ECx00U&EGx00U 系列

QuecOpen SPI 开发指导

LTE Standard 模块系列

版本：1.0

日期：2021-09-01

状态：受控文件

上海移远通信技术股份有限公司（以下简称“移远通信”）始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 5108 6236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，请随时登陆网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

移远通信提供该文档内容以支持客户的产品设计。客户须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，移远通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，尽管移远通信采取了商业范围内的合理努力来提供尽可能好的体验，但本文档和其所涉及服务是在“可用”基础上提供给您的。移远通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

使用和披露限制

许可协议

除非移远通信特别授权，否则我司所提供硬软件、材料和文档的接收方须对接收的内容保密，不得将其用于除本项目的实施与开展以外的任何其他目的。

版权声明

移远通信产品和本协议项下的第三方产品可能包含受移远通信或第三方材料、硬软件和文档版权保护的相关资料。除非事先得到书面同意，否则您不得获取、使用、向第三方披露我司所提供的文档和信息，或对此类受版权保护的资料进行复制、转载、抄袭、出版、展示、翻译、分发、合并、修改，或创造其衍生作品。移远通信或第三方对受版权保护的资料拥有专有权，不授予或转让任何专利、版权、商标或服务商标权的许可。为避免歧义，除了正常的非独家、免版税的产品使用许可，任何形式的购买都不可被视为授予许可。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，移远通信有权追究法律责任。

商标

除另行规定，本文档中的任何内容均不授予在广告、宣传或其他方面使用移远通信或第三方的任何商标、商号及名称，或其缩略语，或其仿冒品的权利。

第三方权利

您理解本文档可能涉及一个或多个属于第三方的硬软件和文档（“第三方材料”）。您对此类第三方材料的使用应受本文档的所有限制和义务约束。

移远通信针对第三方材料不做任何明示或暗示的保证或陈述，包括但不限于任何暗示或法定的适销性或特定用途的适用性、平静受益权、系统集成、信息准确性以及与许可技术或被许可人使用许可技术相关的不侵犯任何第三方知识产权的保证。本协议中的任何内容都不构成移远通信对任何移远通信产品或任何其他硬软件、设备、工具、信息或产品的开发、增强、修改、分销、营销、销售、提供销售或以其他方式维持生产的陈述或保证。此外，移远通信免除因交易过程、使用或贸易而产生的任何和所有保证。

免责声明

1) 移远通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。

2) 移远通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。

3) 移远通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则移远通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，移远通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。

4) 移远通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2021-03-08	Ryan YI	文档创建
1.0	2021-09-01	Ryan YI	受控版本

目录

表格索引

1 引言

移远通信 ECx00U 系列和 EGx00U 模块支持 QuecOpen®方案；QuecOpen®是基于 RTOS 的嵌入式开发平台，可简化 IoT 应用的软件设计和开发过程。有关 QuecOpen®的详细信息，请参考***文档 [1]***。

本文档主要介绍在 QuecOpen®方案下，ECx00U 系列和 EGx00U 模块的 SPI 开发准备、相关 API、开发示例及相关调试。

1.1. 适用模块

表 1：适用模块

模块系列	模块
ECx00U	EC200U 系列
	EC600U 系列
EGx00U	EG500U-CN
	EG700U-CN

2 SPI 开发准备工作

在 ECx00U 系列和 EGx00U QuecOpen 模块上实现 SPI 功能需使用移远通信提供的如下固件库：

- #include “ql_api_spi.h”

3 SPI 相关 API

3.1. 头文件

SPI API 的头文件为 *ql_api_spi.h*，位于 *components\ql-kernel\inc* 目录下。若无特别说明，本文档所提及的头文件均位于该目录下。

3.2. 函数概览

表 2：函数概览

函数	说明
ql_spi_init()	初始化 SPI
ql_spi_init_ext()	初始化 SPI（配置 SPI 总线参数）
ql_spi_read_follow_write()	设置通过 SPI 先发送数据、再接收数据
ql_spi_write_read()	设置通过 SPI 同时发送和接收数据
ql_spi_read()	设置通过 SPI 接收数据
ql_spi_write()	设置通过 SPI 发送数据
ql_spi_release()	释放 SPI 总线
ql_spi_cs_low()	强制拉低 SPI 的 CS 引脚
ql_spi_cs_high()	强制拉高 SPI 的 CS 引脚
ql_spi_cs_auto()	设置由系统控制 SPI 的 CS 引脚
ql_spi_set_irq()	设置 SPI 使用 DMA 中断回调函数
ql_spi_get_tx_fifo_free()	获取 SPI TX FIFO 中空闲数据数目

ql_spi_request_sys_clk()	申请不能进入慢时钟
ql_spi_release_sys_clk()	释放不能进入慢时钟的申请

3.3. 函数详解

3.3.1. ql_spi_init

该函数用于初始化 SPI，应在使用 SPI 其他 API 前调用。调用该函数前，需通过 *ql_pin_set_func()*设置指定 GPIO 引脚为 SPI 功能，详情请参考***文档 [2]***。

