iTE SDK

RS485模組開發指南

V0.9

ITE TECH. INC.

This is a Preliminary document release. All specifications are subject to change without notice.
The material contained in this document supersedes all previous material issued for the products herein referenced. Please contact ITE Tech. Inc. for the latest document(s).

All sales are subject to ITE's Standard Terms and Conditions, a copy of which is included in the back of this document.

All specifications are subject to change without notice.

Additional copies of this manual or other ITE literature may be obtained from:

ITE Tech. Inc. Tel: 886-2-29126889
Marketing Department Fax: 886-2-2910-2551, 886-2-2910-2552
7F, No.233-1, Baociao Rd., Sindian City,
Taipei County 23145, Taiwan, ROC

You may also find the local sales representative nearest you on the ITE web site.

To find out more about ITE, visit our World Wide Web at:
http://www.ite.com.tw

Or e-mail itesupport@ite.com.tw for more product information/services

修訂記錄

修訂日期	修訂說明	頁次
2014/10/31	初建版本 V0.1	
2015/01/29	版本 V0.2 新增GPIO Pin設定	
2015/05/26	版本 V0.8 新增RS485_1與Software Uart 功能設定	
2015/06/24	版本 V0.9 新增奇偶檢查設定	

目錄

1. 前言

1.1 編寫目的

介紹RS485 模組之功能, 說明RS485模組相關的API之操作及使用.

1.2 適用範圍

RS485通常用來作為智能家居或是工控等應用。RS485使用差動傳輸,所以擁有抗雜訊能力強，傳輸距離較遠與傳輸速率快的優點，因為RS485使用半雙工網路(同一時間內只能傳送或接收)，只需二條線可以達成1對多甚至是多對多通訊)，故可利用RS485介面方便地建立起設備網路。

1.3 適用人員

軟體應用程式，驅動程式開發者

2. RS485模組介紹

2.1 Description

智能家居領域中經常用到RS485 serial bus。以下將介紹在IT9079的SDK中，RS485相關應用的API與使用方式。

2.2 RS485 模組

ITP RS485 driver相關程式放在”\sdk\driver\itp\itp_ RS485.c”
ITP DRIVER是依據POSIX規範實作的API，可以使用OPEN/READ/WRITE/IOCTL等函式對I/O DEVICE進行如讀寫檔案般的操作。以下是基於這種規範所實做的RS485 ITP API。

2.2.1 IOCTL (RS485初始化與重置)

DEVICE ID : ITP_DEVICE_RS485_0 (參考“sdk\include\ite\itp.h”)
ITP_DEVICE_RS485_1

RS485 device 註冊

```
itpRegisterDevice(ITP_DEVICE_RS485_0, &itpDeviceRS485_0);
itpRegisterDevice(ITP_DEVICE_RS485_1, &itpDeviceRS485_1);
```

使用此函式可以將RS485 device註冊到ITP的driver中，使用者可以透過ioctl/rerad/write等函式來操作RS485的功能。本函式已經含在ITP driver的初始化過程當中(參考 “sdk\driver\itp_init_openrtos.c”)，所以一般不必再執行此註冊函式。

RS485 device的初始化

```
/**
 * Device ioctl method (controls device operating parameters).
 *
 * @param file File descriptor referring to an open device.
 * @param request Selects the control function to be performed.
 * @param ptr Additional information that is needed by this specific device to perform the requested function.
 * @param info Device custom data.
 * @return Upon successful completion, it shall return a value other than -1 that depends upon the device control function. Otherwise, it shall return -1 and set errno to indicate the error.
 */
int (*ioctl)(int file, unsigned long request, void* ptr, void* info);

ioctl(ITP_DEVICE_RS485, ITP_IOCTL_INIT, (void*)baud_rate);
```

