
* msg)
```

功能

用户按 BC 最近处理的消息。调用此函数前推荐调用 `GT1553B_BCGetMsgNumNewly` 函数来确保存在新消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

msg：BC 接收消息结构，存放读取到的消息。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_BC_RCV_MSG_FRAME msg;
    status = GT1553B_BCReadLastMsg(ph, &msg);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCReadLastMsg failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.2.31. GT1553B_BCGetRunningState

原型

```
STDGT1553BCALL GT1553B_BCGetRunningState(GT1553BHANDLE ph, GT1553B_BOOL *
state)
```

功能

获取 BC 运行状态。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

state：BC 运行状态， GT1553B_TRUE - BC 正在运行， GT_1553B_FALSE - BC 未在运行。

返回值:
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL state;
    status = GT1553B_BCGetRunningState(ph, &state);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_BCGetRunningState failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3. RT 操作接口

9.3.1. GT1553B_RTInit

原型

```
STDGT1553BCALL GT1553B_RTInit(GT1553BHANDLE ph, GT1553B_UINT8 id)
```

功能
初始化指定 ID 的 RT。
RT 初始化执行下述操作：
禁止 RT
设置发送缓冲区为单缓冲模式
发送缓冲区设置为全 0
清接收缓冲区
清零 status
bitword 与 vectorword 设置为 0
使能 RT

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID， 0~30。

输出参数

无。

返回值:
成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR
函数还可能返回其它错误码。

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_RTInit(ph, (GT1553B_UINT8)0);        //初始化 ID 为 0 的 RT
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTInit failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.2. GT1553B_RTSetTxMode

原型

```
STDGT1553BCALL GT1553B_RTSetTxMode(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT32 mode)
```

功能

设置 RT 数据缓冲区模式。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID， 0~30。
mode：发送缓冲区模式，0 - 单缓冲模式， 1 - 循环缓冲模式

输出参数

无。

返回值：
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_RTSetTxMode(ph, (GT1553B_UINT8)0, 0);          //设置 ID 为 0 的 RT 的发送缓冲模式为单缓冲
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSetTxMode failed. Return value = 0x%08x\r\n", status);
        return status;
```

```
    }
```

9.3.3. GT1553B_RTGetTxMode

原型

```
STDGT1553BCALL GT1553B_RTGetTxMode(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT32 * mode)
```

功能

获取 RT 数据缓冲区模式。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID， 0~30。

输出参数

mode：发送缓冲区模式，0 - 单缓冲模式， 1 - 循环缓冲模式。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 mode;
    status = GT1553B_RTGetTxMode(ph, (GT1553B_UINT8)0, &mode);      //获取 ID 为 0 的 RT 的发送缓冲区模式
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetTxMode failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.4. GT1553B_RTSetCycleBufferThreshold

原型

```
STDGT1553BCALL GT1553B_RTSetCycleBufferThreshold(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT8 sub, GT1553B_UINT32 mode)
```

功能

设置 RT 数据循环缓冲区长度。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID， 0~30。
sub: RT 子地址， 1~30
threshold：循环缓冲区长度，取值范围 1 ~ 2048

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_BAD_PARAMETER_4
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_RTSetCycleBufferThreshold(ph, (GT1553B_UINT8)0, (GT1553B_UINT8)3, 32); //设置RT0，子地址3的循环缓冲区长度为32
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSetCycleBufferThreshold failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.5. GT1553B_RTGetCycleBufferThreshold

原型

```
STDGT1553BCALL GT1553B_RTGetCycleBufferThreshold(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT8 sub, GT1553B_UINT32 * threshold)
```

功能

获取 RT 数据缓冲区模式。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID， 0~30。
sub: RT 子地址， 1~30

输出参数

threshold：循环缓冲区长度，取值范围 1 ~ 2048

返回值：
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_BAD_PARAMETER_4
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 threshold;
    status = GT1553B_RTGetCycleBufferThreshold(ph, (GT1553B_UINT8)0, (GT1553B_UINT8)3, &threshold);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetCycleBufferThreshold failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.6. GT1553B_RTSetEnable

原型

```
STDGT1553BCALL GT1553B_RTSetEnable(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_BOOL enable)
```

功能
设置 RT 是否使能。
参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID， 0~30。
enable：GT1553B_TRUE - 使能 RT， GT1553B_FALSE - 禁止 RT。

