

Android Bluetooth/Ethernet/USB 函式库使用说明

1. GTSPL_openPort()

★ Bluetooth

GTSPL_openPort(MacAddress)

■ 函式说明：指定蓝芽的 MAC 地址(请使用 BR/EDR MacAddress)，开启输出埠

■ 参数说明：

➔ address: 字符串型别，指定联机的蓝芽地址(Bluetooth MacAddress)，
如： " DC:1D:30:00:1D:87"

★ Ethernet

GTSPL_openPort(IP,Port,time)

■ 函式说明：指定打印机的 IP 地址与端口，开启输出埠

■ 参数说明：

➔ IP: 字符串型别，指定联机的 IP 地址 ， 如： "192.168.1.109"

➔ Port: int 型别，指定联机的端口如： 8899

➔ time: int 型别，延迟时间，1000=1 秒，范例： GTSPL_closePort(2000)

★ USB

GTSPL_openPort(manager,device)

■ 函式说明：指定 USB 相关变数，开启输出端口

■ 参数说明：

➔ manager: USBManager 型别，输入 USBManager 变量

➔ device: USBDevice 型别，输入 USBDevice 变量

2. GTSPL_closePort()

■ 函式说明：关闭输出埠

■ 参数说明：无

3. GTSPL_closePort(time)

■ 函式说明：关闭输出埠

■ 参数说明：

➔ time: int 型别，延迟时间，1000=1 秒，范例：GTSPL_closePort(2000)

4. GTSPL_setCmdSendMode(mode)

■ 函式说明：设定命令传送至打印机或档案

■ 参数说明：

➔ mode: 字符串型别

F:将命令传送至档案

(档案位置在内部储存空间/android/data/packageName/files 下)

P:将命令传送至打印机

5. GTSPL_setup(width, height, speed, density, sensor, sensorDistance, sensorOffset, context)

■ 函式说明：设定卷标的宽度、高度、打印速度、打印热度、传感器类别、间隙/黑标垂直间距、间隙/黑标偏移距离

■ 参数说明：

参数	型别	说明
width	int	设定卷标宽度，单位 mm
height	int	设定卷标高度，单位 mm
speed	int	设定打印速度，1~15，代表每秒 1~15 吋打印速度(随机型不同会有不同打印最高上限，最高为每秒 15 吋打印速度)
density	int	设定打印浓度，0~15，数字越大打印结果越黑
sensor	int	设定使用传感器之类别： 0: 表示使用间隙传感器(gap sensor) 1: 表示使用黑标传感器(black mark sensor)
sensorDistance	int	设定间隙/黑标垂直间距高度，单位 mm
sensorOffset	int	设定间隙/黑标垂直间距高度，单位 mm，此参数若使用一般标签时均设为 0
context	Context	带入当前画面的 Context

6. GTSPL_setup(width, height, speed, density, sensor, sensorDistance, sensorOffset, context)

- 函式说明：设定卷标的宽度、高度、打印速度、打印热度、传感器类别、间隙/黑标垂直间距、间隙/黑标偏移距离

- 参数说明：

参数	型别	说明
width	Double	设定卷标宽度，单位 mm
height	Double	设定卷标高度，单位 mm
speed	int	设定打印速度，1~15，代表每秒 1~15 吋打印速度(随机型不同会有不同打印最高上限，最高为每秒 15 吋打印速度)
density	int	设定打印浓度，0~15，数字越大打印结果越黑
sensor	int	设定使用传感器之类别； 0：表示使用间隙传感器(gap sensor) 1：表示使用黑标传感器(black mark sensor)
sensorDistance	Double	设定间隙/黑标垂直间距高度，单位 mm
sensorOffset	Double	设定间隙/黑标垂直间距高度，单位 mm，此参数若使用一般标签时均设为 0
context	Context	带入当前画面的 Context

7. GTSPL_setDirectionAndMirror(direction,mirror, context)

- 函式说明：设定标签打印时的出纸方向与是否使用镜像打印

- 参数说明：

参数	型别	说明
direction	int	设定出纸方向，预设为 0 0：顶端出纸 1：底端出纸
mirror	int	设定是否镜像打印 0：否 1：是
context	Context	带入当前画面的 Context

8. GTSPL_setShift(shiftY, context)

- 函式说明：设定图像垂直位移距离，数值为正时，图像会往打印方向移动，数值为负时，图像会背离打印方向
- 参数说明：
 - ➔ shiftY: int 型别，垂直位移距离，单位为 dot
 - ➔ context: Context 类别，带入当前画面的 Context

9. GTSPL_printReverse(x_start, y_start, x_width, y_height, context)

- 函式说明：将指定的区域于打印时反白
- 参数说明：

参数	型别	说明
x_start	int	指定 X 起始坐标位置，以点(dot)表示
y_start	int	指定 Y 起始坐标位置，以点(dot)表示
x_width	int	指定 X 坐标宽度，以点(dot)表示
y_height	int	指定 Y 坐标高度，以点(dot)表示
context	Context	带入当前画面的 Context

10.GTSPL_setOffset(offset, context)

- 函式说明：设定每次出纸后额外偏移的距离(通常与剥纸模式和裁切模式组合使用)
- 参数说明：
 - ➔ offset: double 型别，额外的出纸偏移，单位为 mm
 - ➔ context: Context 类别，带入当前画面的 Context

11.GTSPL_setCutMode(mode, piece, context)

- 函式说明：设定裁切模式与张数
- 参数说明：

参数	型别	说明
mode	int	设定裁切方式，预设 为 1 0: 反切 1: 正切
piece	int	设定裁切张数
context	Context	带入当前画面的 Context

12.GTSPL_setAfterPrintAction(mode, context)

- 函式说明：设定打印后动作
- 参数说明：

参数	型别	说明
mode	int	设定打印后动作，预设值为 1 0：停在原地 1：撕纸 2：剥纸 3：裁切
context	Context	带入当前画面的 Context

13.GTSPL_genericDefault (context)

- 函式说明：将打印机之一般设定值初始化
- 参数说明：
 - ➔ context：Context 类别，带入当前画面的 Context

14.GTSPL_sensorDefault (context)

- 函式说明：将打印机之传感器设定值初始化
- 参数说明：
 - ➔ context：Context 类别，带入当前画面的 Context

15. GTSPL_clearBuffer(context)

- 函式说明：清除图像缓冲
- 参数说明：
 - ➔ context：Context 类别，带入当前画面的 Context

16. GTSPL_barcode(x, y, type, height, readable, rotation, narrow, wide, content, context)

- 函式说明：使用打印机内建条形码打印
- 参数说明：

参数	型别	说明
x	int	条形码 x 方向起始点，以点(dot)表示

y	int	条形码 Y 方向起始点，以点(dot)表示
type	字符串	设定条形码类型(Code Type) ， 请参考附件一
height	int	设定条形码高度，高度以点来表示
readable	int	设定是否打印条形码码文 0:不打印 1:打印条形码码文置左 2:打印条形码码文置中 3:打印条形码码文置右
rotation	int	设定条形码旋转角度 0： 旋转0度 90： 旋转90度 180： 旋转180度 270： 旋转270度
narrow	int	设定条形码窄 bar 比例因子， 请参考附件一
wide	int	设定条形码宽 bar 比例因子， 请参考附件一
content	字符串	设定欲打印之条形码内容
context	Context	带入当前画面的 Context