baud_rate: 輸入參數為baud rate值。

RS485 device的重置

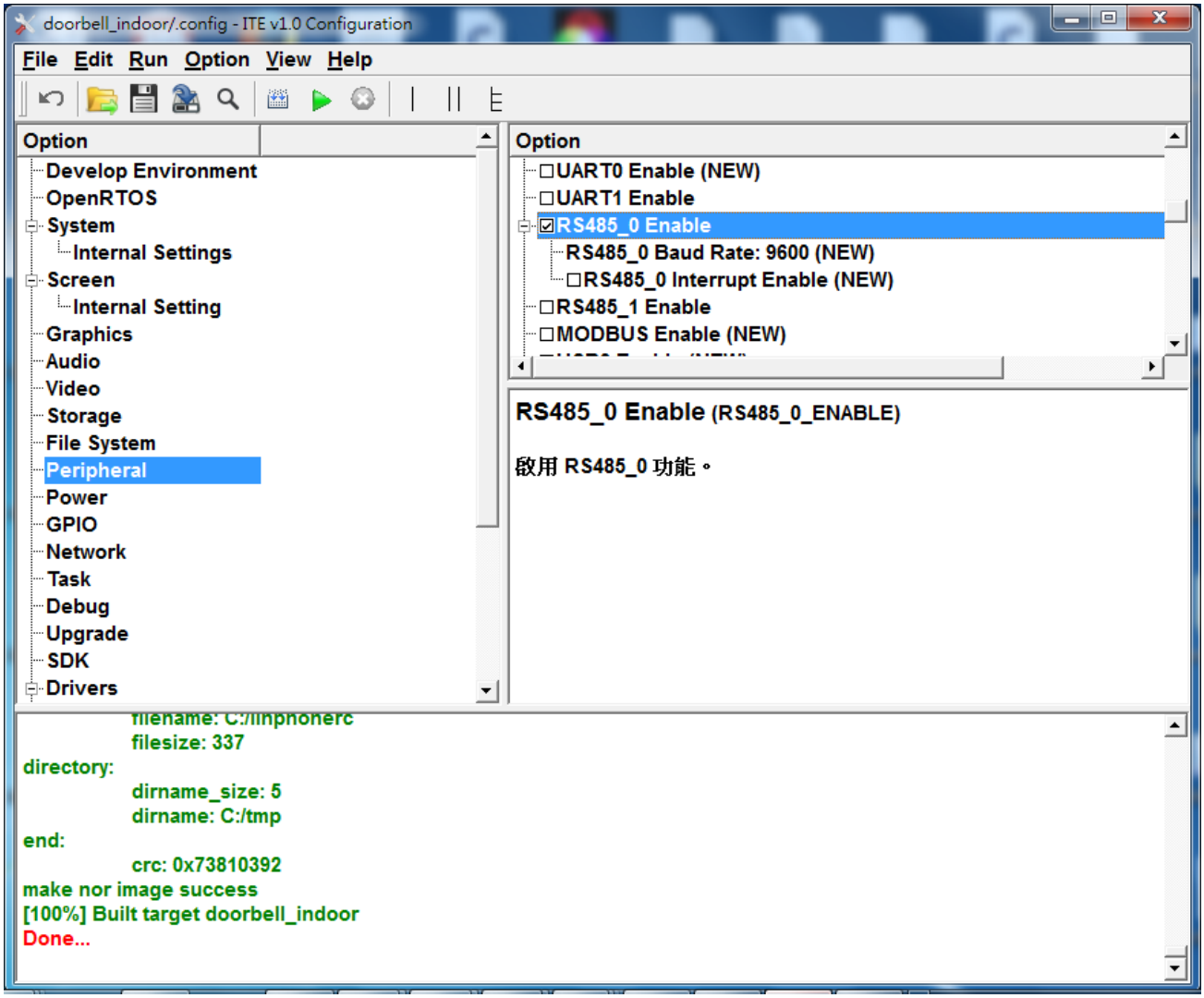
```
ioctl(ITP_DEVICE_RS485, ITP_IOCTL_RESET, (void*)baud_rate);
```