输出参数

无。
返回值：
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_RTSetEnable(ph, (GT1553B_UINT8)0, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSetEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.7. GT1553B_RTGetEnable

原型

```
STDGT1553BCALL GT1553B_RTGetEnable(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_BOOL * enable)
```

功能

获取 RT 是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID， 0~30。

输出参数

enable：GT1553B_TRUE - 使能 RT， GT1553B_FALSE - 禁止 RT。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
    status = GT1553B_RTGetEnable(ph, (GT1553B_UINT8)0, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.8. GT1553B_SetRtClrTTagOnSync

原型

```
STDGT1553BCALL GT1553B_SetRtClrTTagOnSync(GT1553BHANDLE ph, GT1553B_BOOL enable)
```

功能

设置时标在接收到不带数据的同步方式指令后自动清零。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

enable：GT1553B_TRUE - 使能， GT1553B_FALSE - 不使能。

输出参数

无。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_SetRtClrTTagOnSync(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_SetRtClrTTagOnSync failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.9. GT1553B_GetRtClrTTagOnSync

原型

```
STDGT1553BCALL GT1553B_GetRtClrTTagOnSync(GT1553BHANDLE ph, GT1553B_BOOL *enable)
```

功能

获取是否使能时标在接收到不带数据的同步方式指令后自动清零。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

enable：GT1553B_TRUE - 使能， GT1553B_FALSE - 不使能。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
    status = GT1553B_GetRtClrTTagOnSync(ph, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_GetRtClrTTagOnSync failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.10. GT1553B_SetRtLoadTTagOnSync

原型

```
STDGT1553BCALL GT1553B_SetRtLoadTTagOnSync(GT1553BHANDLE ph, GT1553B_BOOL enable)
```

功能

设置时标在接收到带数据的同步指令后，将命令中的数据加载到时标中作为初始值。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

enable：GT1553B_TRUE - 使能， GT1553B_FALSE - 不使能。

输出参数

无。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_SetRtLoadTTagOnSync(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_SetRtLoadTTagOnSync failed. Return value = 0x%08x\r\n", status);
```

```
        return status;
    }
```

9.3.11. GT1553B_GetRtLoadTTagOnSync

原型

```
STDGT1553BCALL GT1553B_GetRtLoadTTagOnSync(GT1553BHANDLE ph, GT1553B_BOOL *enable)
```

功能

获取是否使能时标在接收到带数据的同步指令后，将命令中的数据加载到时标中作为初始值。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

enable：GT1553B_TRUE - 使能， GT1553B_FALSE - 不使能。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
    status = GT1553B_GetRtLoadTTagOnSync(ph, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_GetRtLoadTTagOnSync failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.12. GT1553B_RTSetStatus

原型

```
STDGT1553BCALL GT1553B_RTSetStatus(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_BOOL busy, GT1553B_BOOL dbusctl, GT1553B_BOOL servicereq, GT1553B_BOOL subsystem, GT1553B_BOOL terminal)
```

功能

RT 状态字置位。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。
busy：RT 状态字 busy 位是否置位。
dbusctl：RT 状态字 dbusctl 位是否置位。
servicereq：RT 状态字 servicereq 位是否置位。
subsystem：RT 状态字 subsystem 位是否置位。
terminal：RT 状态字 terminal 位是否置位。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_NULL_DESCRIPTOR
GT1553B_BAD_PARAMETER_2
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_RTSetStatus(ph, 0, GT1553B_TRUE,  GT1553B_TRUE, GT1553B_TRUE, GT1553B_TRUE,
GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSetStatus failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.13. GT1553B_RTGetStatus

原型

