

WS73V100 Linux SLE 软件

# 开发指南

文档版本 06

发布日期 2024-10-21

# 前言

## 概述

本文档介绍了 WS73V100 SDK 在 Linux 平台下的星闪开发指导，帮助用户快速实现对 Linux 系统的星闪开发。

## 产品版本

与本文档对应的产品版本如下。

产品名称	产品版本
WS73	V100

## 读者对象





本文档主要适用以下工程师：

- 技术支持工程
- 软件开发工程师

## 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
----	----

符号	说明
 危险	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
06	2024-10-21	<ul style="list-style-type: none"><li>更新 “1 概述” 章节内容。</li><li>更新 “2 驱动编译” 章节内容。</li><li>更新 “3 协议栈获取” 章节内容。</li><li>更新 “4.1.3 注意事项” 章节内容。</li><li>更新 “4.4.2 开发流程” 章节内容。</li><li>更新 “5.1 sle_uuid sample 工程构建指导” 章节内容。</li><li>新增 “6 AT 指令指南” 章节内容。</li></ul>
05	2024-08-05	更新 “5.1 sle_uuid sample 工程构建指导” 章节内容。
04	2024-03-29	更新 “5.1 sle_uuid sample 工程构建指导” 章节内容。

文档版本	发布日期	修改说明
03	2024-03-21	新增 “5.1 sle_uuid sample 工程构建指导” 章节内容。
02	2024-01-29	更新 “4.4.2 开发流程” 章节内容。
01	2023-12-11	第一次正式版本发布。
00B02	2023-11-07	新增 “5 Sample 示例” 章节内容。
00B01	2023-10-24	第一次临时版本发布。

# 目 录

前言 .....	i
1 概述 .....	1
1.1 错误码 .....	1
2 驱动编译 .....	3
3 协议栈获取 .....	4
4 接口说明 .....	5
4.1 Device Discovery 接口 .....	5
4.1.1 概述 .....	5
4.1.2 开发流程 .....	5
4.1.3 注意事项 .....	8
4.2 Connection Manager 接口 .....	8
4.2.1 概述 .....	8
4.2.2 开发流程 .....	9
4.3 SSAP Server 接口 .....	11
4.3.1 概述 .....	11
4.3.2 开发流程 .....	11
4.4 SSAP client 接口 .....	15
4.4.1 概述 .....	15
4.4.2 开发流程 .....	15
5 Sample 示例 .....	19
5.1 sle_uuid sample 工程构建指导 .....	19
6 AT 指令指南 .....	21

# 1 概述

WS73V100 通过 API (Application Programming Interface) 面向开发者提供 SLE 功能的开发和应用接口，包括 Device Discovery, Connection Manager, SSAP 等。

各组件功能说明如下：

- Device Discovery：星闪设备发现协议，包括设备管理、设备公开和设备发现接口。
- Connection Manager：星闪连接管理协议，包括设备连接、配对相关接口。
- SSAP：星闪服务交互协议 (SparkLink Service Access Protocol)，包含服务注册、服务发现、属性数据读写等功能相关接口。
- Low Latency：低时延初始化和低时延数据收发接口。

## 说明

该文档描述各个模块功能的基本流程和 API 接口描述。

### 1.1 错误码

## 1.1 错误码

SLE SDK 使用错误码指示用户当前任务执行结果，如表 1-1 所示。

表1-1 错误码

序号	定义	实际数值	描述
1	ERRCODE_SLE_SUCCESS	0	执行成功错误码。
2	ERRCODE_SLE_CONTINUE	0x80006000	继续执行错误码。
3	ERRCODE_SLE_DIRECT_RE	0x80006001	直接返回错误码。