此函式是重置RS485 DEVICE，其中輸入參數為baud rate值。

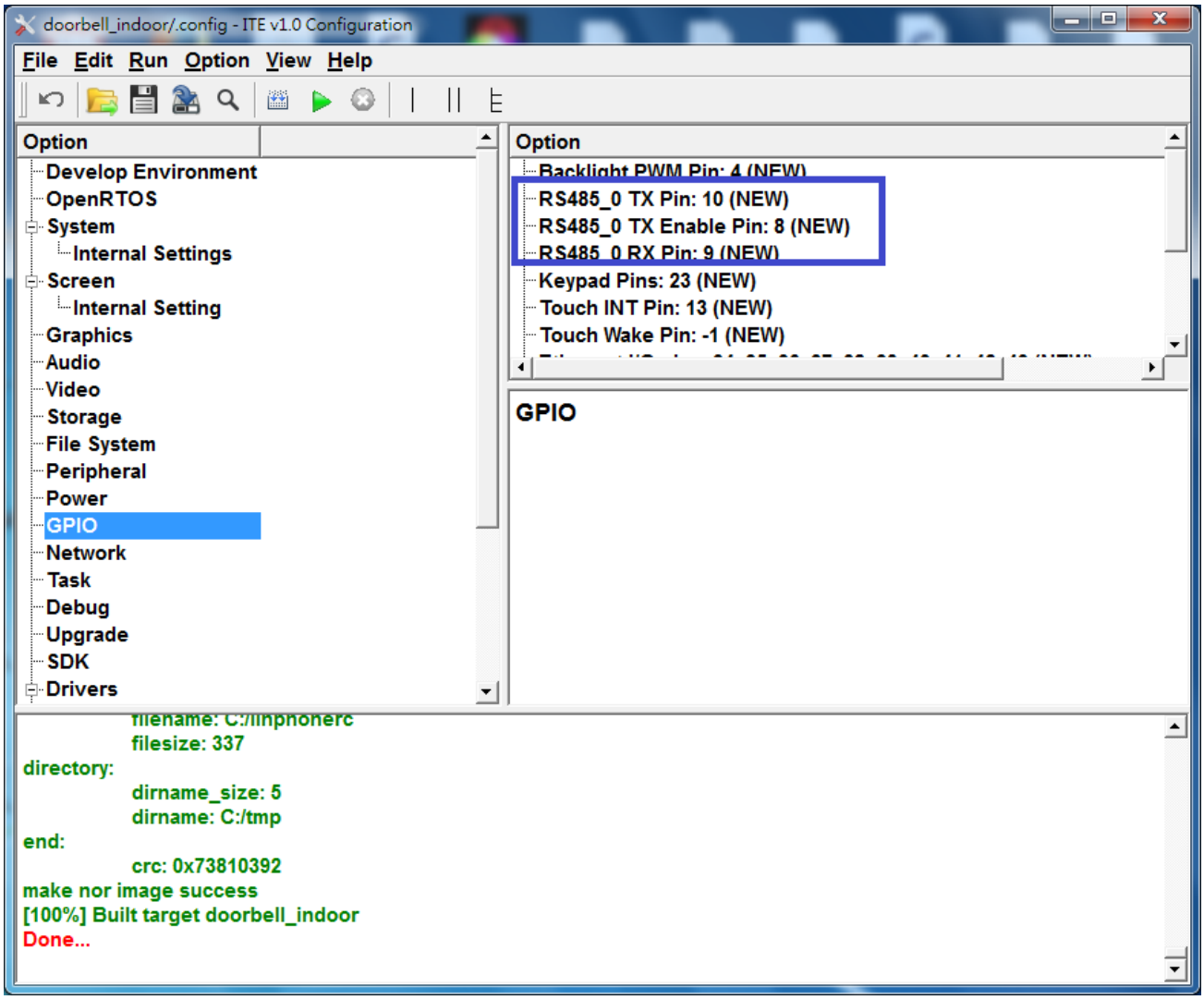
3. 軟件配置說明

3.1 RS485_0參數設定 (KConfig)

To set Kconfig Depend on necessary of project application