17. GTSPL_formFeed(context)

- 函式说明：跳页，该函式需在 setup 后使用
- 参数说明：
 - ➔ context: Context 类别，带入当前画面的 Context

18. GTSPL_noBackFeed(context)

- 函式说明：设定纸张不回吐
- 参数说明：
 - ➔ context: Context 类别，带入当前画面的 Context

19. GTSPL_sendCommand (context, command)

- 函式说明：送内建指令到打印机
- 参数说明：
 - ➔ command: 字符串型别，设定指令内容，详细指令请参考 TSPL 使用手册
 - ➔ context: Context 类别，带入当前画面的 Context

20. GTSPL_printerFont(x, y, size, rotation, x_scale, y_scale, content, context)

- 函式说明：使用打印机内建文字打印
- 参数说明：

参数	型别	说明
x	int	文字 X 方向起始点，以点(dot)表示
y	int	文字 Y 方向起始点，以点(dot)表示
size	字符串	内建字型名称，共五种 1: 8*/12 dots 2: 12*20 dots 3: 16*24 dots 4: 24*32 dots 5: 32*48 dots TST24.BF2: 繁体中文24*24 TST16.BF2: 繁体中文16*16 TSS24.BF2: 简体中文24*24 TSS16.BF2: 简体中文16*16
rotation	int	设定文字旋转角度 0: 旋转0度 90: 旋转90度 180: 旋转180度 270: 旋转 270 度
x_scale	int	设定文字 X 方向放大倍率，1~10
y_scale	int	设定文字Y方向放大倍率，1~10
content	字符串	设定欲打印之文字内容
context	Context	带入当前画面的 Context

21.GTSPL_qrcode(x, y, size, ECCLevel, cellWidth, mode, rotation, content)

■ 函式说明：使用打印机打印 QRcode

■ 参数说明：

参数	型别	说明
x	int	QRCode X 方向起始点，以点(dot)表示
y	int	QRCode Y 方向起始点，以点(dot)表示
ECCLevel	字符串	容错率 L : 7% M : 15% Q : 25% H : 30%
cellWidth	int	设定 QRCode 大小，1~10
mode	int	设定自动或手动编码 A: 自动 M: 手动
rotation	int	设定QRCode旋转角度 0: 旋转0度 90: 旋转90度 180: 旋转180度 270: 旋转270度
content	字符串	设定数据内容 数据内容限制： 1) 数字数据: (数字 0~9) 2) 字母数据 数字 0-9 大写字母 A-Z 9 种其它字符: 空格, \$ % * + - . / :) *如果” A”是数据内容的第一个字符，那么数据内容将会被设置为字母数据。 *如果” N”是数据内容的第一个字符，那么数据内容将会被设置为数字数据。

		* “!” 用来转换数据的格式,” N”、“ A”等数据类型可通过”!” 来转换。
context	Context	带入当前画面的 Context

22. GTSPL_printLabel(set, copy, context)

- 函式说明：打印标签内容
- 参数说明：
 - ➔ set: int 型别，设定打印标签式数(set)
 - ➔ copy: int 型别，设定打印标签份数(copy)
 - ➔ context: Context 类别，带入当前画面的 Context

23. GTSPL_downloadPCX(filename, context)

- 函式说明：下载单色 PCX 格式图文件至打印机
- 参数说明：
 - ➔ filename: 字符串型别，文件名
(档案需存在内部储存空间/android/data/packageName/files 文件夹下)
 - ➔ context: Context 类别，带入当前画面的 Context

24. GTSPL_downloadBMP(filename, context)

- 函式说明：下载单色 BMP 格式图文件至打印机
- 参数说明：
 - ➔ filename: 字符串型别，文件名
(档案需存在内部储存空间/android/data/packageName/files 文件夹下)
 - ➔ context: Context 类别，带入当前画面的 Context

25. GTSPL_download_Not1BitDepthBMP (filename, context)

■ 函式说明：将非单色之 BMP 格式图档转档后下载至打印机

■ 参数说明：

➔ filename：字符串型别，文件名

(档案需存在内部储存空间/android/data/packageName/files 文件夹下)

➔ context：Context 类别，带入当前画面的 Context

26. GTSPL_downloadTTF(filename, context)

■ 函式说明：下载 True Type Font 字型至打印机

■ 参数说明：

➔ filename：字符串型别，文件名

(档案需存在内部储存空间/android/data/packageName/files 文件夹下)

➔ context：Context 类别，带入当前画面的 Context

27. GTSPL_printerStatus(delaytime)

■ 函式说明：回传打印机状态，需用字符串变量接收回传讯息

■ 参数说明：

➔ delaytime：int 型别，设定延迟时间

■ 回传字符串说明：

回传字符串	打印机状态
00	就绪
01	上盖开启
02	卡纸
03	卡纸且上盖开启
04	标签用尽
05	标签用尽且上盖开启
08	碳带用尽
09	碳带用尽且上盖开启

0A	碳带用尽且卡纸
0B	碳带用尽、卡纸且上盖开启
0C	碳带用尽且标签用尽
0D	碳带用尽、标签用尽且上盖开启
10	暂停
20	打印中
80	其他错误

28. GTSPL_getSDKVersion (returnWay, context)

- 函式说明：回传此 SDK 版本号
- 参数说明：
 - ➔ retrunWay: int 型别，输入 0 除返回 SDK 版本号外，会跳出 SDK 版本讯息
 - ➔ context: Context 类别，带入当前画面的 Context

29. GTSPL_writeUHF (dataFormat,startBlockNo,byteSize,Gen2MemoryBank,datastring, context)

- 函式说明：将数据写入 UHF 卷标内存中
- 参数说明：

参数	型别	说明
dataFormat	string	设定字符串数据编码格式，默认为 H A: ASCII H: Hexadecimal
startBlockNo	int	设定数据区块起始位置，默认为 2
byteSize	int	设定写入数据byte长度，默认为1
Gen2Memory Bank	string	设定 Gen2 数据区段，默认为 E R: 保留 E: EPC T: TID(Tag ID) U: User
datastring	string	欲写入之字符串数据
context	Context	带入当前画面的 Context

30. GTSPL_EPCPWD_Action(action, password, context)

- 函式说明：将 UHF GNE2 的 EPC 资料区块上锁或解锁
- 参数说明：

参数	型别	说明
action	string	设定执行动作 U: 解锁资料区块 L: 上锁资料区块 O: 永久解锁资料区块 P: 永久上锁资料区块
password	string	密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

31. GTSPL_TIDPWD_Action(action, password, context)

- 函式说明：将 UHF GNE2 的 TID 资料区块上锁或解锁
- 参数说明：

参数	型别	说明
action	string	设定执行动作 U: 解锁资料区块 L: 上锁资料区块 O: 永久解锁资料区块 P: 永久上锁资料区块
password	string	密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

32. GTSPL_USERPWD_Action(action, password, context)

- 函式说明：将 UHF GNE2 的 USER 资料区块上锁或解锁
- 参数说明：

参数	型别	说明
action	string	设定执行动作 U: 解锁资料区块 L: 上锁资料区块 O: 永久解锁资料区块

		P: 永久上锁资料区块
password	string	密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