调用该函数将默认设置 *ql_spi_input_mode_e* 为 *QL_SPI_INPUT_TRUE*，*framesize* 为 *8*，*ql_spi_cs_pol_e* 为 *QL_SPI_CS_ACTIVE_LOW*，*ql_spi_cpol_pol_e* 为 *QL_SPI_CPOL_LOW*，*ql_spi_cpha_pol_e* 为 *QL_SPI_CPHA_1Edge*，*ql_spi_input_sel_e* 为 *QL_SPI_DI_1*，*ql_spi_cs_sel_e* 为 *QL_SPI_CS0*，*ql_spi_clk_delay_e* 为 *QL_SPI_CLK_DELAY_0*。相关参数描述详情请见***第 3.3.2 章***。

- **函数原型**

```
ql_errcode_spi_e ql_spi_init(ql_spi_port_e port, ql_spi_transfer_mode_e transmode, ql_spi_clk_e spiclk)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

transmode：
[In] SPI 传输模式。详见***第 3.3.1.2 章***。

spiclk：
[In] SPI 时钟频率。详见***第 3.3.1.3 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.1.1. ql_spi_port_e

SPI 总线编号枚举定义如下：

```
typedef enum
{
```

```
    QL_SPI_PORT1,
    QL_SPI_PORT2,
}ql_spi_port_e;
```

- **参数**

参数	描述
QL_SPI_PORT1	SPI1 总线
QL_SPI_PORT2	SPI2 总线

3.3.1.2. ql_spi_transfer_mode_e

SPI 传输模式枚举定义如下：

```
typedef enum
{
    QL_SPI_DIRECT_POLLING = 0,
    QL_SPI_DIRECT_IRQ,
    QL_SPI_DMA_POLLING,
    QL_SPI_DMA_IRQ,
}ql_spi_transfer_mode_e;
```

- **参数**

参数	描述
QL_SPI_DIRECT_POLLING	FIFO 读写，轮询等待。
QL_SPI_DIRECT_IRQ	FIFO 读写，中断通知。暂不支持。
QL_SPI_DMA_POLLING	DMA 读写，轮询等待。此模式下，SPI 不可与 SD 卡同时使用。若已使用 SD 卡，则 SPI 将申请 DMA 通道失败、初始化失败。
QL_SPI_DMA_IRQ	DMA 读写，中断通知。

备注

使用 DMA 轮询模式时，SPI 与 SD 卡抢占 DMA 通道；若 SD 卡已使用 DMA 通道，SPI 初始化则会失败。

3.3.1.3. ql_spi_clk_e

SPI 时钟频率枚举定义如下：

```
typedef enum
{
    QL_SPI_CLK_INVALID=-1,
    QL_SPI_CLK_781_25KHZ = 781250,
    QL_SPI_CLK_1_5625MHZ = 1562500,
    QL_SPI_CLK_3_125MHZ = 3125000,
    QL_SPI_CLK_5MHZ = 5000000,
    QL_SPI_CLK_6_25MHZ = 6250000,
    QL_SPI_CLK_10MHZ = 10000000,
    QL_SPI_CLK_12_5MHZ = 12500000,
    QL_SPI_CLK_20MHZ = 20000000,
    QL_SPI_CLK_25MHZ = 25000000,
    QL_SPI_CLK_33_33MHZ = 33000000,
    QL_SPI_CLK_50MHZ_MAX = 50000000,
}ql_spi_clk_e;
```

- **参数**

参数	描述
QL_SPI_CLK_INVALID	无效参数
QL_SPI_CLK_781_25KHZ	时钟频率为 781.25 kHz
QL_SPI_CLK_1_5625MHZ	时钟频率为 1.5625 MHz
QL_SPI_CLK_3_125MHZ	时钟频率为 3.125 MHz
QL_SPI_CLK_5MHZ	时钟频率为 5 MHz
QL_SPI_CLK_6_25MHZ	时钟频率为 6.25 MHz
QL_SPI_CLK_10MHZ	时钟频率为 10 MHz
QL_SPI_CLK_12_5MHZ	时钟频率为 12.5 MHz
QL_SPI_CLK_20MHZ	时钟频率为 20 MHz
QL_SPI_CLK_25MHZ	时钟频率为 25 MHz
QL_SPI_CLK_33_33MHZ	时钟频率为 33.33 MHz
QL_SPI_CLK_50MHZ_MAX	时钟频率为 50 MHz（最大时钟频率）

3.3.1.4. ql_errcode_spi_e

SPI 错误码枚举定义如下：