序号	定义	实际数值	描述
	TURN		
5	ERRCODE_SLE_PARAM_ERR	0x80006002	参数错误错误码。
6	ERRCODE_SLE_FAIL	0x80006003	执行失败错误码。
7	ERRCODE_SLE_TIMEOUT	0x80006004	执行超时错误码。
8	ERRCODE_SLE_UNSUPPOR TED	0x80006005	参数不支持错误码。
9	ERRCODE_SLE_GETRECOR D_FAIL	0x80006006	获取当前记录失败错误 码。
10	ERRCODE_SLE_POINTER_N ULL	0x80006007	指针为空错误码。
11	ERRCODE_SLE_NO_RECOR D	0x80006008	无记录返回错误码。
12	ERRCODE_SLE_STATUS_ER R	0x80006009	状态错误错误码。
13	ERRCODE_SLE_NOMEM	0x8000600a	内存不足错误码。
14	ERRCODE_SLE_AUTH_FAIL	0x8000600b	认证失败错误码。
15	ERRCODE_SLE_AUTH_PKEY _MISS	0x8000600c	PIN 码或密钥丢失致认 证失败错误码。
16	ERRCODE_SLE_RMT_DEV_D OWN	0x8000600d	对端设备关闭错误码。
17	ERRCODE_SLE_PAIRING_RE JECT	0x8000600e	配对拒绝错误码。
18	ERRCODE_SLE_BUSY	0x8000600f	系统繁忙错误码。
19	ERRCODE_SLE_NOT_READY	0x80006010	系统未准备好错误码。
20	ERRCODE_SLE_CONN_FAIL	0x80006011	连接失败错误码。
21	ERRCODE_SLE_OUT_OF_RA NGE	0x80006012	越界错误码。
22	ERRCODE_SLE_MEMCPY_FA IL	0x80006013	拷贝失败错误码。
23	ERRCODE_SLE_MALLOC_FA IL	0x80006014	内存申请失败错误码。

# 2 驱动编译

参考《WS73V100 Linux 平台驱动移植 用户指南》。



# 3 协议栈获取

查看 WS73 SDK 中 “application/lib” 文件夹下是否有对应主控文件夹，若有可使用对应主控中的协议栈；否则需要提供主控对应的交叉编译工具链给原厂，原厂编译协议栈后提供给客户。

# 4 接口说明

WS73V100 通过 API (Application Programming Interface) 向开发者提供接入和使用星闪低功耗的相关接口, 包括广播、连接以及 SSAP 服务注册、服务发现等, 其他协议相关接口将在后续增量发布。

## 4.1 Device Discovery 接口

## 4.2 Connection Manager 接口

## 4.3 SSAP Server 接口

## 4.4 SSAP client 接口

## 4.1 Device Discovery 接口

### 4.1.1 概述

Device Discovery 接口是星闪设备发现协议的软件实现, 主要功能有 SLE 设备开关、设备管理、设备公开和设备发现。

### 4.1.2 开发流程

#### 使用场景

打开 SLE 设备开关是使用 SLE 功能的首要条件, SLE 启动后可进行设备信息管理, 包括获取与设置本地设备名称、获取与设置本地设备地址和设置本地设备外观。

- 当 SLE 设备需要进行设备公开时, 可先设置设备公开参数、设备公开数据, 然后使能设备公开。

- 当 SLE 设备需要进行设备发现时，可先设置设备发现参数，然后使能设备发现，并通过回调函数观察发现到的设备公开数据包。

功能

Device Discovery 提供的接口如表 4-1 所示。

表4-1 Device Discovery 接口描述

接口名称	描述	参数说明	返回信息说明
enable_sle	使能 SLE。	-	接口返回值：错误码。
disable_sle	去使能 SLE。	-	接口返回值：错误码。
sle_set_local_addr	设置本地设备地址。	addr：本地设备地址。	接口返回值：错误码。
sle_get_local_addr	获取本地设备地址。	addr：[out]本地设备地址。	接口返回值：错误码。
sle_set_local_name	设置本地设备名称。	name：本地设备名称； len：本地设备名称长度。	接口返回值：错误码。
sle_get_local_name	获取本地设备名称。	name：[out]本地设备名称； len：[inout]入参时为用户预留内存大小，出参时为本地设备名称长度。	接口返回值：错误码。
sle_set_announce_data	设置设备公开数据。	announce_id：设备公开 ID； data：设备公	接口返回值：错误码。