33. GTSPL_AccessPWD_Action (action, password, context)

- 函式说明：将 UHF GNE2 的存取密码进行设定、上锁或解锁
- 参数说明：

参数	型别	说明
action	string	设定执行动作 U: 解锁存取密码 L: 上锁存取密码 O: 永久解锁存取密码 P: 永久上锁存取密码 S: 设定存取密码
password	string	密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

34. GTSPL_KillPWD_Action (action, password, context)

- 函式说明：将 UHF GNE2 的删除密码进行设定、上锁或解锁
- 参数说明：

参数	型别	说明
action	string	设定执行动作 U: 解锁删除密码 L: 上锁删除密码 O: 永久删除存取密码 P: 永久删除存取密码 S: 设定删除密码
password	string	密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

35. GTSPL_Set_RFIDPorcedure (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times, context)

■ 函式说明：RFID 设定

■ 参数说明：

参数	型别	说明
tagType	int	设定卷标类型，1~10，默认值为 8 1：EPC Class 1 Generation 2-Q，8：EPC Class 1 Generation 2-R，10：UHF-J
rw_position	int	设卷标读写位置(卷标顶部起算)，范围为 0~9999(dot)，预设 0
void_printout	int	设定无效打印长度(dot)，范围为0~卷标长度，默认为卷标长度
tryEncodie_times	string	设定最大无效标签数，范围为 0~10，预设 3
error_handle	string	设定无效时采取的动作，预设 N N: No action(继续) P: Pause mode(暂停) E: Error mode(停止)
speed	int	设定无效打印速度，范围 2~10(IPS)，默认值 2(IPS)
retry_times	int	设定标签重试次数，范围 0~10，默认值 6
context	Context	带入当前画面的 Context

36. GTSPL_Set_RFIDPorcedure (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times,dpi, context)

■ 函式说明：RFID 设定

■ 参数说明：

参数	型别	说明
tagType	int	设定卷标类型，1~10，默认值为 8 1：EPC Class 1 Generation 2-Q，8：EPC Class 1 Generation 2-R，10：UHF-J
rw_position	int	设卷标读写位置(卷标顶部起算)，范围 203dpi:0 ~ 1251 (mm)、300dpi:0 ~ 846 (mm)、600dpi:0 ~ 423 (mm)，预设 0
void_printout	int	设定无效打印长度(dot)，范围为0~卷标长度，默认为卷标长度
tryEncodie_ti	string	设定最大无效标签数，范围为 0~10，预设 3

mes		
error_handle	string	设定无效时采取的动作，预设为 N N: No action(继续) P: Pause mode(暂停) E: Error mode(停止)
speed	int	设定无效打印速度，范围 2~10(IPS)，默认值 2(IPS)
retry_times	int	设定标签重试次数，范围 0~10，默认值 6
dpi	String	设定打印机的 DPI 203: 203 dpi 300: 300 dpi 600: 600 dpi
context	Context	带入当前画面的 Context

37. GTSPL_writeHF (dataFormat,startBlockNo,byteSize,datastring, context)

- 函式说明：将数据写入 HF 卷标内存中
- 参数说明：

参数	型别	说明
dataFormat	string	设定字符串数据编码格式，默认为 H A: ASCII H: Hexadecimal
startBlockNo	int	设定数据区块起始位置，默认为 2
byteSize	int	设定写入数据byte长度，默认为1
datastring	string	欲写入之字符串数据
context	Context	带入当前画面的 Context

38. GTSPL_printerFontBlock (x, y, width, height, fontname, rotation, x_scale, y_scale, space, align, content, context)

- 函式说明：打印段落文字内容
- 参数说明：

参数	型别	说明
x	string	文字 X 方向起始点，以点(dot)表示
y	string	文字 Y 方向起始点，以点(dot)表示
width	string	设定段落区块宽度，以点(dot)表示

height	string	设定段落区块高度，以点(dot)表示
fontname	string	内建字型名称 1: 8*12 dots 2: 12*20 dots 3: 16*24 dots 4: 24*32 dots 5: 32*48 dots TST24.BF2: 繁体中文24*24 TST16.BF2: 繁体中文16*16 TSS24.BF2: 简体中文24*24 TSS16.BF2: 简体中文16*16
rotation	string	设定文字旋转角度 0: 旋转0度 90: 旋转90度 180: 旋转180度 270: 旋转 270 度
x_scale	string	设定文字 X 方向放大倍率，1~10
y_scale	string	设定文字Y方向放大倍率，1~10
space	string	行距，以点(dot)表示
align	string	对齐位置 0: 预设(置左) 1: 置左 2: 置中 3: 置右
content	string	设定欲打印之文字内容
context	Context	带入当前画面的 Context

39. GTSPL_readUHF(dataFormat,startBlockNo,byteSize,Gen2MemoryBank,context)

■ 函式说明：读取 UHF 卷标内存数据

■ 参数说明：

参数	型别	说明
dataFormat	string	设定字符串数据编码格式，默认为 H A: ASCII H: Hexadecimal
startBlockNo	int	设定数据区块起始位置，默认为 0
byteSize	int	设定读取数据byte长度，默认为1

Gen2Memory Bank	string	设定 Gen2 数据区段，默认为 E R: 保留 E: EPC T: TID(Tag ID) U: User
context	Context	带入当前画面的 Context

■ 回传字符串说明:

dataFormat	回传字符串(范例)
A	卷标数据以 ASCII 显示 (ex: 24051324000103456400)
H	卷标数据以 Hexadecimal 显示 (ex: 3234303531333234303030313033343536343030)
*发生错误回传错误代码，错误代码说明 请参考附件二 ，UHFReaderE710 请参考附件三	

40. GTSPL_readUHF(dataFormat,startBlockNo,byteSize,Gen2MemoryBank,delaytime,context)

■ 函式说明: 读取 UHF 卷标内存数据(目前 delaytime 只支持 USB)

■ 参数说明:

参数	型别	说明
dataFormat	string	设定字符串数据编码格式，默认为 H A: ASCII H: Hexadecimal
startBlockNo	int	设定数据区块起始位置，默认为 0
byteSize	int	设定读取数据byte长度，默认为1
Gen2Memory Bank	string	设定 Gen2 数据区段，默认为 E R: 保留 E: EPC T: TID(Tag ID) U: User
delaytime	int	设定读取的延迟时间
context	Context	带入当前画面的 Context

■ 回传字符串说明:

dataFormat	回传字符串(范例)
A	卷标数据以 ASCII 显示 (ex: 24051324000103456400)

H	卷标数据以 Hexadecimal 显示 (ex: 3234303531333234303030313033343536343030)
*发生错误回传错误代码，错误代码说明 请参考附件二 ，UHFReaderE710 请参考附件三	

41. GTSPL_readUHFQ(dataFormat, PCReturnStatus, CRCReturnStatus,context)

■ 函式说明：读取 UHF 卷标内存数据

■ 参数说明：

参数	型别	说明
dataFormat	string	设定字符串数据编码格式，默认为 H A: ASCII H: Hexadecimal
PCReturnStatus	int	PC 返回状态，预设为 0 0: 不回传 PC 值 1: 回传 PC 值
CRCReturnStatus	int	CRC-16 返回状态，预设为 0 0: 不回传 CRC-16 1: 回传CRC-16
context	Context	带入当前画面的 Context

