

# SDK\_Docs\_Template

---

Documentation

*This is a GitHub template repository specifically designed for building SDK documentation, offering a complete automated documentation construction mechanism.*

Version 1.1.0

| English

| November 04, 2025

[kurisaw.eu.org](https://kurisaw.eu.org)

2025, KurisaW

# Contents

- 1. Basics .....
  - 1.1. Basic Example Project .....
- 2. Drivers .....
  - 2.1. Driver Example Project .....
- 3. Components .....
  - 3.1. Component Example Project .....

# 1. Basics

---

## 1.1. Basic Example Project

---

This is a basic example project that demonstrates the fundamental usage of the SDK.

### Features

- LED blinking control
- Button interrupt handling
- Basic timer usage

### Quick Start

#### Hardware Requirements

- One development board
- USB cable
- Debugger (optional)

#### Build Steps

1. Open the project files
2. Configure the build options
3. Compile the project
4. Download to the development board

## Running Result

After the program runs, the LED will blink at the configured frequency, and pressing the button will trigger an interrupt handler.

## Code Structure

```
basic_example/
├── src/
│   ├── main.c      # Main program
│   ├── led.c       # LED control
│   └── key.c       # Button handling
├── inc/
│   ├── led.h
│   └── key.h
└── README.md
```

## Notes

- Ensure hardware connections are correct
- Check that the power supply voltage is normal
- Monitor the serial output information

## Troubleshooting

If issues occur, please check:

1. Hardware connections
2. Power supply
3. Program configuration
4. Build options

## 2. Drivers

---

### 2.1. Driver Example Project

---

This is a peripheral driver example project that demonstrates how to use various driver interfaces in the SDK.

#### Features

- UART communication driver
- SPI interface usage
- I2C device access
- ADC data acquisition

#### Quick Start

##### Hardware Requirements

- One development board
- USB-to-serial module
- SPI/I2C peripheral modules
- Analog signal source

##### Build Steps

1. Configure hardware interfaces
2. Modify driver parameters
3. Compile the project
4. Download and run

## Running Result

The program will initialize various peripheral drivers and output test results via the serial port.

## Code Structure

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

## Driver Interfaces

### UART Driver

```
// Initialize UART  
uart_init(115200);  
  
// Send data  
uart_send("Hello World\n");  
  
// Receive data  
char data = uart_receive();
```

### SPI Driver

```
// Initialize SPI  
spi_init(SPI_MODE0, 1000000);  
  
// Send data  
spi_send(0x55);
```

```
// Receive data
uint8_t data = spi_receive();
```

## I2C Driver

```
// Initialize I2C
i2c_init(100000);

// Write data
i2c_write(0x50, 0x00, 0x55);

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

## ADC Driver

```
// Initialize ADC
adc_init(ADC_CH0);

// Read data
uint16_t value = adc_read(ADC_CH0);
```

## Notes

- Check hardware connections
- Confirm driver parameters
- Observe serial output
- Verify data correctness

## Troubleshooting

Common issues and solutions:

1. **No serial output** – Check baud rate settings
2. **SPI communication failure** – Verify clock polarity and phase
3. **No response from I2C device** – Check address and timing
4. **Abnormal ADC data** – Verify reference voltage

## 3. Components

---

### 3.1. Component Example Project

---

This is a network component example project that demonstrates how to use various network components and protocol stacks in the SDK.

#### Features

- TCP/IP network communication
- MQTT message transmission
- HTTP client
- Network configuration management

#### Quick Start

##### Hardware Requirements

- One development board
- Ethernet connection
- Wi-Fi module (optional)
- Network debugging tools

##### Build Steps

1. Configure network parameters
2. Set the server address
3. Compile the project
4. Download and run



## Running Result

The program will connect to the network, establish various network connections, and perform data transmission tests.

## Code Structure

```
component_example/  
├── src/  
│   ├── main.c           # Main program  
│   ├── tcp_client.c     # TCP client  
│   ├── mqtt_client.c    # MQTT client  
│   ├── http_client.c    # HTTP client  
│   └── network_config.c # Network configuration  
├── inc/  
│   ├── tcp_client.h  
│   ├── mqtt_client.h  
│   ├── http_client.h  
│   └── network_config.h  
└── README.md
```

## Network Components

### TCP Client

```
// Create a TCP connection  
tcp_connect("192.168.1.100", 8080);  
  
// Send data  
tcp_send("Hello Server\n");  
  
// Receive data  
char buffer[1024];  
int len = tcp_receive(buffer, sizeof(buffer));
```

### MQTT Client

```
// Connect to MQTT server  
mqtt_connect("mqtt.example.com", 1883);  
  
// Subscribe to a topic
```

```
mqtt_subscribe("sensor/data");

// Publish a message
mqtt_publish("device/status", "online");


// Receive messages
void mqtt_message_callback(const char* topic, const char* message) {
    printf("Message received: %s -> %s\n", topic, message);
}
```

## HTTP Client

```
// Send GET request
http_get("http://api.example.com/data");

// Send POST request
http_post("http://api.example.com/upload", "{\"data\":\"value\"}");

// Handle response
void http_response_callback(int status, const char* body) {
    printf("HTTP Status: %d\n", status);
    printf("Response Body: %s\n", body);
}
```



## Network Configuration

```
// Configure IP address
network_set_ip("192.168.1.100");

// Configure gateway
network_set_gateway("192.168.1.1");

// Configure DNS
network_set_dns("8.8.8.8");

// Start network
network_start();
```

# Network Protocols

## Supported Protocols

- **TCP/UDP** – Transport layer protocols
- **HTTP/HTTPS** – Application layer protocols
- **MQTT** – Message queue protocol
- **CoAP** – Constrained Application Protocol
- **WebSocket** – Real-time communication protocol

## Security Features

- TLS/SSL encryption
- Certificate validation
- Authentication
- Data integrity

## Notes

- Ensure the network connection is stable
- Check the server address and port
- Verify network configuration parameters
- Monitor network status indicators

## Troubleshooting

Common network issues and solutions:

1. **Unable to connect to the network** – Check IP configuration and gateway
2. **TCP connection failed** – Verify server address and port
3. **MQTT connection timeout** – Check network latency and firewall
4. **HTTP request failed** – Validate URL format and server status