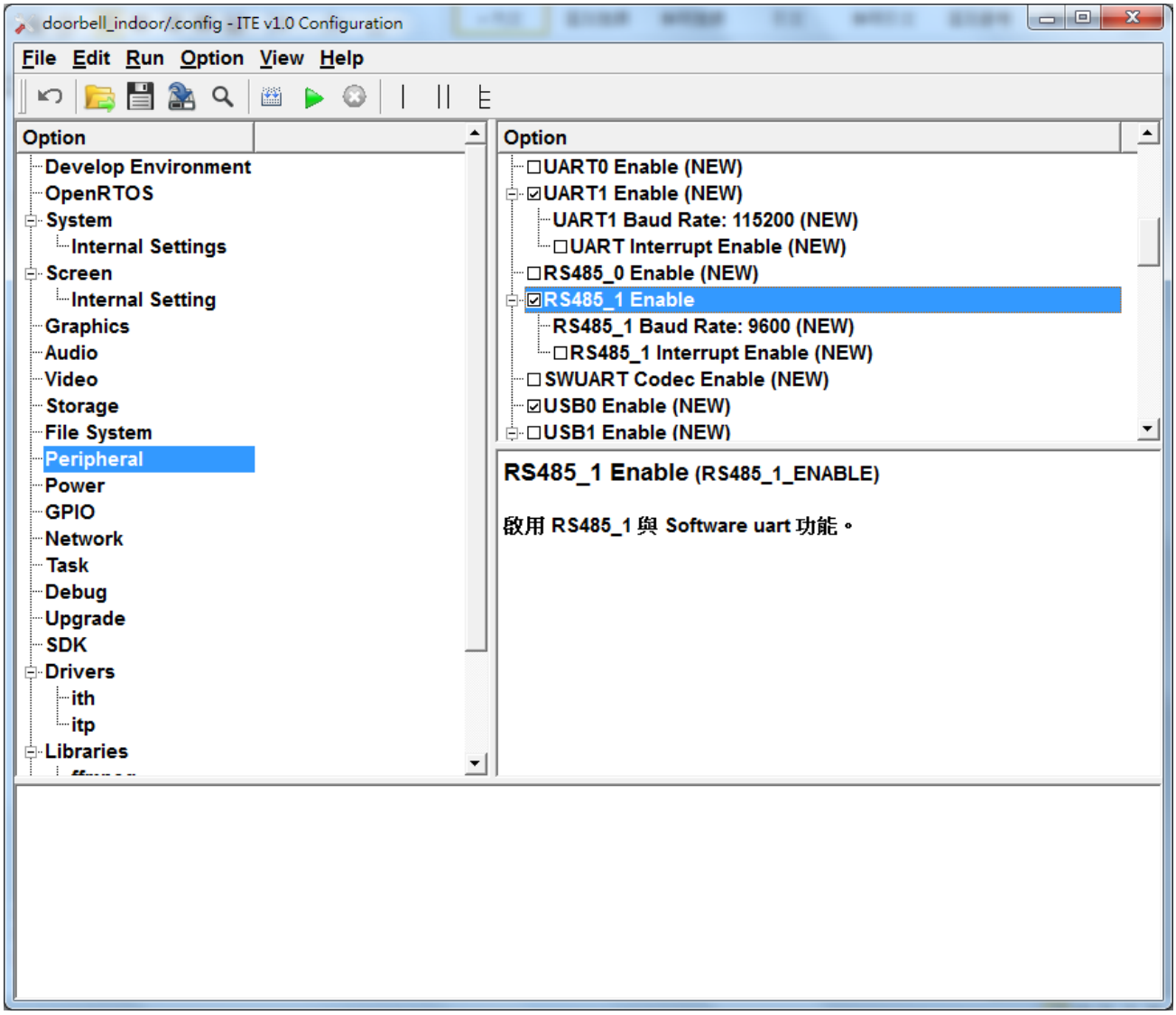
RS485_0 Enable: To enable RS485_0 module.
RS485_0 Baud Rate :To set the Baud Rate of RS485_0 (default 9600bps)。
※Don't enable UART1 and RS485_0 at the same time.