■ 回传字符串说明：

dataFormat	回传字符串(范例)
A	卷标数据以 ASCII 显示 (ex: 24051324000103456400)
H	卷标数据以 Hexadecimal 显示 (ex: 3234303531333234303030313033343536343030)
*发生错误回传错误代码，错误代码说明 请参考附件二 ，UHFReaderE710 请参考附件三	

42. GTSPL_LabelCalibration (Context context)

■ 函式说明：执行 RFID Tag 校准动作

■ 参数说明：

➔ context: Context 类别，带入当前画面的 Context

43. GTSPL_rfidSetupDefault (Context context)

- 函式说明：将 RFID 设定值初始化
- 参数说明：

➔ context: Context 类别，带入当前画面的 Context

44. GTSPL_writeGJB(String dataFormat, int startBlockNo, int byteSize, String GJBMemoryBank, String datastring, String writePWD, Context context)

- 函式说明：将数据写入 UHF GJB 卷标内存中
- 参数说明：

参数	型别	说明
dataFormat	string	设定字符串数据编码格式，默认为 H A: ASCII H: Hexadecimal
startBlockNo	int	设定数据区块起始位置，GJB 默认为 1
byteSize	int	设定写入数据byte长度，默认为1
Gen2MemoryBank	string	设定 GJB 数据区段，默认为 E R: 安全区 E: EPC T: TID(Tag ID) U: User
datastring	string	欲写入之字符串数据
writePWD	string	写入密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

45. GTSPL_readGJB(String dataFormat, int startBlockNo, int byteSize, String GJBMemoryBank, String ReadPWD, Context context)

- 函式说明：读取 UHF GJB 卷标内存数据，需用字符串变量接收回传讯息
- 参数说明：

参数	型别	说明
dataFormat	string	设定字符串数据编码格式，默认为 H

		A: ASCII H: Hexadecimal
startBlockNo	int	设定数据区块起始位置，默认为 0
byteSize	int	设定读取数据byte长度，默认为1
Gen2MemoryBank	string	设定 GJB 数据区段，默认为 E R: 安全区 E: EPC T: TID(Tag ID) U: User
ReadPWD	string	读取密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

■ 回传字符串说明：

dataFormat	回传字符串(范例)
A	卷标数据以 ASCII 显示 (ex: 24051324000103456400)
H	卷标数据以 Hexadecimal 显示 (ex: 3234303531333234303030313033343536343030)

46. GTSPL_Set_GJB_Pwd(String pwdArea, String action, String pwdSet, String writePWD, Context context)

■ 函式说明：设定 UHF GJB 各密码区之新密码

■ 参数说明：

参数	型别	说明
pwdArea	string	设定密码区域，默认为 W K: Kill 删除 W: Write 写入 R: Read 读取 S: Status 状态
action	string	设定动作 S: Set Password
pwdSet	string	设定密码区域的新密码，应为8 hex字符(0~9,A,B,C,D,E,F)
writePWD	string	目前之写入密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

47. GTSPL_Set_GJB_Status(String GJBMemoryBank, String action, String statusPWD, Context context)

- 函式说明：设定 UHF GJB 各数据区块读写状态
- 参数说明：

参数	型别	说明
Gen2MemoryBank	string	设定 GJB 数据区段，默认为 E F: 安全区 E: EPC T: TID(Tag ID) U: User
action	string	设定状态，预设为 A A=Lock0(可读可写) B=Lock1(可读不可写) C=Lock2(不可读可写) D=Lock3(不可读不可写)
statusPWD	string	状态密码，应为8 hex字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的Context

48. GTSPL_Kill_GJB_Tag(String kill_PWD, Context context)

- 函式说明：删除 UHF GJB 标签
- 参数说明：

参数	型别	说明
kill_PWD	string	删除密码，应为 8 hex 字符(0~9,A,B,C,D,E,F)
context	Context	带入当前画面的 Context

49. GTSPL_WifiFrequency (String Frequency, Context context)

- 函式说明：使用兼容 5G 频段 WIFI 模块时，可用于切换使用频段
- 参数说明：

参数	型别	说明
Frequency	string	设定模块频段 2.4G: 使用 2.4G 频段 5G: 使用 5G 频段 BOTH: 使用双频频段
context	Context	带入当前画面的 Context

50. GTSPL_printBMP (String x, String y, int width, int height, int mode, String filename, Context context)

- 函式说明：将图片转为单色点阵图，使用打印机直接打印
- 参数说明：

参数	型別	说明
x	string	文字 X 方向起始点，以点(dot)表示
y	string	文字 Y 方向起始点，以点(dot)表示
width	int	图片宽度，以字节(byte)表示
height	int	图片高度，以点(dot)表示
mode	int	图片格式 0: OVERWRITE 1: OR 2: XOR
filename	string	档案名称 (档案需存在内部储存空间/android/data/packageName/files文件夹下) 图档仅支援以下格式： 1. BMP (Bitmap)：位图格式 2. JPG (JPEG)：压缩的图像格式 3. PNG (Portable Network Graphics)：无损压缩的图像格式 4. GIF (Graphics Interchange Format)：支援多张图片的格式，通常用于动画 5. TIFF (Tagged Image File Format)：高品质的无损压缩图像格式 6. ICO (Icon)：图示格式，用于显示档案、程式或资料夹的图示 7. WMF (Windows Metafile)：Windows 绘图文件格式 8. EMF (Enhanced Metafile)：扩展的 Windows 绘图文件格式
context	Context	带入当前画面的Context

51. GTSPL_printBMP_Compression (String x, String y, int width, int height, String filename, Context context)

- 函式说明：将图片转为单色点阵图，压缩后再使用打印机打印
- 参数说明：

参数	型別	说明
x	string	文字 X 方向起始点，以点(dot)表示
y	string	文字 Y 方向起始点，以点(dot)表示

width	int	图片宽度，以字节(byte)表示
height	int	图片高度，以点(dot)表示
filename	string	档案名称 (档案需存在内部储存空间/ <code>android/data/packageName/files</code> 文件夹下) 图档仅支援以下格式： 1. BMP (Bitmap)：位图格式 2. JPG (JPEG)：压缩的图像格式 3. PNG (Portable Network Graphics)：无损压缩的图像格式 4. GIF (Graphics Interchange Format)：支援多张图片的格式，通常用于动画 5. TIFF (Tagged Image File Format)：高品质的无损压缩图像格式 6. ICO (Icon)：图示格式，用于显示档案、程式或资料夹的图示 7. WMF (Windows Metafile)：Windows 绘图文件格式 8. EMF (Enhanced Metafile)：扩展的 Windows 绘图文件格式
context	Context	带入当前画面的 Context

52. GTSPL_autoConnect ((MacAddress, isManual)

- 函式说明：非手动关闭的情况下，7 秒过后指定蓝芽的 MAC 地址(请使用 BR/EDR MacAddress)，开启输出埠(目前只支援 BT 使用)
- 只支持 GE2408DE1168 的机种
- 参数说明：
 - ➔ **MacAddress**：字符串型别，指定联机的蓝芽地址(Bluetooth MacAddress)，
如： ” DC:1D:30:00:1D:87”
 - ➔ **isManual**：Boolean 型别，确认是否为手动关闭

53. GTSPL_labelCalibration(width, height, sensor, sensorDistance, tagType, context)

- 函式说明：设定卷标的宽度、高度、传感器类别、间隙/黑标垂直间距、卷标类型，设定完成后执行 RFID Tag 校准动作
- 只支持 GE2408DE1168 的机种
- 参数说明：