```
STDGT1553BCALL  GT1553B_RTGetStatus(GT1553BHANDLE  ph,  GT1553B_UINT8  id,
GT1553B_BOOL  *  busy,  GT1553B_BOOL  *  dbusctl,  GT1553B_BOOL  *  servicereq,
GT1553B_BOOL * subsystem, GT1553B_BOOL * terminal)
```

功能

获取 RT 的状态字。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

id：RT 的 ID，也即 RT 地址。

输出参数

busy：GT1553B_TRUE - 置位， GT1553B_FALSE - 不置位。
dbusctl：GT1553B_TRUE - 置位， GT1553B_FALSE - 不置位。
servicereq：GT1553B_TRUE - 置位， GT1553B_FALSE - 不置位。
subsystem：GT1553B_TRUE - 置位， GT1553B_FALSE - 不置位。
terminal：GT1553B_TRUE - 置位， GT1553B_FALSE - 不置位。

返回值:
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_BAD_PARAMETER_4
GT1553B_BAD_PARAMETER_5
GT1553B_BAD_PARAMETER_6
GT1553B_BAD_PARAMETER_7
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL busy;
    GT1553B_BOOL dbusctl;
    GT1553B_BOOL servicereq;
    GT1553B_BOOL subsystem;
    GT1553B_BOOL terminal;
    status = GT1553B_RTGetStatus(ph, id, &busy, &dbusctl, &servicereq, &subsystem, &terminal);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetStatus failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.14. GT1553B_SetRtIllegalCmdEnable

原型

```
STDGT1553BCALL GT1553B_SetRtIllegalCmdEnable(GT1553BHANDLE ph, GT1553B_BOOL enable)
```

功能

使能 RT 非法命令表。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
enable：GT1553B_TRUE - 使能非法命令表， GT1553B_FALSE - 不使能非法命令表。

输出参数

无。

返回值：
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_SetRtIllegalCmdEnable(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_SetRtIllegalCmdEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.15. GT1553B_GetRtIllegalCmdEnable

原型

```
STDGT1553BCALL GT1553B_GetRtIllegalCmdEnable(GT1553BHANDLE ph, GT1553B_BOOL *
enable)
```

功能

获取是否使能 RT 非法命令表。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

enable：GT1553B_TRUE - 使能非法命令表， GT1553B_FALSE - 不使能非法命令表。

返回值：
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
```

```
    status = GT1553B_GetRtIllegalCmdEnable(ph, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_GetRtIllegalCmdEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.16. GT1553B_SetRtRcvIllegalDataEnable

原型

```
STDGT1553BCALL GT1553B_SetRtRcvIllegalDataEnable(GT1553BHANDLE ph, GT1553B_BOOL enable)
```

功能

设置是否使能 RT 接收非法指令数据。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

enable：GT1553B_TRUE - 使能接收非法指令数据， GT1553B_FALSE - 不使能非法指令数据。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_SetRtRcvIllegalDataEnable(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_SetRtRcvIllegalDataEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.17. GT1553B_GetRtRcvIllegalDataEnable

原型

```
STDGT1553BCALL GT1553B_GetRtRcvIllegalDataEnable(GT1553BHANDLE ph, GT1553B_BOOL * enable)
```

功能

获取是否使能 RT 接收非法指令数据。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

enable：GT1553B_TRUE - 使能接收非法指令数据， GT1553B_FALSE - 不使能非法指令数据。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
    status = GT1553B_GetRtRcvIllegalDataEnable(ph, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_GetRtRcvIllegalDataEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.18. GT1553B_RTSetIllegalCmdTable

原型

```
STDGT1553BCALL GT1553B_RTSetIllegalCmdTable(GT1553BHANDLE ph, GT1553B_UINT8 id,
ST_ILLEGAL_CMD_RT_PTR table)
```

功能

设置 RT 非法指令表。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

id：RT 的 ID，也即 RT 地址。

table：RT 非法命令表结构。

输出参数

无。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_ILLEGAL_CMD_RT table;
    Memset(&table, 0, sizeof(ST_ILLEGAL_CMD_RT));    //允许接收所有的非法命令数据
    status = GT1553B_RTSetIllegalCmdTable(ph, 0, &table);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSetIllegalCmdTable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.19. GT1553B_RTGetIllegalCmdTable

原型

```
STDGT1553BCALL GT1553B_RTGetIllegalCmdTable(GT1553BHANDLE ph, GT1553B_UINT8 id,
ST_ILLEGAL_CMD_RT_PTR table)
```

功能

获取 RT 非法指令表。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

table：RT 非法命令表结构。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_ILLEGAL_CMD_RT table;
    status = GT1553B_RTGetIllegalCmdTable(ph, 0, &table);
    if (status != GT1553B_SUCCESS) {
```

```
        debug_printf("GT1553B_RTGetIllegalCmdTable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.20. GT1553B_RTSetVectorWord

原型

```
STDGT1553BCALL GT1553B_RTSetVectorWord(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT16 word)
```

功能

设置 RT 向量字。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。
word：矢量字。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 vector = 0xAAAA;
    status = GT1553B_RTSetVectorWord(ph, 0, vector);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSetVectorWord failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.21. GT1553B_RTGetVectorWord

原型

