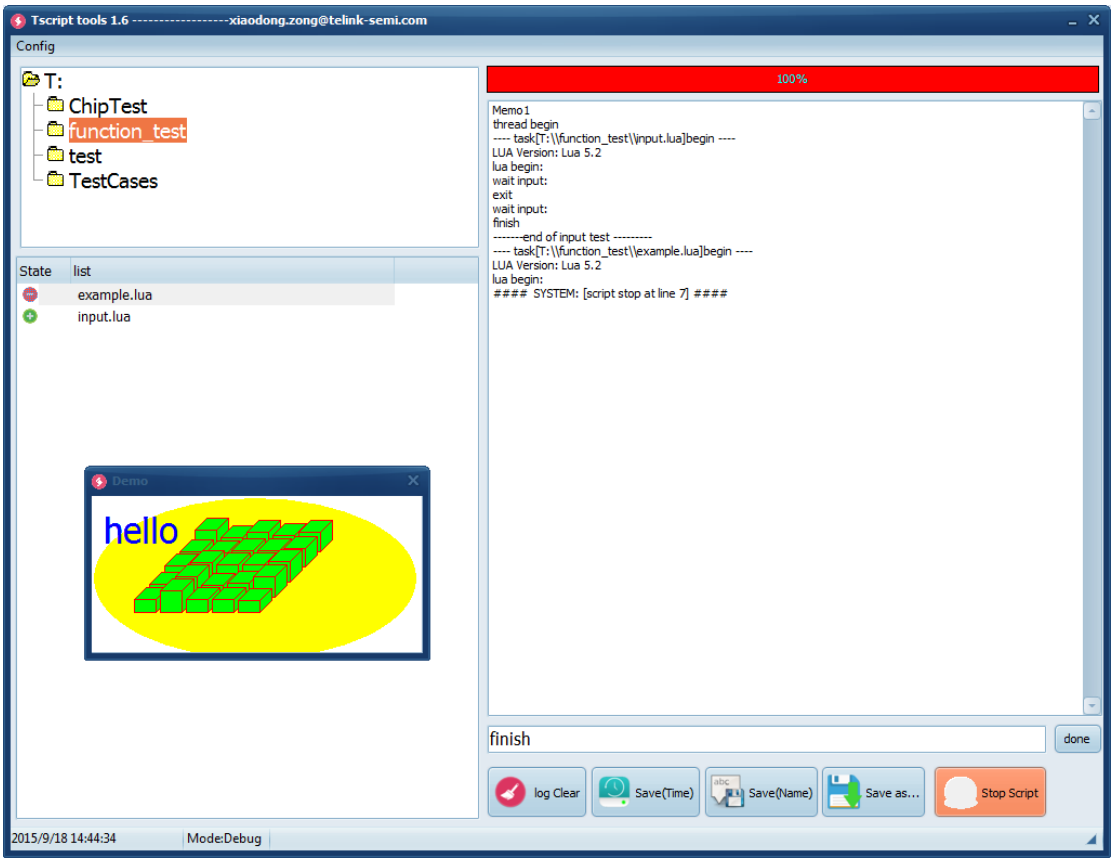
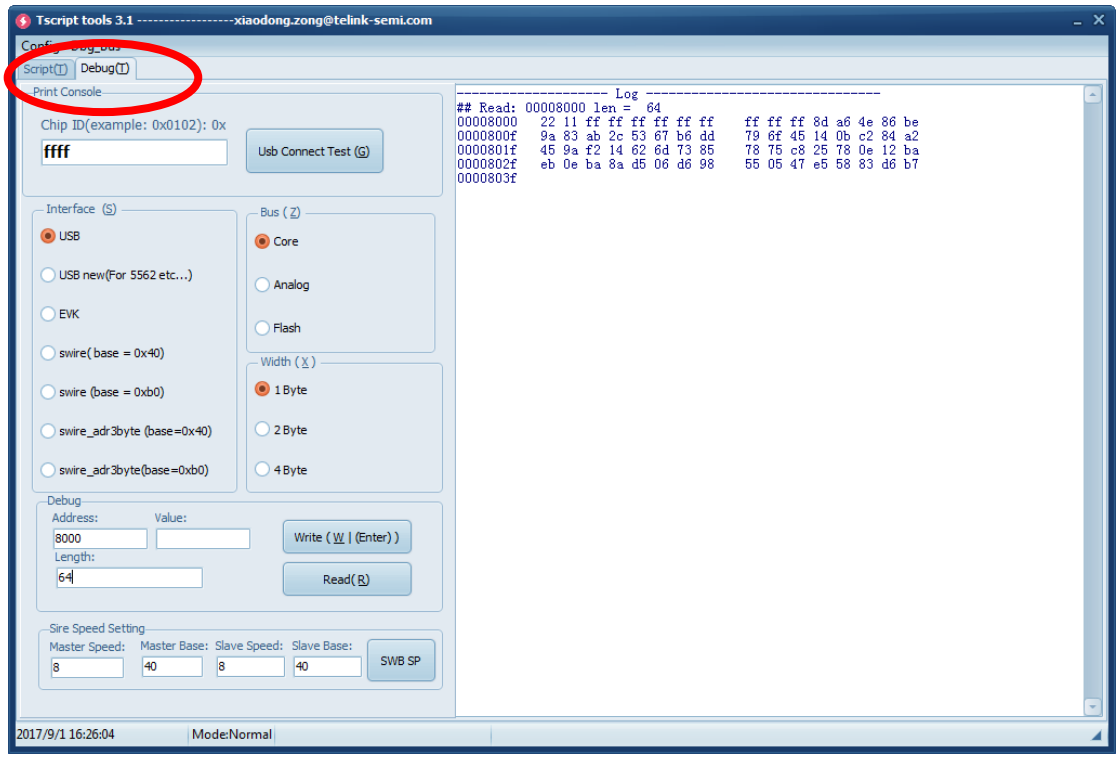
Tscript 使用说明