参数	型别	说明
width	Double	设定卷标宽度，单位 mm
height	Double	设定卷标高度，单位 mm

sensor	int	设定使用传感器之类别; 0：表示使用间隙传感器(gap sensor) 1：表示使用黑标传感器(black mark sensor)
sensorDistance	Double	设定间隙/黑标垂直间距高度，单位 mm
tagType	int	设定卷标类型，1~10，默认值为 8 1：EPC Class 1 Generation 2-Q , 8：EPC Class 1 Generation 2-R ,10：UHF-J
context	Context	带入当前画面的 Context

54. GTSPL_setBicolor(color, density, context)

- 函式说明：设定双色指令
- 仅支持部分机种，详情请洽代理商
- 参数说明：

参数	型別	说明
color	string	设定颜色 R：紅色 B：黑色
density	int	设定打印浓度，0~15，数字越大打印结果越深(color 為黑色時設定為 0)
context	Context	带入当前画面的 Context

Android Bluetooth 范例说明

1.在 build.gradle(Module)中的 dependencies{}选择要使用 jar 檔或是 aar 檔

启用 jar 檔:

```
implementation fileTree(dir: 'libs', include: ['*.jar'])
```

启用 aar 檔:

```
implementation fileTree(dir: 'libs', include: ['*.aar'])
```

2.需先于 AndroidManifest.xml 设定下列权限:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

3.汇入 GTSPL_SDK:

```
import com.gto.gtspl_sdk.GTSPLActivity;
```

4.范例程序:

```
public class MainActivity extends Activity {  
  
    GTSPLActivity mGtsplCmdTest = new GTSPLActivity();  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.activity_main);  
  
        mGtsplCmdTest.GTSPL_setCmdSendMode ("P");  
  
        mGtsplCmdTest.GTSPL_openPort ("DC:1D:30:00:1D:87");  
  
        mGtsplCmdTest.GTSPL_autoConnect("DC:1D:30:00:1D:87", false);  
  
        mGtsplCmdTest.GTSPL_setup(62, 45, 2, 6, 0, 3, 0, this);  
  
        mGtsplCmdTest.GTSPL_setup(62.0, 45.0, 2, 6, 0, 3.0, 0.0, this);  
  
        mGtsplCmdTest.GTSPL_sendCommand(this, "DIRECTION 1\n\n");  
  
        mGtsplCmdTest.GTSPL_clearBuffer(this);  
  
        mGtsplCmdTest.GTSPL_printerFont(100, 100, "3", 0, 1, 1, "Print Font 123456", this);  
  
        mGtsplCmdTest.GTSPL_barcode(30, 30, "128", 100, 1, 0, 2, 2, "barcode1234567", this);  
  
        mGtsplCmdTest.GTSPL_qrcode(300, 100, "H", 4, "A", 0, "ABCabc123", this);  
    }  
}
```

```

mGtsplCmdTest.GTSPL_downloadBMP("CIRCLE.BMP", this);

mGtsplCmdTest.GTSPL_sendCommand(this, "PUTBMP 150,30,\"CIRCLE.BMP\"\\r\\n");

mGtsplCmdTest.GTSPL_download_Not1BitDepthBMP("printTest4.BMP", MainActivity.this);

mGtsplCmdTest.GTSPL_sendCommand(MainActivity.this, "PUTBMP

10,10,\"PrintTest4.BMP\"\\r\\n");

String sBlock = "We stand behind our products with one of the most comprehensive support
programs in the Auto-ID industry.";

mGtsplCmdTest.GTSPL_printFontBlock("15", "15", "790", "90", "0", "0", "8", "8", "20",
"2",sBlock,this);

mGtsplCmdTest.GTSPL_printLabel(1, 1, this);


//初始化

mGtsplCmdTest.GTSPL_genericDefault(); //一般设定初始化
mGtsplCmdTest.GTSPL_sensorDefault(); //传感器初始化
mGtsplCmdTest.GTSPL_rfidSetupDefault(); //rfid 设定初始化


//修改 Wifi 频段

mGtsplCmdTest.GTSPL_WifiFrequency("5G");


//打印设定功能(需搭配打印功能执行)

mGtsplCmdTest.GTSPL_setDirectionAndMirror(1, 0); //设定打印方向与镜像
mGtsplCmdTest.GTSPL_setShift(50); //设定垂直偏移
mGtsplCmdTest.GTSPL_printReverse(10, 10, 160, 160); //设定反白
mGtsplCmdTest.GTSPL_setOffset(20); //设定出纸偏移
mGtsplCmdTest.GTSPL_setCutMode(0, 2); //设定切刀模式与张数
mGtsplCmdTest.GTSPL_setAfterPrintAction(2); //设定打印后动作
mGtsplCmdTest.GTSPL_printLabel(1, 1);

```

//Bitmap 打印

```
mGtsplCmdTest.GTSPL_printBMP(-500,30,400,300,1,"CIRCLE.BMP ",this);
```

```
mGtsplCmdTest.GTSPL_printBMP_Compression(-500,30,400,300," CIRCLE.BMP ",this);
```

```
mGtsplCmdTest.GTSPL_printLabel(1, 1);
```

//双色打印

```
mGtsplCmdTest.GTSPL_setBicolor("R",7, this);
```

//GEN2 RFID

```
mGtsplCmdTest.GTSPL_writeUHF("H", 2, 12, "E", "414142424343444445454646", this);
```

```
mGtsplCmdTest.GTSPL_GTSPL_EPCPWD_Action("U", "12345678", this);
```

```
mGtsplCmdTest.GTSPL_GTSPL_TIDPWD_Action("L", "12345678", this);
```

```
mGtsplCmdTest.GTSPL_USERPWD_Action("L", "12345678", this);
```

```
mGtsplCmdTest.GTSPL_AccessPWD_Action("S", "12345678", this);
```

```
mGtsplCmdTest.GTSPL_KillPWD_Action("S", "12345678", this);
```

```
mGtsplCmdTest.GTSPL_Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2, this);
```

```
mGtsplCmdTest.GTSPL_Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2, "203", this);
```

```
mGtsplCmdTest.GTSPL_writeHF("H", 2, 12, "414142424343444445454646", this);
```

```
mGtsplCmdTest.GTSPL_printLabel(1, 1, this);
```

```
String uhfData = mGtsplCmdTest.GTSPL_readUHF("H",2,12,"E", this);
```

```
String uhfData = mGtsplCmdTest.GTSPL_readUHFQ("H",0,0, this);
```

//GJB RFID 设定密码

//带入写入密码，设定新的读取密码

```
mGtsplCmdTest.GTSPL_Set_GJB_Pwd("R","S","87654321","12345678",this);
```