```
STDGT1553BCALL GT1553B_RTGetVectorWord(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT16 * word)
```

功能

获取 RT 向量字。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

word：向量字。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 word;
    status = GT1553B_RTGetVectorWord(ph, 0, &word);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetVectorWord failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.22. GT1553B_RTSetBitWord

原型

```
STDGT1553BCALL GT1553B_RTSetBitWord(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT16 word)
```

功能

设置 RT 自检字。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。
word：自检字。

输出参数

无。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 word = 0x5555;
    status = GT1553B_RTSetBitWord(ph, 0, word);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSetBitWord failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.23. GT1553B_RTGetBitWord

原型

```
STDGT1553BCALL GT1553B_RTGetBitWord(GT1553BHANDLE ph, GT1553B_UINT8 id, GT1553B_UINT16 * word)
```

功能

获取 RT 自检字。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

id：RT 的 ID，也即 RT 地址。

输出参数

word：自检字。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_BAD_PARAMETER_3

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 word;
```

```
    status = GT1553B_RTGetBitWord(ph, 0, &word);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetBitWord failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.24. GT1553B_RTSendMsg

原型

```
STDGT1553BCALL GT1553B_RTSendMsg(GT1553BHANDLE ph, GT1553B_UINT8 id,
GT1553B_UINT8 addr, GT1553B_UINT16 len, GT1553B_UINT16 * msg)
```

功能

RT 发送数据。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。
addr：待发送数据的 RT 终端子地址。
len：待发送数据的长度，单缓冲模式下有效长度为 32， 循环缓冲为 8192。
msg：存放待发送的数据。

输出参数

无。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_BAD_PARAMETER_4
GT1553B_BAD_PARAMETER_5
GT1553B_LOW_MEMORY
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 msg[4] = {0xAAAA, 0x5555, 0xA5A5, 0x5A5A};       // 更新发送缓冲 4 个字的数据
    status = GT1553B_RTSendMsg(ph, 0, 0, 4, msg);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTSendMsg failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.25. GT1553B_RTGetSendMsg

原型

```
STDGT1553BCALL  GT1553B_RTGetSendMsg(GT1553BHANDLE  ph,  GT1553B_UINT8  id,
GT1553B_UINT8 addr, GT1553B_UINT16 len, GT1553B_UINT16 * msg)
```

功能

RT 发送数据。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。
addr：待发送数据的 RT 终端子地址。
len：待发送数据的长度，单缓冲模式下有效长度为 32， 循环缓冲为 8192。

输出参数

msg：存放读取的发送缓冲区中的数据。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_BAD_PARAMETER_4
GT1553B_BAD_PARAMETER_5
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT16 msg[4];
    status = GT1553B_RTGetSendMsg(ph, 0, 0, 4, msg);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetSendMsg failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.26. GT1553B_RTRxClearMsgBuffer

原型

```
STDGT1553BCALL GT1553B_RTRxClearMsgBuffer(GT1553BHANDLE ph, GT1553B_UINT8 id)
```

功能

清空硬件上 RT 接收 RX 消息缓存。在接收 RX 消息之前执行此操作，以避免读出硬件缓存之

前残留的消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

无。

返回值:
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_RTRxClearMsgBuffer(ph, 0);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTRxClearMsgBuffer failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.27. GT1553B_RTTxClearMsgBuffer

原型

```
STDGT1553BCALL GT1553B_RTTxClearMsgBuffer(GT1553BHANDLE ph, GT1553B_UINT8 id)
```

功能

清空硬件上 RT 接收 TX 消息缓存。在接收 TX 消息之前执行此操作，以避免读出硬件缓存之前残留的消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

无。

返回值:
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_RTTxClearMsgBuffer(ph, 0);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTTxClearMsgBuffer failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.28. GT1553B_RTGetRxNewMsgNum

原型

```
STDGT1553BCALL GT1553B_RTGetRxNewMsgNum(GT1553BHANDLE ph, GT1553B_UINT8 id,
GT1553B_UINT32 * num)
```

功能

查询 RT 新 RX 消息数目（即用户还未来得及处理的新消息）。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

num：存放新消息的数目。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 num;
    status = GT1553B_RTGetRxNewMsgNum(ph, 0, &num);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetRxNewMsgNum failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.29. GT1553B_RTGetTxNewMsgNum

原型