日期	版本	作者	说明
2015/8/18	v1.5	宗晓东	第一版
2015/9/18	v1.6	宗晓东	加入了绘图函数,以及键盘输入,tl_stop 函数
2016/9/20	v2.1	宗晓东	加入了 usb_bulk, rs232 相关函数, 加入虚拟磁盘映射配置,可以启动多个 Tscript 实例
2016/9/23	v2.2	宗晓东	usb_bulk_in 数据复制了一次，解决了一个 bug, 加入打印颜色控制，现在可以打印彩色字符
2016/10/28	v2.4	宗晓东	修改了 rs232 接收的部分,从第一次收到数据开始,最多可以接收 20ms 的数据,如果打开非法的 RS232 port 加入了报错信息
2016/11/25	v2.5	宗晓东	修改了 tl_sleep_ms 功能,可以在 debug 模式及时停止脚本运行
2016/12/15	v2.6	宗晓东	修复了一个 print 会造成死机的 bug
2017/2/10	v2.8	宗晓东	usb bulk in 线程中,readfile 函数会造成 block 从而引起切换 usb 设备时,无法再读到 usb bluk in 数据,在 tl_usb_bulk_monitor_end()中加入了 CancelIoEx 函数解决.
2017/2/17	v 2.9	宗晓东	添加一个 tl_rs232_delay_set 函数,可以设置 rs232 线程获得数据包的 delay 时间
2017/6/12	v3.0	宗晓东	加入 5562 芯片的 usb 接口函数 tl_usb_init2 tl_usb_write2 tl_usb_read2
2017/9/1	V3.1	宗晓东	加入了一个调试界面，可以提供类似 wpcdb 的读写寄存器接口。 增加了一个主菜单目录，可以在脚本执行前定义一个变量。 增加了如下函数： tl_evk_read,tl_evk_write,tl_swire_read,tl_swire_write,tl_flash_read,tl_flash_write,tl_analog_read,tl_analog_write 详细见函数说明
2017/9/4	V3.2	宗晓东	增加了按键 button,edit 相关函数，可以从界面获得输入： tl_button_show,tl_button_clear,tl_edit_show,tl_edit_clear, tl_message_get