//带入写入密码，设定新的写入密码

```

mGtsplCmdTest.GTSPL_Set_GJB_Pwd("W","S","87654321","12345678",this);

//带入写入密码，设定新的删除密码

mGtsplCmdTest.GTSPL_Set_GJB_Pwd("K","S","87654321","12345678",this);

//带入写入密码，设定新的状态密码

mGtsplCmdTest.GTSPL_Set_GJB_Pwd("S","S","87654321","12345678",this);

mGtsplCmdTest.GTSPL_printLabel(1, 1, this);


//GJB RFID 设定不同数据区块的写入读取状态

mGtsplCmdTest.GTSPL_Set_GJB_Status("E","C","11112222",this);

mGtsplCmdTest.GTSPL_printLabel(1, 1, this);


//GJB RFID 写入 EPC 资料

mGtsplCmdTest.GTSPL_writeGJB("H",2,12,"E","404041414242434344444545","12345678",this);

mGtsplCmdTest.GTSPL_printLabel(1, 1, this);


//GJB RFID 读取 EPC 数据

String GJBData=mGtsplCmdTest.GTSPL_readGJB("H",2,12,"E","33334444",MainActivity.this);


//GJB RFID 删除标签

mGtsplCmdTest.GTSPL_Kill_GJB_Tag("11224455",this);

mGtsplCmdTest.GTSPL_printLabel(1,1,this);


//简中打印

String stString="默认简体中文测试";

mGtsplCmdTest.GTSPL_clearBuffer(this);

mGtsplCmdTest.GTSPL_printerFont(100, 10, "TSS24.BF2", 0, 1, 1, stString, this);

```

```

mGtsplCmdTest.GTSPL_printLabel(1, 1, this);

//繁中打印
String ttString="默認繁體中文測試";

mGtsplCmdTest.GTSPL_clearBuffer(this);

mGtsplCmdTest.GTSPL_printerFont(100, 10, " TST24.BF2", 0, 1, 1, ttString, this);

mGtsplCmdTest.GTSPL_printLabel(1, 1, this);

String status = mGtsplCmdTest.GTSPL_printersStatus(1000);

mGtsplCmdTest.GTSPL_closePort(1000);

String version= mGtsplCmdTest .GTSPL_getSDKVersion(0,this);


//RFID 自动校准

mGtsplCmdTest.GTSPL_LabelCalibration(this);


//UHFReaderE710 自动校准

mGtsplCmdTest.GTSPL_labelCalibration(62.0, 45.0, 0, 3.0, 8, this)
}
}

```

Android Ethernet 范例说明

1.在 build.gradle(Module)中的 dependencies{}选择要使用 jar 檔或是 aar 檔

启用 jar 檔:

```
implementation fileTree(dir: 'libs', include: ['*.jar'])
```

启用 aar 檔:

```
implementation fileTree(dir: 'libs', include: ['*.aar'])
```

2.需先于 AndroidManifest.xml 设定下列权限:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

3.汇入 GTSP_L_SDK:

```
import com.gto.gtspl_sdk.GTSPLWIFIActivity;
```

4.范例程序:

```
public class MainActivity extends Activity {  
  
    GTSPLWIFIActivity mGtsplWIFICmdTest = new GTSPLWIFIActivity();  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.activity_main);  
  
        mGtsplWIFICmdTest.GTSPL_setCmdSendMode("P");  
  
        mGtsplWIFICmdTest.GTSPL_openPort("192.168.1.109",8899,5000);  
  
        mGtsplWIFICmdTest.GTSPL_setup(62, 45, 2, 6, 0, 3, 0, this);  
  
        mGtsplWIFICmdTest.GTSPL_setup(62.0, 45.0, 2, 6, 0, 3.0, 0.0, this);  
  
        mGtsplWIFICmdTest.GTSPL_sendCommand(this, "DIRECTION 1\n\n");  
  
        mGtsplWIFICmdTest.GTSPL_clearBuffer(this);  
  
        mGtsplWIFICmdTest.GTSPL_printerFont(100, 10, "5", 0, 1, 1, "Print Font 123456", this);  
  
        mGtsplWIFICmdTest.GTSPL_barcode(30, 30, "128", 100, 1, 0, 2, 2, "barcode1234567", this);  
  
        mGtsplWIFICmdTest.GTSPL_qrcode(300, 100, "H", 4, "A", 0, "ABCabc123", this);  
  
        mGtsplWIFICmdTest.GTSPL_downloadBMP("CIRCLE.BMP", this);  
  
        mGtsplWIFICmdTest.GTSPL_sendCommand(this, "PUTBMP 150,30,\"CIRCLE.BMP\"\\r\\n");  
    }  
}
```

```

mGtsplWIFICmdTest.GTSPL_download_Not1BitDepthBMP("printTest4.BMP",
MainActivity.this);

mGtsplWIFICmdTest.GTSPL_sendCommand(MainActivity.this, "PUTBMP
10,10,\"PrintTest4.BMP\"\\r\\n");

String sBlock = "We stand behind our products with one of the most comprehensive support
programs in the Auto-ID industry.";

mGtsplWIFICmdTest.GTSPL_printFontBlock("15", "15", "790", "90", "0", "0", "8", "8", "20",
"2",sBlock,this);

mGtsplWIFICmdTest.GTSPL_printLabel(1, 1, this);


//初始化
mGtsplWIFICmdTest.GTSPL_genericDefault(); //一般设定初始化
mGtsplWIFICmdTest.GTSPL_sensorDefault(); //传感器初始化
mGtsplWIFICmdTest.GTSPL_rfidSetupDefault(); //rfid 设定初始化


//修改 Wifi 频段
mGtsplWIFICmdTest.GTSPL_WifiFrequency("5G");


//打印设定功能(需搭配打印功能执行)
mGtsplWIFICmdTest.GTSPL_setDirectionAndMirror(1, 1); //设定打印方向与镜像
mGtsplWIFICmdTest.GTSPL_setShift(50); //设定垂直偏移
mGtsplWIFICmdTest.GTSPL_printReverse(10, 10, 160, 160); //设定反白
mGtsplWIFICmdTest.GTSPL_setOffset(20); //设定出纸偏移
mGtsplWIFICmdTest.GTSPL_setCutMode(0, 2); //设定切刀模式与张数
mGtsplWIFICmdTest.GTSPL_setAfterPrintAction(2); //设定打印后动作
mGtsplWIFICmdTest.GTSPL_printLabel(1, 1);

```

//Bitmap 打印

```
mGtsplWIFICmdTest.GTSPL_printBMP(-500,30,400,300,1,"CIRCLE.BMP ",this);  
mGtsplWIFICmdTest.GTSPL_printBMP_Compression(-500,30,400,300," CIRCLE.BMP ",this);  
mGtsplWIFICmdTest.GTSPL_printLabel(1, 1);
```

//双色打印

```
mGtsplWIFICmdTest.GTSPL_setBicolor("R",7, this);
```

//GEN2 RFID

```
mGtsplWIFICmdTest.GTSPL_writeUHF("H", 2, 12, "E", "414142424343444445454646", this);  
mGtsplWIFICmdTest.GTSPL_EPCPWD_Action("L", "12345678", this);  
mGtsplWIFICmdTest.GTSPL_TIDPWD_Action("L", "12345678", this);  
mGtsplWIFICmdTest.GTSPL_USERPWD_Action("L", "12345678", this);  
mGtsplWIFICmdTest.GTSPL_AccessPWD_Action("S", "12345678", this);  
mGtsplWIFICmdTest.GTSPL_KillPWD_Action("S", "12345678", this);  
mGtsplWIFICmdTest.GTSPL_Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2, this);  
mGtsplWIFICmdTest.GTSPL_Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2, "203", this);  
mGtsplWIFICmdTest.GTSPL_writeHF("H", 2, 12, "414142424343444445454646", this);  
mGtsplWIFICmdTest.GTSPL_printLabel(1, 1, this);  
String uhfData = mGtsplWIFICmdTest.GTSPL_readUHF("H",2,12,"E", this);  
String uhfData = mGtsplWIFICmdTest.GTSPL_readUHFQ("H",0,0,this);
```