接口名称	描述	参数说明	返回信息说明
		开数据。	
sle_set_announce_param	设置设备公开参数。	announce_id：设备公开ID； data：设备公开参数。	接口返回值：错误码。
sle_start_announce	开始设备公开。	announce_id：设备公开ID。	接口返回值：错误码。
sle_stop_announce	停止设备公开。	announce_id：设备公开ID。	接口返回值：错误码。
sle_set_seek_param	设置设备发现参数。	param：设备发现参数。	接口返回值：错误码。
sle_start_seek	开始设备发现。	-	接口返回值：错误码。
sle_stop_seek	停止设备发现。	-	接口返回值：错误码。
sle_announce_seek_register_callbacks	注册设备公开和设备发现回调函数。	func：用户回调函数。	接口返回值：错误码。

开发流程

Device Discovery 开发的典型流程如下，具体编程实例可参考“application/samples/bt”。

Terminal Node:

- 步骤 1 调用 enable sle，打开 SLE 开关。
- 步骤 2 调用 sle\_announce\_seek\_register\_callbacks，注册设备公开和设备发现回调函数。

步骤 3 调用 sle\_set\_local\_addr, 设置本地设备地址。

步骤 4 调用 sle\_set\_local\_name, 设置本地设备名称。

步骤 5 调用 sle\_set\_announce\_param, 设置设备公开参数

步骤 6 调用 sle\_set\_announce\_data, 设置设备公开数据

步骤 7 调用 sle\_start\_announce, 启动设备公开。

----结束

#### Grant Node:

步骤 1 调用 enable\_sle, 打开 SLE 开关。

步骤 2 调用 sle\_announce\_seek\_register\_callbacks, 注册设备公开和设备发现回调函数。

步骤 3 调用 sle\_set\_local\_addr, 设置本地设备地址。

步骤 4 调用 sle\_set\_local\_name, 设置本地设备名称。

步骤 5 调用 sle\_set\_seek\_param, 设置设备发现参数。

步骤 6 调用 sle\_start\_seek, 启动设备发现, 并在回调函数中获得正在进行设备公开的设备信息。

----结束

### 4.1.3 注意事项

若扫描不到设备, 请先检查设备是否已在配对设备列表中, 或者设备是否已与其他设备配对 (此情况下需要先清除设备端配对信息)

## 4.2 Connection Manager 接口

### 4.2.1 概述

Connection Manager 接口是星闪连接管理协议的软件实现, 主要功能有连接、配对和读远端设备 RSSI 值。

4.2.2 开发流程

使用场景

当设备需要与对端设备建立连接时，可向对端设备发起连接请求。在连接过程中，设备可读取远端设备 RSSI 值。当设备需要更新连接参数时，可向对端设备发起连接参数更新请求。

当设备需要与对端设备配对时，可向对端设备发起配对请求。在配对过程中，可获取当前本端设备与指定对端设备的配对状态。设备可获取当前配对设备数量以及当前配对设备信息链表。

功能

Connection Manager 提供的接口如表 4-2 所示。

表4-2 Connection Manager 接口描述

接口名称	描述	参数说明	返回信息说明
sle_connect_remote_device	向对端设备发起连接请求。	addr: 对端设备地址。	接口返回值：错误码。
sle_disconnect_remote_device	向对端设备发起断连请求。	addr: 对端设备地址。	接口返回值：错误码。
sle_update_connect_param	连接参数更新。	params: 连接参数	接口返回值：错误码。
sle_pair_remote_device	向对端设备发起配对请求（目前星闪鉴权流程仅支持免输入模式）。	addr: 对端设备地址。	接口返回值：错误码。
sle_remove_paired_remote_device	与对端设备取消配对。	addr: 对端设备地址。	接口返回值：错误码。
sle_remove_all_pairs	取消与所有对端设备的配对。	-	接口返回值：错误码。
sle_get_paired_devices_num	获取配对设备数量。	number: [out]配对设备数量。	接口返回值：错误码。