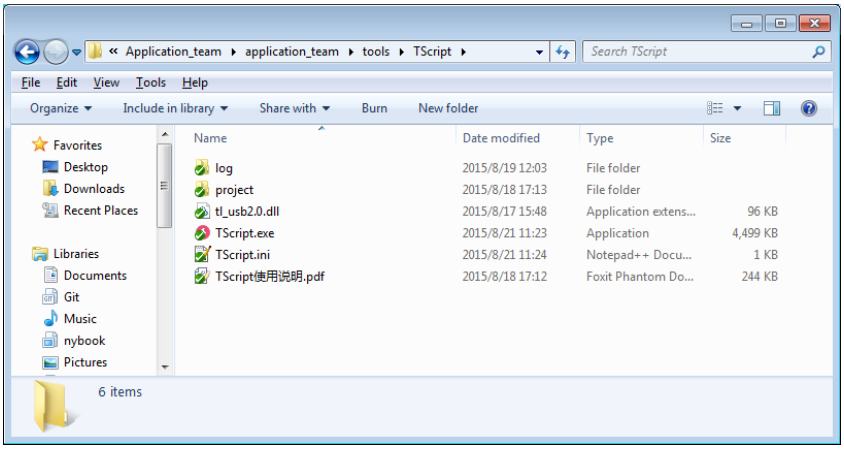
1 工具介绍



V3.1 之后新增加的界面如下：



Tscript 工具主要用来运行 lua 脚本，在 lua 脚本中可以执行一些 USB 的操作函数.以及画图操作,字符串命令输入等函数,从而可以做一些芯片测试,以及工具扩展.
工具的目录结构如下:



project 目录:　放置 lua 脚本的目录,里面可以建立任意子目录,脚本扩展名为*.lua
脚本尽量不要放在根目录下,要创建子目录进行脚本分组.

log 目录:　默认 log 的保存目录,点击 Save(Time)按钮可以保存 log 窗口的内容到 log 目录下，文件名类似: log2015-8-19 11h22m27s.txt，点击 Save(Name)按钮,保存 log 窗口的内容到 log 文件夹下,文件名为脚本名字类似: Case_t1.txt.

Tscript.ini　配置文件，配置 Tscript.exe 的一些基本属性，目前只有两个：

EDITOR= "C:\Windows\notepad.exe"	:	默认编辑器
DEBUGMODE = 0	:	默认调试模式,1 为 debug 模式
VDISK = "T"	:	配置虚拟磁盘的盘符

2　脚本介绍

Tscript.exe 中使用的脚本为 lua 格式,文件扩展名为*.lua
其中加入了一些自定义函数:

底层接口函数:

注意:

5560 芯片,5562 芯片的 usb，swire 硬件接口和早期芯片是不同的,所以针对 5562 芯片有 3 个函数是专用的, tl_usb_init2,　tl_usb_write2, tl_usb_read2　其他 usb 相关的函数是共用的. 这 3 个函数的参数以及用法和　tl_usb_init, tl_usb_write,tl_usb_read 一样,可以参照下面说明.

1　int tl_usb_init(id)

usb 初始化函数
id　:　芯片的{0x7f,0x7e}地址里的芯片 ID
返回值为 usb 的设备句柄

例子：

```
handle = tl_usb_init(0x1234)
```

2　int tl_usb_write(handle, adr, buffer, len)

handle : 通过 tl_usb_init 获得的 usb 设备句柄
adr　:　地址

buffer : array 类型的变量

len : 写数据的长度

返回值为写成功的数据长度

例子：

```
a = array.new(4)
a[1] = 0x11; a[2] = 0x22; a[3] = 0x33; a[4] = 0x44;
result_len = tl_usb_write(handle,0x8000,a,16)
```

3 r ,r_len = tl_usb_read(handle, adr, len)

handle : 通过 tl_usb_init 获得的 usb 设备句柄

adr : 地址

len : 写数据的长度

返回值 r 为 table 类型的变量，保存读到的数据值

返回值 r_len 为读取到的数据长度

例子：

```
r,r2 = tl_usb_read(handle,0x8000,16)
```

4 int tl_usb_bulk_out(handle,buffer,len)

handle : 通过 tl_usb_init 获得的 usb 设备句柄

buffer : array 类型的变量

len : 写数据的长度

返回值为写成功的数据长度

例子:

```
a = array.new(4)
a[1] = 0x11; a[2] = 0x22; a[3] = 0x33; a[4] = 0x44;
len = tl_usb_bulk_out(handle,a,4)
```

5 int tl_usb_bulk_monitor_init(id);

usb bulk 初始化函数

id : 芯片的{0x7f,0x7e}地址里的芯片 ID

返回值为 usb 的设备句柄

例子：

```
handle_bulk = tl_usb_bulk_monitor_init(0xffff);
```

6 tl_usb_bulk_monitor_start(handle_bulk)

handle_bulk : 通过 tl_usb_bulk_monitor_init 获得的 USB Bulk 设备句柄

这个函数使能了 usb bulk 接收线程

7 tl_usb_bulk_monitor_end()

这个函数暂停了 usb bulk 接收线程

8 r, r_len = tl_usb_bulk_read()

这个函数查询了 usb bulk 的接收 buffer

返回值 r 为 table 类型的变量，保存读到的数据值

返回值 r_len 为读取到的数据长度

关于上面 monitor 相关的函数,一个简单的例子如下:

```
handle_bulk = tl_usb_bulk_monitor_init(0xffff);
tl_usb_bulk_monitor_start(handle_bulk)
```

```
repeat
    result_tbl,result_len = tl_usb_bulk_read()
    tl_sleep_ms(50);
until (result_len>0)
tl_usb_bulk_monitor_end()
```

注意: 目前一个 Tscript 只支持一个 usb bulk in 接收, 但可以支持多个 usb bulk out.