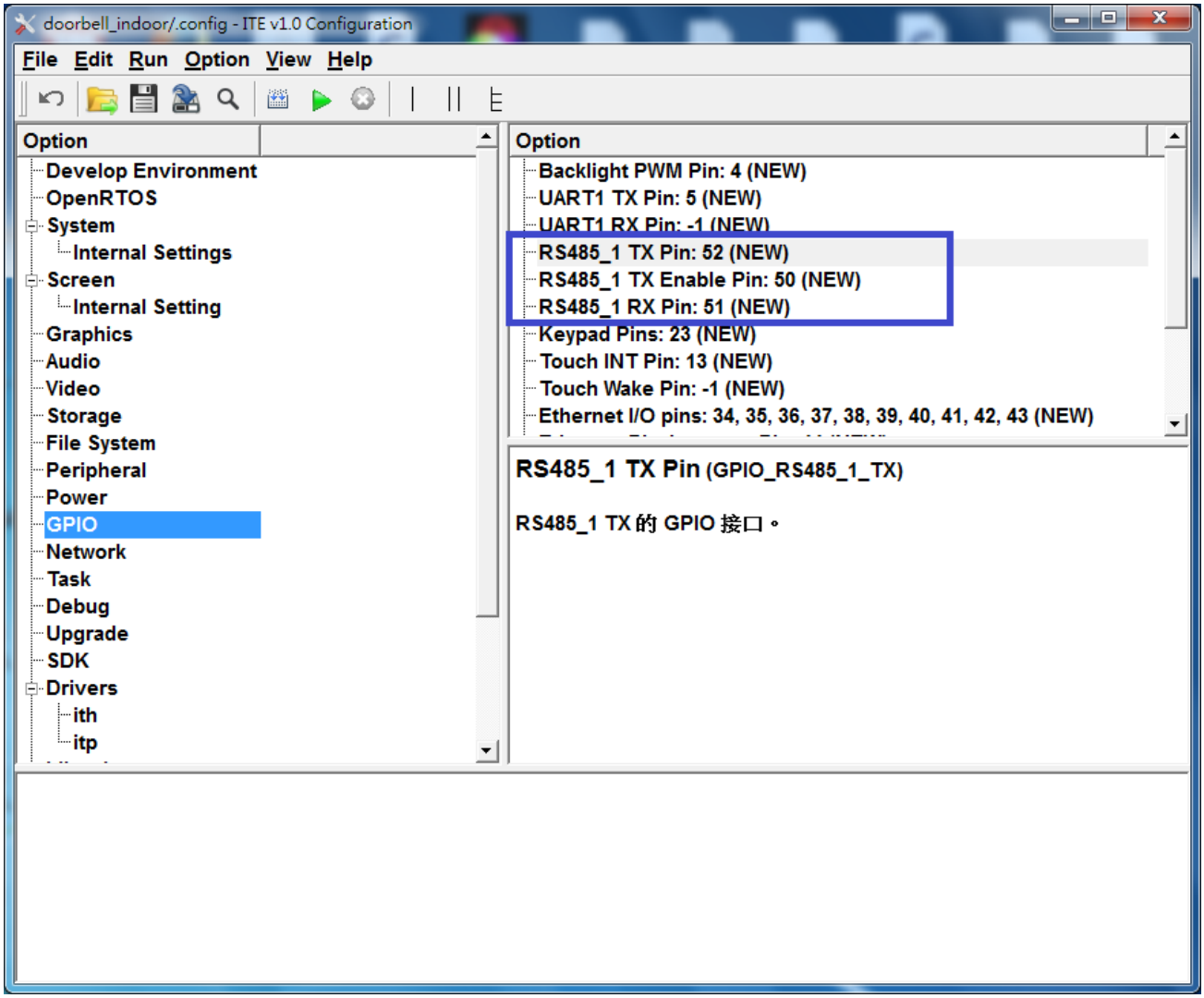
Set RS485_0 TX pin for Write, R485_0 TX Enable pin for Write Enable, and RS485_0 RX pin for Read.

3.2 RS485_1參數設定 (KConfig)

To set Kconfig Depend on necessary of project application



RS485_1 Enable: To enable RS485_1 module.
RS485_1 Baud Rate :To set the Baud Rate of RS485_1 (default 9600bps)。
※It can enable Hardware UART1 and RS485_1(with Software Uart) at the same time.