接口名称	描述	参数说明	返回信息说明
sle_get_paired_devices	获取配对设备信息。	addr: [out]设备地址链表; number: [inout]入参为用户预留内存大小, 出参时为设备数量。	接口返回值: 错误码。
sle_get_pair_state	获取配对状态。	addr: 设备地址; state: [out]配对状态。	接口返回值: 错误码。
sle_read_remote_device_rssi	读对端设备 RSSI 值。	conn_id: 连接 id	接口返回值: 错误码。
sle_connection_register_callbacks	注册连接管理回调函数。	func: 用户回调函数。	接口返回值: 错误码。

开发流程

Connection Manager 开发的典型流程如下，具体编程实例可参考 application/samples/bt。

Terminal Node:

- 步骤 1 调用 enable\_sle，打开 SLE 开关。
- 步骤 2 调用 sle\_announce\_seek\_register\_callbacks，注册设备公开和设备发现回调函数。
- 步骤 3 调用 sle\_connection\_register\_callbacks，注册连接管理回调函数。
- 步骤 4 调用 sle\_set\_local\_addr，设置本地设备地址。
- 步骤 5 调用 sle\_set\_local\_name，设置本地设备名称。
- 步骤 6 调用 sle\_set\_announce\_param，设置设备公开参数
- 步骤 7 调用 sle\_set\_announce\_data，设置设备公开数据
- 步骤 8 调用 sle\_start\_announce，启动设备公开。

#### ----结束

#### Grant Node:

步骤 1 调用 `enable_sle`, 打开 SLE 开关。

步骤 2 调用 `sle_announce_seek_register_callbacks`, 注册设备公开和设备发现回调函数。

步骤 3 调用 `sle_connection_register_callbacks`, 注册连接管理回调函数。

步骤 4 调用 `sle_set_local_addr`, 设置本地设备地址。

步骤 5 调用 `sle_set_local_name`, 设置本地设备名称。

步骤 6 调用 `sle_set_seek_param`, 设置设备发现参数。

步骤 7 调用 `sle_start_seek`, 启动设备发现, 并在回调函数中获得正在进行设备公开的设备信息。

步骤 8 调用 `sle_connect_remote_device`, 向对端设备发起连接请求。

步骤 9 调用 `sle_pair_remote_device`, 向对端设备发起配对请求。

步骤 10 调用 `sle_get_paired_devices_num`, 获取当前配对设备数量

步骤 11 调用 `sle_get_paired_devices`, 获取当前配对设备信息。

步骤 12 调用 `sle_get_pair_state`, 获取配对状态。

#### ----结束

## 4.3 SSAP Server 接口

### 4.3.1 概述

SSAP 是 SLE 发送和接收数据的通用规范, 支持在两个 SLE 设备间进行数据传输。

### 4.3.2 开发流程

#### 使用场景

SSAP Server 主要接收对端的请求和命令, 向对端发送响应、通知和指示。



功能

SSAP Server 提供的接口如表 4-3 所示。

表4-3 SSAP Server 接口描述

接口名称	描述	参数说明	返回信息说明
ssaps_register_server	注册 SSAP server。 <b>注：目前只支持注册一个 SSAP server。</b>	app_uuid：应用 UUID 指针； server_id：[out] server id 指针。	接口返回值：错误码。
ssaps_unregister_server	注销 SSAP server。	server_id：server id。	接口返回值：错误码。
ssaps_add_service	添加服务。	server_id：server id； service_uuid：服务 UUID； is_primary：是否是首要服务。	接口返回值：错误码。
ssaps_add_property	添加特征。	server_id：server id； service_handle：服务句柄； property：特征信息。	接口返回值：错误码。
ssaps_add_descriptor	添加特征描述符。	server_id：server id； service_handle：服务句柄； property_handle：特征句柄； descriptor：描述符信息。	接口返回值：错误码。