9 r ,r_len = tl_swire_read(handle, adr, len,mode,base)

handle : 通过 tl_usb_init 获得的 usb 设备句柄

adr : 地址

len : 写数据的长度

mode : 决定 swire 的模式（协议地址是 2byte 还是 3byte）

0 : 2 byte (5562 之前的芯片)

1 : 3byte(5562 芯片)

base : swire master 寄存器基地址，一般为 0x40 或者 0xb0

返回值 r 为 table 类型的变量，保存读到的数据值

返回值 r_len 为读取到的数据长度

10 int tl_swire_write(handle, adr, buffer, len,mode,base)

handle : 通过 tl_usb_init 获得的 usb 设备句柄

adr : 地址

len : 写数据的长度

buffer : array 类型的变量

mode : 决定 swire 的模式（协议地址是 2byte 还是 3byte）

0 : 2 byte (5562 之前的芯片)

1 : 3byte(5562 芯片)

base : swire master 寄存器基地址，一般为 0x40 或者 0xb0

返回值为写成功的数据长度

11 int tl_evk_write(handle,adr,buffer,len)

参数，用法和 tl_usb_write 一样。需要注意，此处 handle 为 evk 板子的 usb handle 值

12 int tl_evk_read(handle,adr,len)

参数用法和 tl_usb_read 一样，此处 handle 为 evk 板子的 usb handle 值

高级接口函数：

本节中的函数是基于底层接口函数实现的
本节中所有的函数都有一个参数 mode,主要定义了接口类型。
mode 的定义如下：

mode	底层接口
1	usb(5562 芯片之前的老的 usb 接口)
2	usb new(5562 芯片的 usb 接口)
3	原来的 swire (master 的 swire 寄存器基地址在 0x40)
4	原来的 swire (master 的 swire 寄存器基地址在 0xb0)
5	地址为 3byte 的 swire　(master 的 swire 寄存器基地址在 0x40)
6	地址为 3byte 的 swire　(master 的 swire 寄存器基地址在 0xb0)
7	通过 evk 访问 dut

另外，本节中所有的 handle 都为直接跟电脑相连的设备的 handle。例如：如果通过 evk 访问 dut,那么函数中的 handle 应该为 evk 的 handle

int　tl_analog_read（handle,adr,len,mode）
handle, adr,len 参数和 tl_usb_read 是一样的，mode 可以参照本节开始的定义。
说明：　读取芯片的模拟寄存器

int tl_analog_write(handle,adr,buffer,len,mode)
handle,adr,buffer,len 的定义和 tl_usb_write 是一样的，mode 可以参照本节开始的定义。
说明：　写芯片的模拟寄存器

int tl_flash_erase(handle,adr)
说明：　擦除 adr 开始的 4K 的 flash 内容，返回值固定为 1

int　tl_flash_read（handle,adr,len,mode）
handle, adr,len 参数和 tl_usb_read 是一样的，mode 可以参照本节开始的定义。
说明：　读芯片的 flash

int tl_flash_write(handle,adr,buffer,len,mode)
handle,adr,buffer,len 的定义和 tl_usb_write 是一样的，mode 可以参照本节开始的定义。
说明：　写芯片的 flash

系统相关函数:

1　tl_sleep_ms(t)
睡眠函数
t　：　睡眠时间(ms 为单位)
2　tl_error(error_bit)

报错函数,如果脚本里面调用过此函数,脚本运行后进度条为红色,否则为绿色

error_bit : 错误标志

3 tl_stop(pos)

强制脚本结束,脚本异常结束,进度条为红色.

4 tl_progress(pos)

进度条控制函数

pos : 设置进度条的当前值(0 ~ 100)

5 str = tl_input_get()

读取输入, 用户可以通过 log 窗口下的文本框窗口输入字符串,然后点击 done 按键或者按键盘上的"enter"键来将字符串写入输入缓存.

如果输入缓存没有数据, 该函数返回"NULL"字符串.

如果输入缓存有数据,该函数返回相应的字符串.