Set RS485_1 TX pin for Write, R485_1 TX Enable pin for Write Enable, and RS485_1 RX pin for Read.

3.3 MACRO參數定義

Refer to the “sdk\include\ite\itp.h”

```
/**
 * Device types.
 */
typedef enum
{
    // Standard IO devices
    ITP_DEVICE_STD          = (0 << ITP_DEVICE_BIT),    ///< Standard IO
    ITP_DEVICE_SOCKET       = (1 << ITP_DEVICE_BIT),    ///< LWIP socket

    // File system devices
    ITP_DEVICE_FS           = (2 << ITP_DEVICE_BIT),    ///< File systems
    ITP_DEVICE_FAT          = (2 << ITP_DEVICE_BIT),    ///< FAT file system
    ITP_DEVICE_NTFS         = (3 << ITP_DEVICE_BIT),    ///< NTFS file system

    // Custom devices
    ITP_DEVICE_CUSTOM       = (4 << ITP_DEVICE_BIT),    ///< Custom devices
    ITP_DEVICE_PRINTBUF     = (5 << ITP_DEVICE_BIT),    ///< Print buffer
    ITP_DEVICE_SWUART       = (6 << ITP_DEVICE_BIT),    ///< Software UART
    ITP_DEVICE_UART0        = (7 << ITP_DEVICE_BIT),    ///< UART0
    ITP_DEVICE_UART1        = (8 << ITP_DEVICE_BIT),    ///< UART1
    ITP_DEVICE_LCDCONSOLE   = (9 << ITP_DEVICE_BIT),    ///< LCD console
    ITP_DEVICE_OSDCONSOLE   = (10 << ITP_DEVICE_BIT),   ///< OSD console
    ITP_DEVICE_SCREEN       = (11 << ITP_DEVICE_BIT),   ///< Screen
    ITP_DEVICE_I2C          = (12 << ITP_DEVICE_BIT),   ///< I2C
    ITP_DEVICE_SPI          = (13 << ITP_DEVICE_BIT),   ///< SPI
    ITP_DEVICE_IR           = (14 << ITP_DEVICE_BIT),   ///< IR
    ITP_DEVICE_NAND         = (15 << ITP_DEVICE_BIT),   ///< NAND
    ITP_DEVICE_NOR          = (16 << ITP_DEVICE_BIT),   ///< NOR
    ITP_DEVICE_SD0          = (17 << ITP_DEVICE_BIT),   ///< SD0
    ITP_DEVICE_SD1          = (18 << ITP_DEVICE_BIT),   ///< SD1
    ITP_DEVICE_USBDFSG      = (19 << ITP_DEVICE_BIT),   ///< USB acts as a USB Mass Storage device
    ITP_DEVICE_CARD         = (20 << ITP_DEVICE_BIT),   ///< Card
    ITP_DEVICE_DRIVE        = (21 << ITP_DEVICE_BIT),   ///< Drive
    ITP_DEVICE_KEYPAD       = (22 << ITP_DEVICE_BIT),   ///< Keypad
    ITP_DEVICE_POWER        = (23 << ITP_DEVICE_BIT),   ///< Power
    ITP_DEVICE_GSENSOR      = (24 << ITP_DEVICE_BIT),   ///< G-Sensor
    ITP_DEVICE_HEADSET      = (25 << ITP_DEVICE_BIT),   ///< Headset
    ITP_DEVICE_AMPLIFIER    = (26 << ITP_DEVICE_BIT),   ///< Audio amplifier
    ITP_DEVICE_STC          = (27 << ITP_DEVICE_BIT),   ///< STC
    ITP_DEVICE_DECOMPRESS   = (28 << ITP_DEVICE_BIT),   ///< Decompress
    ITP_DEVICE_CODEC        = (29 << ITP_DEVICE_BIT),   ///< Audio codec
    ITP_DEVICE_ETHERNET     = (30 << ITP_DEVICE_BIT),   ///< Ethernet
    ITP_DEVICE_WIFI         = (31 << ITP_DEVICE_BIT),   ///< WiFi
    ITP_DEVICE_FILE         = (32 << ITP_DEVICE_BIT),   ///< File
    ITP_DEVICE_DEMOD        = (33 << ITP_DEVICE_BIT),   ///< Demod
```

```
    ITP_DEVICE_WATCHDOG     = (34 << ITP_DEVICE_BIT),   ///< Watch Dog
    ITP_DEVICE_NETCONSOLE   = (35 << ITP_DEVICE_BIT),   ///< Network console
    ITP_DEVICE_USB          = (36 << ITP_DEVICE_BIT),   ///< USB
    ITP_DEVICE_DPU          = (37 << ITP_DEVICE_BIT),   ///< DPU (encryption/decryption)
    ITP_DEVICE_XD           = (38 << ITP_DEVICE_BIT),   ///< XD
    ITP_DEVICE_LED          = (39 << ITP_DEVICE_BIT),   ///< LED
    ITP_DEVICE_SWITCH       = (40 << ITP_DEVICE_BIT),   ///< Switch
    ITP_DEVICE_TUNER        = (41 << ITP_DEVICE_BIT),   ///< Tuner
    ITP_DEVICE_STNLCD       = (42 << ITP_DEVICE_BIT),   ///< STN LCD
    ITP_DEVICE_USBMOUSE         = (43 << ITP_DEVICE_BIT),    ///< USB Mouse
    ITP_DEVICE_USBKBD           = (44 << ITP_DEVICE_BIT),    ///< USB Keyboard
    ITP_DEVICE_RTC          = (45 << ITP_DEVICE_BIT),  ///< RTC
    ITP_DEVICE_BACKLIGHT        =  (46 << ITP_DEVICE_BIT),   ///< Backlight
    ITP_DEVICE_GPIO_EXTENDER    =  (47 << ITP_DEVICE_BIT),   ///< GPIO Extender
    ITP_DEVICE_RS485_0          =  (48 << ITP_DEVICE_BIT),   ///< RS485
    ITP_DEVICE_RS485_1          =  (49 << ITP_DEVICE_BIT),   ///< RS485

    ITP_DEVICE_LAST
} ITPDeviceType;
```