接口名称	描述	参数说明	返回信息说明
ssaps_add_service_sync	添加服务同步接口，服务句柄同步返回。	server_id: server id;  service_uuid: 服务UUID;  is_primary: 是否是首要服务;  handle: [out]服务句柄指针。	接口返回值: 错误码。
ssaps_add_property_sync	添加特征同步接口，特征句柄同步返回。	server_id: server id;  service_handle: 服务句柄;  property: 特征;  handle: [out]特征句柄指针。	接口返回值: 错误码。
ssaps_add_descriptor_sync	添加特征描述符同步接口，特征描述符句柄同步返回。	server_id: server id;  service_handle: 服务句柄;  property_handle: 特征句柄  descriptor: 特征描述符;  handle: [out]特征描述符句柄指针。	接口返回值: 错误码。
ssaps_start_service	启动服务。	server_id: server id;  service_handle: 服务句柄。	接口返回值: 错误码。
ssaps_delete_all_ser	删除所有服务。	server_id: server	接口返回值: 错

接口名称	描述	参数说明	返回信息说明
vices		id。	误码。
ssaps_send_response	发送响应。	server_id: server id; conn_id: 连接 ID; param: 响应参数。	接口返回值: 错误码。
ssaps_notify_indicate	给对端发送通知或指示。明确下发数据的速率要大于连接间隔 30%。	server_id: server id; conn_id: 连接 ID; param: 通知或指示参数。	接口返回值: 错误码。
ssaps_notify_indicate_by_uuid	按照 uuid 给对端发送通知或指示。明确下发数据的速率要大于连接间隔 30%。	server_id: server id; conn_id: 连接 ID; param: 通知或指示参数。	接口返回值: 错误码。
ssaps_set_info	在连接之前设置 server 信息。	server_id: server id; info: server 信息。	接口返回值: 错误码。
ssaps_register_callbacks	注册 SSAP server 回调函数。	func: 用户回调函数。	接口返回值: 错误码。

开发流程

SSAP server 开发的典型流程：注册 SSAP server，注册本端属性数据库，接收对端的请求和命令，向对端发送通知和指示，具体编程实例可参考“application/samples/bt”。

- 步骤 1 调用 enable\_sle，打开 SLE 开关。
- 步骤 2 调用 ssaps\_register\_callbacks，注册 SSAP server 回调。
- 步骤 3 调用 sle\_announce\_seek\_register\_callbacks，注册设备公开和设备发现回调函数。

- 步骤 4 调用 `ssaps_register_server`，创建一个 `server` 实体。
- 步骤 5 调用 `ssaps_add_service_sync`、`ssaps_add_property_sync`、`ssaps_add_descriptor_sync` 和 `ssaps_start_service` 注册本端属性数据库，每一个服务及其内容添加完成后调用 `ssaps_start_service` 启动服务。
- 步骤 6 调用 `sle_set_local_addr`，设置本地设备地址。
- 步骤 7 调用 `sle_set_local_name`，设置本地设备名称。
- 步骤 8 调用 `sle_set_announce_param`，设置设备公开参数。
- 步骤 9 调用 `sle_set_announce_data`，设置设备公开数据。
- 步骤 10 调用 `sle_start_announce`，启动设备公开。
- 步骤 11 连接建立。
- 步骤 12 接收对端设备的读写请求，当对端设备读写需要授权的特征或描述符时，调用 `ssaps_send_response` 向对端发送响应并修改本端特征值。
- 步骤 13 当某个特征的客户端特征配置描述符为 `0x0001` 时，在特征值变化时向对端设备发送通知，当某个特征的客户端特征配置描述符为 `0x0002` 时，在特征值变化时向对端设备发送指示。