6 tl_input_bufclr()

清除输入缓存数据,一般情况下不需要调用此函数,每次调用 tl_input_get 函数后,会自动清除输入缓存.

7 tl_log_color(color)

color : 颜色是一个 long 类型变量格式如下: 0x00bbggrr

注意:由于 print 的字符不是马上显示出来,有一定的缓存时间,所以调用此函数后,最好调用 tl_sleep_ms(100) 等待 100 毫秒, 否则此函数后的打印颜色可能不能马上改变。

画图函数:

1 tl_form_show(x,y,width,height)

显示画图窗口, 显示在电脑屏幕的(x,y)坐标处,窗口的宽度为 width,高度为 height

2 tl_form_close()

关闭画图窗口

3 tl_form_draw_ratangle(x1,y1,x2,y2,bcolor,pcolor)

画一个矩形,矩形左上角坐标为(x1,y1), 右下角坐标为(x2,y2), 矩形填充颜色为 bcolor

矩形边缘颜色为 pcolor

注: 关于颜色的数值说明

颜色是一个 long 类型变量格式如下: 0x00bbggrr

rr 为红色分量, gg 为绿色分量,bb 为蓝色分量

下面关于颜色的数值都是这个规则.

4 tl_form_draw_ellipse(x1,y1,x2,y2,bcolor,pcolor)

画一个椭圆,椭圆的外切矩形的左上角坐标为(x1,y1),右下角坐标为(x2,y2)

5 tl_form_draw_line(x1,y1,x2,y2,pcolor)

画一条直线

6 tl_form_draw_polygon(pbuffer, point_cnt, bcolor, pcolor)

画一个不规则多边图

pbuffer: arrary 类型变量

(pbuffer[0],pbuffer[1]) 为第一点坐标

(pbuffer[2],pbuffer[3]) 为第二点坐标 …. 以此类推.

point_cnt: 多边行点数

7 tl_form_draw_text(x, y, str, size, pcolor)

在窗口上(x,y)位置显示一个字符串, 字符串字体大小由 size 控制, str 为字符串.

8 tl_button_show(x,y,width,height, caption,id)

在窗口的(x,y)位置创建一个按键，按键的宽度为 width,高度为 height. caption 为按键显示的名字,按键的 id 为按键的编号，取值范围为（1 ~ 20），所以最多 20 个按键，这个编号在后面的 tl_message_get 函数中会用到。

9 tl_edit_show(x,y,width,height,caption,text,id)

在窗口的的(x,y)位置创建一个 editbox，editbox 的宽度为 width,高度为 height. caption 为 editbox 显示的名字, text 为 editbox 中默认的输入字符串，editbox 的 id 为 editbox 的编号，取值范围为（1 ~ 20），所以最多 20 个 editbox，这个编号在后面的 tl_message_get 函数中会用到。

10 tl_button_clear()

清除所有按键

11 tl_edit_clear()

清除所有的 editbox

12 text_tbl, btn_id = tl_message_get()