4. 使用範例

4.1 RS485 使用範例與奇偶檢查使用範例

```
#include "ite/itp.h"

int main(void)
{
        int i;
        uint8_t getstr[8];
        uint8_t sendtr[8] = {0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8};
        int len = 0;
        ITHRS485Port RS485_port = ITH_RS485_0;
        ITHRS485Port RS485_port1 = ITH_RS485_1;
        ITHUartParity RS485_UartParity = ITH_UART_ODD;  // Here enter the odd or even or normal
                                                        //parameter.
                                                        // normal :  ITH_UART_NONE
                                                        // odd      :  ITH_UART_ODD
                                                        // even    :  ITH_UART_EVEN

        printf("Start RS485 test!\n");

        ioctl(ITP_DEVICE_RS485_0, ITP_IOCTL_RESET, (void*)&RS485_UartParity);
        ioctl(ITP_DEVICE_RS485_0, ITP_IOCTL_ON, (void*)&RS485_port);

        ioctl(ITP_DEVICE_RS485_1, ITP_IOCTL_RESET, (void*)&RS485_UartParity);
        ioctl(ITP_DEVICE_RS485_1, ITP_IOCTL_ON, (void*)&RS485_port1);

        while(1)
        {
           write(ITP_DEVICE_RS485_0,sendtr,8);
           len = read(ITP_DEVICE_RS485_0,getstr,8);

           if(len > 0)
           {
                printf("RS485_0 read: ");
                for(i = 0; i <len; i++)
                {
                        printf("[%02x]",getstr[i]);
                }
                printf("\n ");
           }

          write(ITP_DEVICE_RS485_1,sendtr,8);
           len = read(ITP_DEVICE_RS485_1,getstr,8);

           if(len > 0)
```

```
        {
          printf("RS485_1 read: ");
          for(i = 0; i <len; i++)
          {
                printf("[%02x]",getstr[i]);
          }
          printf("\n ");
        }


}
```