----结束

## 4.4 SSAP client 接口

### 4.4.1 概述

SSAP 是 SLE 发送和接收数据的通用规范，支持在两个 SLE 设备间进行数据传输。

### 4.4.2 开发流程

#### 使用场景

SSAP Client 主要向对端发送请求和命令，接收对端的响应、通知和指示。

#### 功能

SSAP Client 提供的接口如表 4-4 所示。

表4-4 SSAP Client 接口描述

接口名称	描述	参数说明	返回信息说明
ssapc_register_client	注册 SSAP client。 <b>注：目前只支持注册一个 SSAP client。</b>	app_uuid：应用 UUID 指针； client_id：[out] client id 指针。	接口返回值：错误码。
ssapc_unregister_client	注销 SSAP client。	client_id：client id。	接口返回值：错误码。
ssapc_find_structure	查找对端服务、特征和描述符。	client_id：client id； conn_id：连接 ID； param：查找参数。	接口返回值：错误码。
ssapc_read_req_by_uuid	向对端发送按照 uuid 读取请求。	client_id：client id； conn_id：连接 ID； param：读取参数。	接口返回值：错误码。
ssapc_read_req	向对端发送读取请求。	client_id：client id； conn_id：连接 ID； handle：句柄； type：类型。	接口返回值：错误码。
ssapc_write_req	向对端发送写请求。明确下发数据的速率要大于连接间隔 30%。	client_id：client id； conn_id：连接 ID； param：写参数。	接口返回值：错误码。
ssapc_write_cmd	向对端发送写命令。明确下发数据的速率要大于连接间隔 30%。	client_id：client id； conn_id：连接 ID； param：写参数。	接口返回值：错误码。
ssapc_exchange_info_req	向对端发送交换信息请求。	client_id：client id； conn_id：连接 ID； param：交换信息参数。	接口返回值：错误码。
ssapc_register	注册 SSAP client 回调函	func：用户回调函	接口返回值：错

接口名称	描述	参数说明	返回信息说明
ster_callbacks	数。	数。	误码。

## 开发流程

SSAP Client 开发的典型流程：注册 SSAP Client，查找对端属性数据库，向对端发送请求和命令，接收对端的通知和指示，具体编程实例可参考 application/samples/bt。

### SSAP Server:

- 步骤 1 调用 enable\_sle，打开 SLE 开关。
- 步骤 2 调用 ssaps\_register\_callbacks，注册 SSAP server 回调。
- 步骤 3 调用 sle\_announce\_seek\_register\_callbacks，注册设备公开和设备发现回调函数。
- 步骤 4 调用 ssaps\_register\_server，创建一个 server 实体。
- 步骤 5 调用 ssaps\_add\_service\_sync、ssaps\_add\_property\_sync、ssaps\_add\_descriptor\_sync 和 ssaps\_start\_service 注册本端属性数据库，每一个服务及其内容添加完成后调用 ssaps\_start\_service 启动服务。
- 步骤 6 调用 sle\_set\_local\_addr，设置本地设备地址。
- 步骤 7 调用 sle\_set\_local\_name，设置本地设备名称。
- 步骤 8 调用 sle\_set\_announce\_param，设置设备公开参数。
- 步骤 9 调用 sle\_set\_announce\_data，设置设备公开数据。
- 步骤 10 调用 sle\_start\_announce，启动设备公开。
- 步骤 11 连接建立。
- 步骤 12 接收对端设备的读写请求，当对端设备读写需要授权的特征或描述符时，调用 ssaps\_send\_response 向对端发送响应并修改本端特征值。
- 步骤 13 当某个特征的客户端特征配置描述符为 0x0001 时，在特征值变化时向对端设备发送通知，当某个特征的客户端特征配置描述符为 0x0002 时，在特征值变化时向对端设备发送指示。

----结束

**SSAP Client:**

- 步骤 1 调用 `enable_sle`, 打开 SLE 开关。
- 步骤 2 调用 `ssapc_register_callbacks`, 注册 SSAP client 回调。
- 步骤 3 调用 `sle_announce_seek_register_callbacks`, 注册设备公开和设备发现回调函数。
- 步骤 4 调用 `ssapc_register_client`, 创建一个 client 实体。
- 步骤 5 递归调用 `ssapc_find_structure` 查找对端属性数据库。
- 步骤 6 如果关注对端某个特征, 可调用 `ssapc_write_req` 或 `ssapc_write_cmd` 将该特征的客户端特征配置描述符写为 0x0001 或 0x0002, 前者可使能对端特征通知, 后者可使能对端特征指示。
- 步骤 7 调用读写接口操作对端属性数据库。