```
typedef enum
{
    QL_SPI_SUCCESS                  =  0,
    QL_SPI_ERROR                    =  1 | (QL_COMPONENT_BSP_SPI << 16),
    QL_SPI_PARAM_TYPE_ERROR,
    QL_SPI_PARAM_DATA_ERROR,
    QL_SPI_PARAM_ACQUIRE_ERROR,
    QL_SPI_PARAM_NULL_ERROR,
    QL_SPI_DEV_NOT_ACQUIRE_ERROR,
    QL_SPI_PARAM_LENGTH_ERROR,
    QL_SPI_MALLOC_MEM_ERROR,
}ql_errcode_spi_e;
```

- **参数**

参数	描述
QL_SPI_SUCCESS	函数执行成功
QL_SPI_ERROR	函数执行失败
QL_SPI_PARAM_TYPE_ERROR	参数类型错误
QL_SPI_PARAM_DATA_ERROR	参数数据错误
QL_SPI_PARAM_ACQUIRE_ERROR	参数无法获取
QL_SPI_PARAM_NULL_ERROR	指针参数为 NULL
QL_SPI_DEV_NOT_ACQUIRE_ERROR	无法获取 SPI 总线
QL_SPI_PARAM_LENGTH_ERROR	参数长度错误
QL_SPI_MALLOC_MEM_ERROR	申请内存错误

3.3.2. ql_spi_init_ext

该函数用于初始化 SPI（配置 SPI 总线参数），应在使用 SPI 其他 API 前调用。

该函数可用于选择 SPI 总线、配置 SPI 输入功能、时钟频率、数据帧大小、CS 引脚和引脚电平、时钟极性、时钟相位、数据输入引脚、传输模式及 MISO 延时采样。调用此函数前，需通过 *ql_pin_set_func()* 设置相关 GPIO 引脚为 SPI 功能，详情请参考***文档 [2]***。

- **函数原型**

```
ql_errcode_spi_e ql_spi_init_ext(ql_spi_config_s spi_config)
```

- **参数**

spi_config：
[In] SPI 总线参数配置。详情请见***第 3.3.2.1 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.2.1. ql_spi_config_s

SPI 总线参数配置结构体定义如下：

```
typedef struct
{
    ql_spi_input_mode_e input_mode;
    ql_spi_port_e port;
    unsigned int framesize;
    ql_spi_clk_e spiclk;
    ql_spi_cs_pol_e cs_polarity0;
    ql_spi_cs_pol_e cs_polarity1;
    ql_spi_cpol_pol_e cpol;
    ql_spi_cpha_pol_e cpha;
    ql_spi_input_sel_e input_sel;
    ql_spi_transfer_mode_e transmode;
    ql_spi_cs_sel_e cs;
    ql_spi_clk_delay_e clk_delay;
} ql_spi_config_s;
```

- **参数**

类型	参数	描述
ql_spi_input_mode_e	*input_mode*	SPI 输入功能。详见***第 3.3.2.2 章***。
ql_spi_port_e	*port*	SPI 总线编号。详见***第 3.3.1.1 章***。
unsigned int	*framesize*	数据帧大小。范围：0~32；默认值：8；单位：bit。
ql_spi_clk_e	*spiclk*	SPI 时钟频率。详见***第 3.3.1.3 章***。

ql_spi_cs_pol_e	*cs_polarity0*	CS0 引脚电平。详见***第 3.3.2.3 章***。
ql_spi_cs_pol_e	*cs_polarity1*	CS1 引脚电平。详见***第 3.3.2.3 章***。
ql_spi_cpol_pol_e	*cpol*	时钟极性。详见***第 3.3.2.4 章***。
ql_spi_cpha_pol_e	*cpha*	时钟相位。详见***第 3.3.2.5 章***。
ql_spi_input_sel_e	*input_sel*	数据输入引脚。详见***第 3.3.2.6 章***。
ql_spi_transfer_mode_e	*transmode*	SPI 传输模式。详见***第 3.3.1.2 章***。
ql_spi_cs_sel_e	*cs*	CS 引脚。详见***第 3.3.2.7 章***。
ql_spi_clk_delay_e	*clk_delay*	MISO 延时采样。详见***第 3.3.2.8 章***。

3.3.2.2. ql_spi_input_mode_e

SPI 输入功能枚举定义如下：