该函数等待按键按下，返回所有的 editbox 的内容和所按下的按键的 ID 号。

text_tbl ： table 类型的变量，内容为所有 editbox 的内容，内容为字符串。

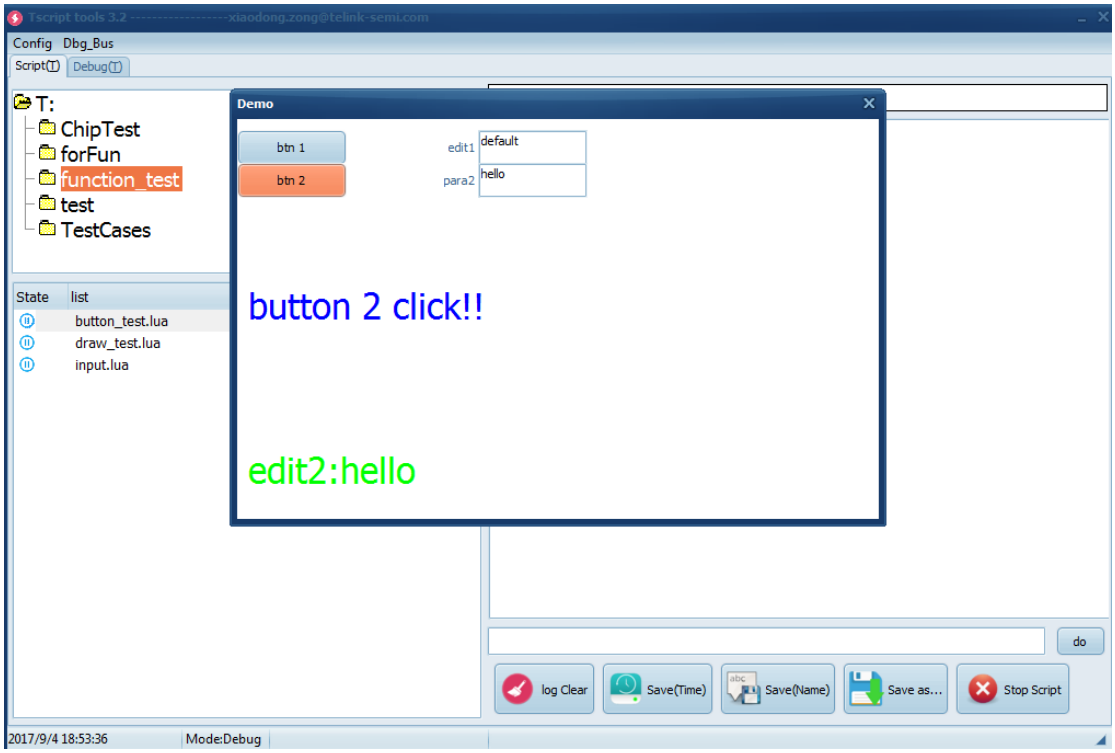
例如 text_tbl[1] 为 id=1 的 editbox 输入的内容

btn_id : 按下的按键的 ID 号，如果 ID=255 表示窗口关闭，这是会造成脚本终止。

上面几个函数组合使用的一个例子如下：

```
tl_form_draw_ratangle(0,0,600,400,0xffffff,0xffffff)
tl_form_show(0,0,600,400)
tl_button_show(0,10,100,32,"btn 1",1)
tl_button_show(0,40,100,32,"btn 2",2)
tl_edit_show(220,10,100,32,"edit1","default",1)
tl_edit_show(220,40,100,32,"para2","hello",2)
result_tbl = {};
idx=1
for loop = 1,100 do
    result_tbl,btn_idx = tl_message_get();
    tl_form_draw_ratangle(0,0,600,400,0xffffff,0xffffff)
    if btn_idx == 1 then
        tl_form_draw_text(10,150,"button 1 click!!",25,0xff0000)
        tl_form_draw_text(10,300,string.format("edit1:%s",result_tbl[1]),25,0x00ff00)
    end
    if btn_idx == 2 then
        tl_form_draw_text(10,150,"button 2 click!!",25,0xff0000)
        tl_form_draw_text(10,300,string.format("edit2:%s",result_tbl[2]),25,0x00ff00)
    end
    if btn_idx>100 then
        break
    end
end
```

上面例子的运行结果如下：



按不同的按键，有不同的反应。

rs232 相关函数:

1 rs232_tbl,rs232_tbl_cnt = tl_rs232_list()

获得 rs232 的列表,

rs232_tbl 是返回的字符串 table,

rs232_tbl_cnt 是返回的设备个数

rs232_tbl 里的内容格式为: {“COM1”, “COM2”, “COM3”, ….}

2 tl_rs232_open(port_name, baudrate_idx)

打开串口

port_name :是端口名字,格式为”COM2”

baudrate_idx :波特率索引,参照下面表格

索引	波特率
0	110
1	1000000
2	600
3	1200
4	2400
5	4800
6	9600
7	14400
8	19200
9	38400
10	56000

11	57600
12	115200
13	128000
14	256000
15	

3 tl_rs232_send(a,len)

串口发送函数

a ： array 类型的变量

len : 发送的数据长度

例子:

```
a = array.new(4)
a[1] = 0x11; a[2] = 0x22; a[3] = 0x33; a[4] = 0x44;
tl_rs232_send(a,4)
```

4 rs232_rcv_tbl,len = tl_rs232_recv()

串口接收函数

rs232_rcv_tbl ： 接收到的数据

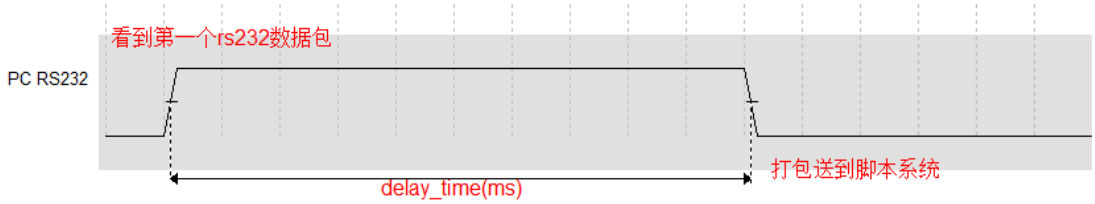
len ： 接收到的长度

5 tl_rs232_close()