```
STDGT1553BCALL GT1553B_RTGetTxNewMsgNum(GT1553BHANDLE ph, GT1553B_UINT8 id,
GT1553B_UINT32 * num)
```

功能

查询 RT 新 TX 消息数目（即用户还未来得及处理的新消息）。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

num：存放新消息的数目。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 num;
    status = GT1553B_RTGetTxNewMsgNum(ph, 0, &num);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTGetTxNewMsgNum failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.30. GT1553B_RTReadRxMsg

原型

```
STDGT1553BCALL GT1553B_RTReadRxMsg(GT1553BHANDLE ph, GT1553B_UINT8 id,
ST_BC_RCV_MSG_FRAME_PTR msg)
```

功能

RT 读取 RX 接收数据消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

msg：存放接收到的消息。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_BC_RCV_MSG_FRAME msg;
    status = GT1553B_RTReadRxMsg(ph, 0, &msg);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_RTReadRxMsg failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.3.31. GT1553B_RTReadTxMsg

原型

```
STDGT1553BCALL GT1553B_RTReadTxMsg(GT1553BHANDLE ph, GT1553B_UINT8 id, ST_BC_RCV_MSG_FRAME_PTR msg)
```

功能

RT 读取 TX 接收数据消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。

输出参数

msg：存放接收到的消息。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
GT1553B_STATUS status = GT1553B_SUCCESS;
ST_BC_RCV_MSG_FRAME msg;
status = GT1553B_RTReadTxMsg(ph, 0, &msg);
if (status != GT1553B_SUCCESS) {
    debug_printf("GT1553B_RTReadTxMsg failed. Return value = 0x%08x\r\n", status);
    return status;
}
```

9.4. MT 操作接口

9.4.1. GT1553B_MTInit

原型

```
STDGT1553BCALL GT1553B_MTInit(GT1553BHANDLE ph)
```

功能

初始化总线监视器。
MT 初始化执行下述操作：
清 MT 的 RX 缓冲区
MT 默认设置为接收所有消息
使能 MT

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

无。

返回值：

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR
函数也可能返回其它错误码。

Code example

```
GT1553B_STATUS status = GT1553B_SUCCESS;
status = GT1553B_MTInit(ph);
```

```
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_MTInit failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.4.2. GT1553B_MTSetCmdFilterTable

原型

```
STDGT1553BCALL GT1553B_MTSetCmdFilterTable(GT1553BHANDLE ph,  GT1553B_UINT8 id,
GT1553B_DIR dir,  GT1553B_UINT32 mode)
```

功能

MT 设置待监测的消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。
id：RT 的 ID，也即 RT 地址。
Dir：GT1553B_TX - 发送， GT1553B_RX - 接收。
Mode：RT 的模式字

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_BAD_PARAMETER_3
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR
函数也可能返回其它错误码。

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_MTSetCmdFilterTable(ph, 0, GT1553B_TX, 0xFFFFFFFF);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_MTSetCmdFilterTable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.4.3. GT1553B_MTSetEnable

原型

```
STDGT1553BCALL GT1553B_MTSetEnable(GT1553BHANDLE ph, GT1553B_BOOL enable)
```

功能

设置 MT 是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

enable：GT1553B_TRUE - 使能 MT， GT1553B_FALSE - 禁止 MT。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_MTSetEnable(ph, GT1553B_TRUE);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_MTSetEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.4.4. GT1553B_MTGetEnable

原型

```
STDGT1553BCALL GT1553B_MTGetEnable(GT1553BHANDLE ph, GT1553B_BOOL * enable)
```

功能

获取 MT 是否使能。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

enable：GT1553B_TRUE - 使能 MT， GT1553B_FALSE - 禁止 MT。

返回值:

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_BOOL enable;
    status = GT1553B_MTGetEnable(ph, &enable);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_MTGetEnable failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.4.5. GT1553B_MTClearMsgBuffer

原型

```
STDGT1553BCALL GT1553B_MTClearMsgBuffer(GT1553BHANDLE ph)
```

功能

设置 MT 接收消息缓存。在接收消息之前执行此操作，以避免读出硬件缓存中之前残留的消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

无。

返回值:

成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    status = GT1553B_MTClearMsgBuffer(ph);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_MTClearMsgBuffer failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.4.6. GT1553B_MTGetNewMsgNum

原型