//GJB RFID 设定密码

//带入写入密码，设定新的读取密码

```
mGtsplWIFICmdTest.GTSPL_Set_GJB_Pwd("R","S","87654321","12345678",this);
```

//带入写入密码，设定新的写入密码

```
mGtsplWIFICmdTest.GTSPL_Set_GJB_Pwd("W","S","87654321","12345678",this);
```



```

//带入写入密码，设定新的删除密码

mGtsplWIFICmdTest.GTSPL_Set_GJB_Pwd("K","S","87654321","12345678",this);

//带入写入密码，设定新的状态密码

mGtsplWIFICmdTest.GTSPL_Set_GJB_Pwd("S","S","87654321","12345678",this);

mGtsplWIFICmdTest.GTSPL_printLabel(1, 1, this);


//GJB RFID 设定不同数据区块的写入读取状态

mGtsplWIFICmdTest.GTSPL_Set_GJB_Status("E","C","11112222",this);

mGtsplWIFICmdTest.GTSPL_printLabel(1, 1, this);


//GJB RFID 写入 EPC 资料

mGtsplWIFICmdTest.GTSPL_writeGJB("H",2,12,"E","404041414242434344444545","123456
78",this);

mGtsplWIFICmdTest.GTSPL_printLabel(1, 1, this);


//GJB RFID 读取 EPC 数据

String GJBData =

mGtsplWIFICmdTest.GTSPL_readGJB("H",2,12,"E","33334444",MainActivity.this);


//GJB RFID 删除标签

mGtsplWIFICmdTest.GTSPL_Kill_GJB_Tag("11224455",this);

mGtsplWIFICmdTest.GTSPL_printLabel(1,1,this);


//简中打印

String stString="默认简体中文测试";

mGtsplWIFICmdTest.GTSPL_clearBuffer(this);

mGtsplWIFICmdTest.GTSPL_printerFont(100, 10, "TSS24.BF2", 0, 1, 1, stString, this);

```

```

mGtspIWIFICmdTest.GTSPL_printLabel(1, 1, this);

//繁中打印
String ttString="默認繁體中文測試";

mGtspIWIFICmdTest.GTSPL_clearBuffer(this);

mGtspIWIFICmdTest.GTSPL_printerFont(100, 10, " TST24.BF2", 0, 1, 1, ttString, this);

mGtspIWIFICmdTest.GTSPL_printLabel(1, 1, this);

String status = mGtspIWIFICmdTest.GTSPL_printersStatus(1000);

mGtspIWIFICmdTest.GTSPL_closePort();

String version=mGtspIWIFICmdTest.GTSPL_getSDKVersion(0,this);


//RFID 自动校准

mGtspIWIFICmdTest.GTSPL_LabelCalibration(this);


//UHFReaderE710 自动校准

mGtspIWIFICmdTest.GTSPL_labelCalibration(62.0, 45.0, 0, 3.0, 8, this)
}
}

```

Android USB 范例说明

1.在 `build.gradle(Module)`中的 `dependencies{}`选择要使用 `jar` 檔或是 `aar` 檔

启用 `jar` 檔:

```
implementation fileTree(dir: 'libs', include: ['*.jar'])
```

启用 `aar` 檔:

```
implementation fileTree(dir: 'libs', include: ['*.aar'])
```

2.汇入 `GTSP_L_SDK`:

```
import com.gto.gtspl_sdk.GTSPLUsbActivity;
```

3.范例程序:

```
public class MainActivity extends Activity {  
  
    GTSPLUsbActivity mUSB = new GTSPLUsbActivity();  
  
    private static final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";  
  
    private static UsbManager mUsbManager;  
  
    private static PendingIntent mPermissionIntent;  
  
    private static boolean hasPermissionToCommunicate = false;  
  
    private static UsbDevice mDevice;  
  
  
    private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {  
  
        public void onReceive(Context context, Intent intent) {  
  
            String action = intent.getAction();  
  
            if (ACTION_USB_PERMISSION.equals(action)) {  
  
                synchronized (this) {  
  
                    UsbDevice device = intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);  
  
                    if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {  
  
                        if (device != null) {hasPermissionToCommunicate = true;}}  
  
                }  
  
            }  
  
        }  
  
    }  
  
}
```

```

    }

};

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);

    mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(ACTION_USB_PERMISSION), 0);

    IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);

    registerReceiver(mUsbReceiver, filter);

    HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();

    Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();

    while (deviceIterator.hasNext()) {

        mDevice = deviceIterator.next();

        if (mDevice.getVendorId() == 1137) {break;}

    }

    mPermissionIntent = PendingIntent.getBroadcast(MainActivity.this, 0, new
Intent(ACTION_USB_PERMISSION), PendingIntent.FLAG_ONE_SHOT);

    mUsbManager.requestPermission(mDevice, mPermissionIntent);

    mUSB.GTSPL_setCmdSendMode("P");

    mUSB.GTSPL_openPort(mUsbManager, mDevice);

    mUSB.GTSPL_setup(62, 45, 2, 3, 0, 3, 0, this);

    mUSB.GTSPL_setup(62.0, 45.0, 2, 3, 0, 3.0, 0.0, this);

    mUSB.GTSPL_sendCommand(this, "DIRECTION 1\r\n");

    mUSB.GTSPL_clearBuffer(this);

    mUSB.GTSPL_barcode(30, 30, "128", 100, 1, 0, 2, 2, "barcode9463521", this);

```

```

mUSB.GTSPL_qrcode(300, 100, "H", 4, "A", 0, "ABCabc123", this);

mUSB.GTSPL_printerFont(100, 50, "2", 0, 1, 1, "PrintFontTest123", this);

mUSB.GTSPL_downloadBMP("LOGO.BMP", this);

mUSB.GTSPL_sendCommand(this, "PUTBMP 100,80,\"LOGO.BMP\"\\r\\n");

mUSB.GTSPL_download_Not1BitDepthBMP("printTest4.BMP", MainActivity.this);

mUSB.GTSPL_sendCommand(MainActivity.this, "PUTBMP10,10,\"PrintTest4.BMP\"\\r\\n");

String sBlock = "We stand behind our products with one of the most comprehensive
support programs in the Auto-ID industry.";

mUSB.GTSPL_printFontBlock("15", "15", "790", "90", "0", "0", "8", "8", "20",
"2",sBlock,this);

mUSB.GTSPL_printLabel(1, 1, this);


//初始化

mUSB.GTSPL_genericDefault(); //一般设定初始化
mUSB.GTSPL_sensorDefault(); //传感器初始化
mUSB.GTSPL_rfidSetupDefault(); //rfid 设定初始化


//修改 Wifi 频段

mUSB.GTSPL_WifiFrequency("5G");


//打印设定功能(需搭配打印功能执行)

mUSB.GTSPL_setDirectionAndMirror(1, 1); //设定打印方向与镜像
mUSB.GTSPL_setShift(50); //设定垂直偏移
mUSB.GTSPL_printReverse(10, 10, 160, 160); //设定反白
mUSB.GTSPL_setOffset(20); //设定出纸偏移
mUSB.GTSPL_setCutMode(0, 2); //设定切刀模式与张数
mUSB.GTSPL_setAfterPrintAction(2); //设定打印后动作