关闭串口

6 tl_rs232_delay_set(delay_time)

设置 rs232 获得数据的 delay 时间



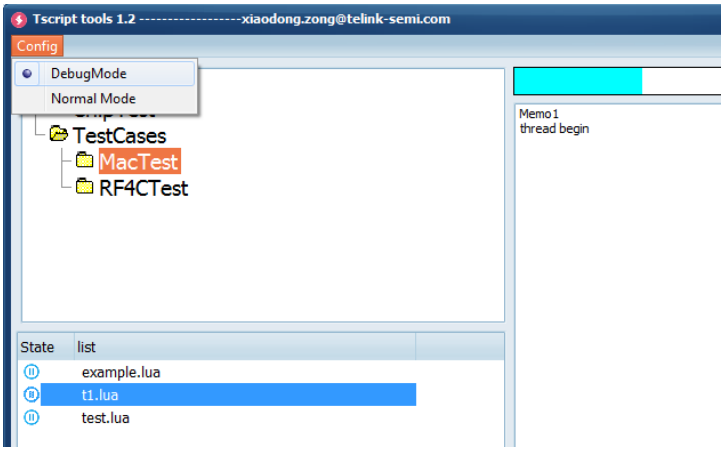
一个简单的 rs232 例子如下(发送和接收回环测试) :

```
rs232_tbl = {}
rs232_tbl,rs232_tbl_cnt = tl_rs232_list()
print ("len:",rs232_tbl_cnt)
for i,v in ipairs(rs232_tbl)
do
    print(string.format("%s",v))
end
print(string.format("open the first device:%s",rs232_tbl[1]))
tl_rs232_open(rs232_tbl[1], 1)
alen = 10
a = array.new(alen)
for i=1,alen do
    a[i] = 0x55;
end
```

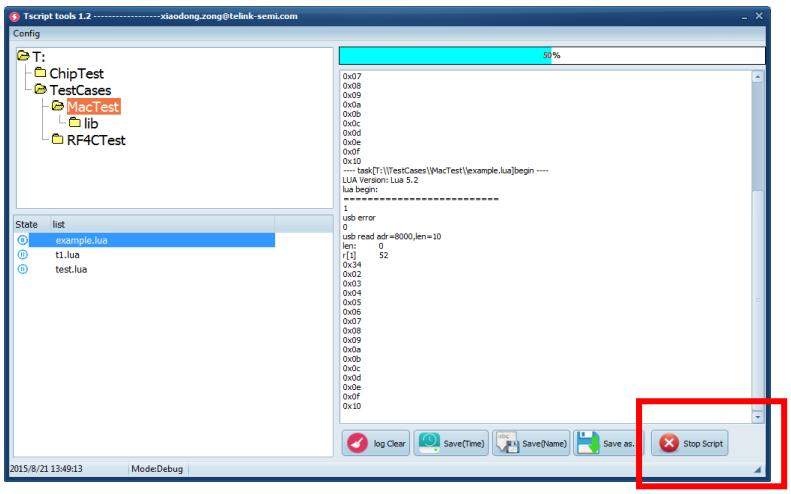
```
tl_rs232_send(a,10)
rs232_rcv_tbl={}
repeat
        rs232_rcv_tbl,len = tl_rs232_recv()
until len>0
for i,v in ipairs(rs232_rcv_tbl)
do
        print(string.format(" %x",rs232_rcv_tbl[i]))
        tl_sleep_ms(50)
end
print("close uart port now")
tl_rs232_close()
```