```
STDGT1553BCALL GT1553B_MTGetNewMsgNum(GT1553BHANDLE ph, GT1553B_UINT32 * num)
```

功能

查询 MT 新消息数目。即用户还未来得及接收的新消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

num：存放新消息的数目。

返回值：

成功时返回 GT1553B_SUCCESS。

失败时可能返回的返回值列表：

GT1553B_BAD_PARAMETER_2

GT1553B_NULL_DESCRIPTOR

GT1553B_NO_BOARD

GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    GT1553B_UINT32 num;
    status = GT1553B_MTGetNewMsgNum(ph, &num);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_MTGetNewMsgNum failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

9.4.7. GT1553B_MTReadMsg

原型

```
STDGT1553BCALL GT1553B_MTReadMsg(GT1553BHANDLE ph, ST_BC_RCV_MSG_FRAME_PTR msg)
```

功能

MT 读取消息。

参数说明

输入参数

ph：板卡句柄，由函数 GT1553B_OpenCard 得到。

输出参数

msg：存放读取到的消息。

返回值:
成功时返回 GT1553B_SUCCESS。
失败时可能返回的返回值列表：
GT1553B_BAD_PARAMETER_2
GT1553B_NULL_DESCRIPTOR
GT1553B_NO_BOARD
GT1553B_OS_ERROR

Code example

```
    GT1553B_STATUS status = GT1553B_SUCCESS;
    ST_BC_RCV_MSG_FRAME msg;
    status = GT1553B_MTReadMsg(ph, &msg);
    if (status != GT1553B_SUCCESS) {
        debug_printf("GT1553B_MTReadMsg failed. Return value = 0x%08x\r\n", status);
        return status;
    }
```

10. 附录

10.1. 注解

10.1.1. 注 1

使用光纤的 MIL-STD-1553 版本称为 MIL-STD-1773。

10.1.2. 注 2

或者称为大周期。

10.1.3. 注 3

或者称为小周期。

10.1.4. 注 4

BC 发送一个 16 位接收命令字，其后跟随着 1 到 32 个 16 位字。选定的 RT 其后发送一

个单独的 16 位状态字响应。

10.1.5. 注 5

BC 发送一个发送命令字到 RT，RT 发送一个单独的状态字响应，其后跟随着 1 到 32 个数据字。

10.1.6. 注 6

BC 发送一个接收命令字其后紧随一个发送命令字。发送 RT 发送一个状态字其后紧随 1 到 32 个数据字。接下来接收 RT 发送它的状态字。

10.1.7. 注 7

BC 发送一个命令字，子地址为 0 或 31 表示一个模式码类型命令。RT 响应一个状态字。

10.1.8. 注 8

BC 发送一个命令字，子地址为 0 或 31 表示一个模式码类型命令。RT 响应一个状态字，其后紧随一个单独数据字。

10.1.9. 注 9

BC 发送一个命令字，子地址为 0 或 31 表示一个模式码类型命令，其后紧随一个单独数据字。RT 响应一个状态字。

10.1.10. 注 10

BC 发送一个接收命令字，终端地址为 31 指定一个广播类型命令，其后紧随 1 到 32 个数据字。所有实现了广播特性的 RT 接收数据，但没有 RT 会响应。

10.1.11. 注 11

BC 发送一个接收命令字，终端地址为 31 指定一个广播类型命令，其后紧随一个发送命令。发送 RT 发送一个状态字，其后紧随 1 到 32 个数据字。所有实现了广播特性的 RT 接收

数据，但没有 RT 会响应。

10.1.12. 注 12

BC 发送一个接收命令字，终端地址为 31 指定一个广播类型命令，和一个子地址为 0 或 31 指定一个模式码类型命令。没有 RT 响应。

10.1.13. 注 13

BC 发送一个接收命令字，终端地址为 31 指定一个广播类型命令，和一个子地址为 0 或 31 指定一个模式码类型命令，其后紧随一个数据字。没有 RT 响应。

11. 公司简介

西安方解石信息技术有限责任公司位于西安市高新技术产业开发区，是一家专门从事计算机板卡研发、生产、销售和服务为一体的专业性公司。

公司自研板卡涉及多个方面：主要包含图像处理、航空总线、数据采集、内存反射、高速存储等多个类别，同时公司还承接各种板卡定制业务，期待与您的合作。