SDK_Docs_Template

开发文档

这是一个专为SDK文档构建设计的GitHub模板仓库，提供完整的自动化文档构建机制。

Version 1.1.0 | 中文 | 2025年09月12日

kurisaw.eu.org

目录

1. 基础篇

1.1. 基础示例项目11

这是一个基础功能示例项目，展示了SDK的基本使用方法。

功能特性

- LED闪烁控制
- 按键中断处理
- 基础定时器使用

快速开始

硬件要求

- 开发板一块
- USB数据线
- 调试器（可选）

编译步骤

1. 打开项目文件
2. 配置编译选项
3. 编译项目
4. 下载到开发板

运行结果

程序运行后，LED会按照设定的频率闪烁，按键按下时会触发中断处理。

代码结构

```
basic_example/
├── src/
│   ├── main.c          # 主程序
│   ├── led.c           # LED控制
│   └── key.c           # 按键处理
├── inc/
│   ├── led.h
│   └── key.h
└── README.md
```

注意事项

- 确保硬件连接正确
- 检查电源电压是否正常
- 观察串口输出信息

故障排除

如果遇到问题，请检查：

1. 硬件连接
2. 电源供应
3. 程序配置
4. 编译选项

2. 驱动篇

2.1. 驱动示例项目

这是一个外设驱动示例项目，展示了如何使用SDK中的各种驱动接口。

功能特性

- UART通信驱动
- SPI接口使用
- I2C设备访问
- ADC数据采集

快速开始

硬件要求

- 开发板一块
- 串口转USB模块
- SPI/I2C外设模块
- 模拟信号源

编译步骤

1. 配置硬件接口
2. 修改驱动参数
3. 编译项目
4. 下载运行

运行结果

程序会初始化各种外设驱动，并通过串口输出测试结果。

代码结构

```
driver_example/
├── src/
│   ├── main.c          # 主程序
│   ├── uart_drv.c      # UART驱动
│   ├── spi_drv.c       # SPI驱动
│   ├── i2c_drv.c       # I2C驱动
│   └── adc_drv.c       # ADC驱动
├── inc/
│   ├── uart_drv.h
│   ├── spi_drv.h
│   ├── i2c_drv.h
│   └── adc_drv.h
└── README.md
```

驱动接口

UART驱动

```
// 初始化UART
uart_init(115200);

// 发送数据
uart_send("Hello World\n");

// 接收数据
char data = uart_receive();
```

SPI驱动

```
// 初始化SPI
spi_init(SPI_MODE0, 1000000);

// 发送数据
spi_send(0x55);
```

```
// 接收数据
uint8_t data = spi_receive();
```

I2C驱动

```
// 初始化I2C
i2c_init(100000);

// 写数据
i2c_write(0x50, 0x00, 0x55);

// 读数据
uint8_t data = i2c_read(0x50, 0x00);
```

ADC驱动

```
// 初始化ADC
adc_init(ADC_CH0);

// 读取数据
uint16_t value = adc_read(ADC_CH0);
```

注意事项

- 检查硬件连接
- 确认驱动参数
- 观察串口输出
- 验证数据正确性

故障排除

常见问题及解决方案：

1. **串口无输出** - 检查波特率设置
2. **SPI通信失败** - 确认时钟极性和相位
3. **I2C设备无响应** - 检查地址和时序
4. **ADC数据异常** - 验证参考电压

3. 组件篇

3.1. 组件示例项目

这是一个网络组件示例项目，展示了如何使用SDK中的各种网络组件和协议栈。

功能特性

- TCP/IP网络通信
- MQTT消息传输
- HTTP客户端
- 网络配置管理

快速开始

硬件要求

- 开发板一块
- 以太网连接
- WiFi模块（可选）
- 网络调试工具

编译步骤

1. 配置网络参数
2. 设置服务器地址
3. 编译项目
4. 下载运行

运行结果

程序会连接到网络，建立各种网络连接，并进行数据传输测试。

代码结构

```
component_example/
├── src/
│   ├── main.c              # 主程序
│   ├── tcp_client.c        # TCP客户端
│   ├── mqtt_client.c       # MQTT客户端
│   ├── http_client.c       # HTTP客户端
│   └── network_config.c    # 网络配置
├── inc/
│   ├── tcp_client.h
│   ├── mqtt_client.h
│   ├── http_client.h
│   └── network_config.h
└── README.md
```

网络组件

TCP客户端

```
// 创建TCP连接
tcp_connect("192.168.1.100", 8080);

// 发送数据
tcp_send("Hello Server\n");

// 接收数据
char buffer[1024];
int len = tcp_receive(buffer, sizeof(buffer));
```

MQTT客户端

```
// 连接MQTT服务器
mqtt_connect("mqtt.example.com", 1883);

// 订阅主题
mqtt_subscribe("sensor/data");
```

```
// 发布消息
mqtt_publish("device/status", "online");

// 接收消息
void mqtt_message_callback(const char* topic, const char* messa
    printf("收到消息: %s -> %s\n", topic, message);
}
```

HTTP客户端

```
// 发送GET请求
http_get("http://api.example.com/data");

// 发送POST请求
http_post("http://api.example.com/upload", "{\"data\":\"value\"

// 处理响应
void http_response_callback(int status, const char* body) {
    printf("HTTP状态: %d\n", status);
    printf("响应内容: %s\n", body);
}
```

网络配置

```
// 配置IP地址
network_set_ip("192.168.1.100");

// 配置网关
network_set_gateway("192.168.1.1");

// 配置DNS
network_set_dns("8.8.8.8");

// 启动网络
network_start();
```

网络协议

支持协议

- **TCP/UDP** - 传输层协议

- **HTTP/HTTPS** - 应用层协议
- **MQTT** - 消息队列协议
- **CoAP** - 受限应用协议
- **WebSocket** - 实时通信协议

安全特性

- TLS/SSL加密
- 证书验证
- 身份认证
- 数据完整性

注意事项

- 确保网络连接正常
- 检查服务器地址和端口
- 验证网络配置参数
- 观察网络状态指示

故障排除

常见网络问题及解决方案：

1. **无法连接网络** - 检查IP配置和网关
2. **TCP连接失败** - 确认服务器地址和端口
3. **MQTT连接超时** - 检查网络延迟和防火墙
4. **HTTP请求失败** - 验证URL格式和服务器状态