```

```
mUSB.GTSPL_printLabel(1, 1);
```

```
//Bitmap 打印
```

```
mUSB.GTSPL_printBMP(-500,30,400,300,1,"CIRCLE.BMP ",this);
```

```
mUSB.GTSPL_printBMP_Compression(-500,30,400,300," CIRCLE.BMP ",this);
```

```
mUSB.GTSPL_printLabel(1, 1);
```

```
//双色打印
```

```
mUSB.GTSPL_setBicolor("R",7, this);
```

```
//GEN2 RFID
```

```
mUSB.GTSPL_writeUHF("H", 2, 12, "E", "414142424343444445454646", this);
```

```
mUSB.GTSPL_EPCPWD_Action("L", "12345678", this);
```

```
mUSB.GTSPL_TIDPWD_Action("L", "12345678", this);
```

```
mUSB.GTSPL_USERPWD_Action("L", "12345678", this);
```

```
mUSB.GTSPL_AccessPWD_Action("S", "12345678", this);
```

```
mUSB. GTSPL_KillPWD_Action ("S", "12345678", this);
```

```
mUSB.GTSPL_Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2, this);
```

```
mUSB.GTSPL_Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2, "203", this);
```

```
mUSB.GTSPL_writeHF("H", 2, 12, "414142424343444445454646", this);
```

```
mUSB.GTSPL_printLabel(1, 1, this);
```

```
String uhfData = mUSB.GTSPL_readUHF("H",2,12,"E", this);
```

```
String uhfData = mUSB.GTSPL_readUHF("H",2,12,"E", 4000, this);
```

```
String uhfData = mUSB.GTSPL_readUHFQ("H",0,0,this);
```

```
//GJB RFID 设定密码
```

```
//带入写入密码，设定新的读取密码
```

```
mUSB.GTSPL_Set_GJB_Pwd("R","S","87654321","12345678",this);

//带入写入密码，设定新的写入密码

mUSB.GTSPL_Set_GJB_Pwd("W","S","87654321","12345678",this);

//带入写入密码，设定新的删除密码

mUSB.GTSPL_Set_GJB_Pwd("K","S","87654321","12345678",this);

//带入写入密码，设定新的状态密码

mUSB.GTSPL_Set_GJB_Pwd("S","S","87654321","12345678",this);

mUSB.GTSPL_printLabel(1, 1, this);


//GJB RFID 设定不同数据区块的写入读取状态

mUSB.GTSPL_Set_GJB_Status("E","C","11112222",this);

mUSB.GTSPL_printLabel(1, 1, this);


//GJB RFID 写入 EPC 资料

mUSB.GTSPL_writeGJB("H",1,12,"E","404041414242434344444545","12345678",this);

mUSB.GTSPL_printLabel(1, 1, this);


//GJB RFID 读取 EPC 数据

String GJBData = mUSB.GTSPL_readGJB("A",0,12,"E","33334444",MainActivity.this);


//GJB RFID 删除标签

mUSB.GTSPL_Kill_GJB_Tag("11224455",this);

mUSB.GTSPL_printLabel(1,1,this);


//简中打印

String stString="默认简体中文测试";

mUSB.GTSPL_clearBuffer(this);
```

```
mUSB.GTSPL_printerFont(100, 10, "TSS24.BF2", 0, 1, 1, stString, this);
```

```
mUSB.GTSPL_printLabel(1, 1, this);
```

```
//繁中打印
```

```
String ttString="默認繁體中文測試";
```

```
mUSB.GTSPL_clearBuffer(this);
```

```
mUSB.GTSPL_printerFont(100, 10, "TST24.BF2", 0, 1, 1, ttString, this);
```

```
mUSB.GTSPL_printLabel(1, 1, this);
```

```
String status = mUSB.GTSPL_printersStatus(1000);
```

```
mUSB.GTSPL_closePort();
```

```
String version= mUSB.GTSPL_getSDKVersion(0,this);
```

```
//RFID 自动校准
```

```
mUSB.GTSPL_LabelCalibration(this);
```

```
//UHFReaderE710 自动校准
```

```
mUSB.GTSPL_labelCalibration(62.0, 45.0, 0, 3.0, 8, this)
```

```
}
```


附件一

Code Type	Description	Narrow : Width					Max. data length
		1:1	1:2	1:3	2:5	3:7	
128	Code 128, switching code subset automatically.	V					
128M	Code 128, switching code subset manually.	V					
EAN128	EAN128, switching code subset automatically.	V					
EAN128M	EAN128M, switching code subset manually.	V					
25	Interleaved 2 of 5.		V	V	V		Length is even
25C	Interleaved 2 of 5 with check digit.		V	V	V		Length is odd
25S	Standard 2 of 5.		V	V	V		
25I	Industrial 2 of 5.		V	V	V		
39	Code 39, switching standard and full ASCII mode automatically.		V	V	V		
39C	Code 39 with check digit.		V	V	V		
93	Code 93.			V			
EAN13	EAN 13.	V					12
EAN13+2	EAN 13 with 2 digits add-on.	V					14
EAN13+5	EAN 13 with 5 digits add-on.	V					17
EANB	EAN 8.	V					7
EANB+2	EAN 8 with 2 digits add-on.	V					96
EANB+5	EAN 8 with 5 digits add-on.	V					12
CODA	Codabar.		V	V	V		
POST	Postnet.	V					5,9,11
UPCA	UPC-A.	V					11
UPCA+2	UPC-A with 2 digits add-on.	V					13
UPA+5	UPC-A with 5 digits add-on.	V					16
UPCE	UPC-E.	V					6
UPCE+2	UPC-E with 2 digits add-on.	V					8
UPE+5	UPC-E with 5 digits add-on.	V					11
MSI	MSI.		V	V	V		
MSIC	MSI with check digit.		V	V	V		
PLESSEY	PLESSEY.		V	V	V		
CPOST	China post.					V	
ITF14	ITF14.		V	V	V		13
EAN14	EAN14.	V					13
11	Code 11.		V	V	V		
TELEPEN	Telepen. *Since V6.89EZ.		V	V	V		
TELEPENN	Telepen number. *Since V6.89EZ.		V	V	V		

PLANET	Planet. *Since V6.89EZ.	V					
CODE49	Code 49. *Since V6.89EZ.	V					
DPI	Deutsche Post Identcode. *Since V6.91EZ.		V	V	V		11
DPL	Deutsche Post Leitcode. *Since V6.91EZ.		V	V	V		13
LOGMARS	A special use of Code 39. *Since V6.88EZ.		V	V	V		

附件二

错误代码	错误代码说明
100	其他错误
101	超过内存空间
102	内存被锁住
103	读取功率不足
104	非特定的错误
105	CRC错误
106	写入中若发生错误时，回复已写入多少 words 数
107	写入中若 Tag 标签回复错误时，错误码加上已写入多少 words 数
108	没有标签存在
109	指令格式错误
110	设定电源强度失败
111	设定法规失败

附件三

错误代码	错误代码说明
00	没有查询到电子标签
05	访问密码错误
FA	有电子标签,但通信不畅,操作失败
FB	无电子标签可操作
FC	电子标签返回错误
FD	命令长度错误
FE	不合法的命令
FF	参数错误