3 调试模式

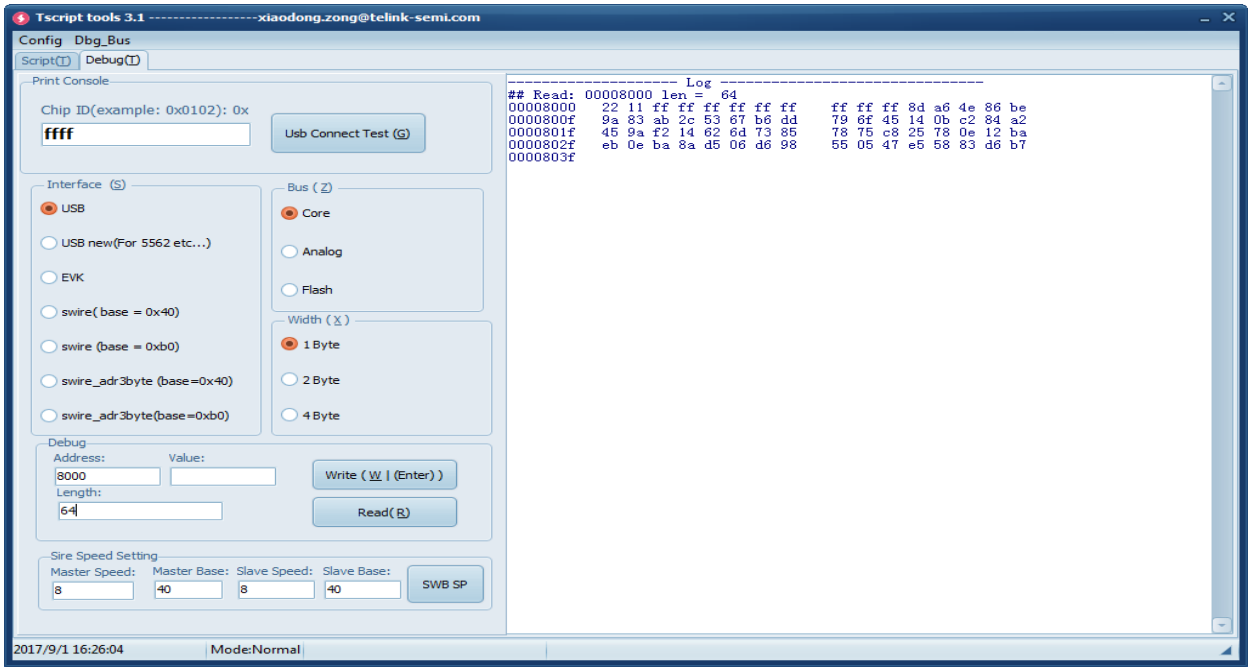
脚本运行有两种模式　debug 模式 和 normal 模式
debug 模式运行速度较慢,但是可以加入断点,可以暂停脚本运行,可以强制停止脚本.
目前 Tscript 工具的 debug 模式只支持强制停止脚本功能.
normal 模式脚本可以全速运行.
建议在 debug 脚本时,用 debug 模式,这样可以在脚本进入死循环时,强制结束脚本.
模式的选择如下所示:



在 debug 模式下,脚本运行时,可以通过下面的按键来强制脚本结束:



新的界面说明：



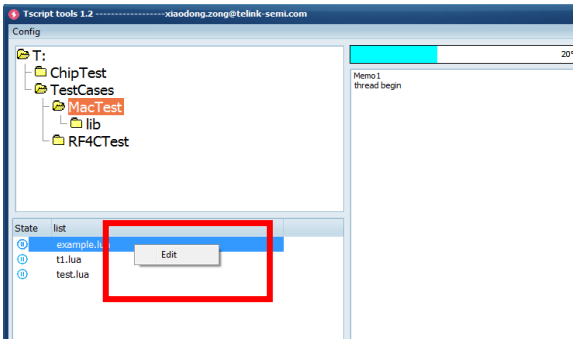
新界面和 wpcdb 操作类似，所有带下滑线的字母都是快捷键（不分大小写）下面是快捷键列表：

快捷键	功能
T	切换脚本界面和调试界面
S	切换接口类型
Z	切换读写区间（core, falsh,analog..）
X	读取数据宽度(1byte,2byte,4byte)
W	写操作
R	读操作（可以读多个 byte,由 length 输入框制定）
G	测试 USB 连通性
Enter	写操作，类似 wpcdb 写 1byte,2byte 或者 4byte
Tab	读操作，类似 wpcdb 读 1byte,2byte 或者 4byte

说明： 整个调试界面只有 address, data 两个输入框可以获得焦点，按”Tab”键可以在两个输入框切换，读取寄存器内容。按”enter”可以把 data 输入框的数据写入 address 输入框指定的寄存器中。

4 编辑脚本

可以使用任何一款文本文件编辑器写脚本,只要扩展名为*.lua 就可以了.
在界面中,可以通过鼠标右键点击脚本名字选择 edit,用文本编辑器打开脚本，文本编辑器在 Tscript.ini 文件中设置.


Tscript tools 1.2 ----------xiaodong.zong@telink-semi.com
Config
T:
ChipTest
TestCases
MacTest
lib
RF4CTest
20°
Memo1
thread begin
State
list
example.lua
Edit
t1.lua
test.lua