----结束

# 5 Sample 示例

## 5.1 sle\_uuid sample 工程构建指导

### 5.1 sle\_uuid sample 工程构建指导

该 sample 用于验证星闪的 client 能力，包括打开星闪、client 注册、扫描、连接、数传和关闭星闪功能。其中设置的 data length 和 phy 默认为 WS73U 参数，若使用 WS73E，请根据要求进行调整（WS73E 的 data size 最大 1450，data length 最大 1530，MTU 最大 1500，phy 支持 1M/2M/4M）。

```
27 #define SLE_DATA_SIZE_DEFAULT 200 // 73E可使用1450
28 #define SLE_SEEK_INTERVAL_DEFAULT 160 // 160
29 #define SLE_SEEK_WINDOW_DEFAULT 40 // 40
30 #define UUID_16BIT_LEN 2
31 #define UUID_128BIT_LEN 16
32 #define SPEED_DEFAULT_CONN_INTERVAL 0x10
33 #define SPEED_DEFAULT_TIMEOUT_MULTIPLIER 0x1f4
34 #define DATA_LEN 251 // 设置datalen, 73e用1530, 73u用251
35 #define DEFAULT_SLE_SPEED_MCS 10
36 #define DEFAULT_SLE_SPEED_MTU_SIZE 1500
37
38 #define MS_100 100000
39 #define THOUSAND 1000
40 #define UUID_14_BYTE 14
41 #define UUID_15_BYTE 15
```



```

121 void sle_sample_pair_complete_cbk(uint16_t conn_id, const sle_addr_t *addr, errcode_t status)
122 {
123     printf("sle_sample_pair_complete_cbk status: %d\n", status);
124     sle_set_mcs(conn_id, DEFAULT_SLE_SPEED_MCS);
125
126     sle_set_phy_t phy_parm = {
127         .tx_format = SLE_RADIO_FRAME_2,
128         .rx_format = SLE_RADIO_FRAME_2,
129         .tx_phy = SLE_PHY_2M, // 73E支持SLE_PHY_4M
130         .rx_phy = SLE_PHY_2M, // 73E支持SLE_PHY_4M
131         .tx_pilot_density = SLE_PHY_PILOT_DENSITY_16_TO_1,
132         .rx_pilot_density = SLE_PHY_PILOT_DENSITY_16_TO_1,
133         .g_feedback = 0,
134         .t_feedback = 0,
135     };
136     errcode_t ret2 = sle_set_phy_param(conn_id, &phy_parm);
137     printf("[ssap client] set phy, ret: %d\n", ret2);
138 }

```

步骤 1 更改 Makefile 中的 CROSS 宏为主控使用的工具链。

```

1  # CROSS 为交叉编译工具链
2  CROSS = arm-himix100-linux-
3  CC = @echo " GCC  $@"; $(CROSS)gcc
4  LD = @echo " LD  $@"; $(CROSS)ld
5  AR = @echo " AR  $@"; $(CROSS)ar
6  RM = @echo " RM  $@"; rm -f
7  STRIP = @echo "STRIP  $@"; $(CROSS)strip

```

步骤 2 在 sample 的 lib 文件夹中放置 BTH 协议栈静态库文件（若 WS73 SDK 中 "application/lib" 文件夹下有对应主控文件夹，可使用对应主控中的协议栈；否则需要提供主控对应的交叉编译工具链给原厂，原厂编译协议栈后提供给客户）。

步骤 3 在 WS73 SDK 的 "application/sample/sle/sle\_uuid/sle\_uuid\_client" 路径下执行 "make clean;make" 即可编译出 sle\_client\_sample 应用（不要更改 sample 的路径，Makefile 中定义了 sample 需要的头文件的相对路径）；sle\_server\_sample 同理，存放在 "application/sample/sle/sle\_uuid/sle\_uuid\_server" 路径下，在此路径下执行 "make clean;make"。

步骤 4 在主控上加载 plat\_soc.ko 和 sle\_soc.ko，并确认正常运行后，在 client 端运行 sle\_client\_sample，在 server 端运行 sle\_server\_sample。两块主控会自动连接并开始数传。

----结束

# 6

## AT 指令指南

参考《WS73V100 命令使用指南》中的“BLE&SLE 模块 AT 指令”章节。