```
typedef enum
{
    QL_SPI_INPUT_FALSE,
    QL_SPI_INPUT_TRUE,
}ql_spi_input_mode_e;
```

- **参数**

参数	描述
QL_SPI_INPUT_FALSE	禁用 SPI 输入（读取）功能
QL_SPI_INPUT_TRUE	启用 SPI 输入（读取）功能

3.3.2.3. ql_spi_cs_pol_e

CS 引脚电平枚举定义如下：

```
typedef enum
{
    QL_SPI_CS_ACTIVE_HIGH,
    QL_SPI_CS_ACTIVE_LOW,
} ql_spi_cs_pol_e;
```

- **参数**

参数	描述
QL_SPI_CS_ACTIVE_HIGH	高电平
QL_SPI_CS_ACTIVE_LOW	低电平

3.3.2.4. ql_spi_cpol_pol_e

时钟极性枚举定义如下：

```
typedef enum
{
    QL_SPI_CPOL_LOW = 0,
    QL_SPI_CPOL_HIGH,
} ql_spi_cpol_pol_e;
```

- **参数**

参数	描述
QL_SPI_CPOL_LOW	未启用 SPI 时，CLK 线为低电平，第一个边沿为上升沿。
QL_SPI_CPOL_HIGH	未启用 SPI 时，CLK 线为高电平，第一个边沿为下降沿。

3.3.2.5. ql_spi_cpha_pol_e

时钟相位枚举定义如下：

```
typedef enum
{
    QL_SPI_CPHA_1Edge,
    QL_SPI_CPHA_2Edge,
}ql_spi_cpha_pol_e;
```

- **参数**

参数	描述
QL_SPI_CPHA_1Edge	数据采样从第一个时钟边沿开始
QL_SPI_CPHA_2Edge	数据采样从第二个时钟边沿开始

该枚举需要和枚举 *ql_spi_cpol_pol_e* 配合使用，构成 SPI 的四种模式：

- SPI mode0：*ql_spi_cpol_pol_e* 选择 *QL_SPI_CPOL_LOW*，*ql_spi_cpha_pol_e* 选择 *QL_SPI_CPHA_1Edge*。
- SPI mode1：*ql_spi_cpol_pol_e* 选择 *QL_SPI_CPOL_LOW*，*ql_spi_cpha_pol_e* 选择 *QL_SPI_CPHA_2Edge*。
- SPI mode2：*ql_spi_cpol_pol_e* 选择 *QL_SPI_CPOL_HIGH*，*ql_spi_cpha_pol_e* 选择 *QL_SPI_CPHA_1Edge*。
- SPI mode3：*ql_spi_cpol_pol_e* 选择 *QL_SPI_CPOL_HIGH*，*ql_spi_cpha_pol_e* 选择 *QL_SPI_CPHA_2Edge*。

3.3.2.6. ql_spi_input_sel_e

数据输入引脚枚举定义如下：

```
typedef enum
{
    QL_SPI_DI_0 = 0,
    QL_SPI_DI_1,
    QL_SPI_DI_2,
}ql_spi_input_sel_e;
```

- **参数**

参数	描述
QL_SPI_DI_0	DI0 为数据输入引脚（暂不支持）
QL_SPI_DI_1	DI1 为数据输入引脚
QL_SPI_DI_2	DI2 为数据输入引脚（暂不支持）

3.3.2.7. ql_spi_cs_sel_e

CS 引脚枚举定义如下：

```
typedef enum
{
    QL_SPI_CS0 = 0,
    QL_SPI_CS1,
    QL_SPI_CS2,
    QL_SPI_CS3,
}ql_spi_cs_sel_e;
```

- **参数**

参数	描述
QL_SPI_CS0	CS0 为 SPI 的 CS 引脚
QL_SPI_CS1	CS1 为 SPI 的 CS 引脚
QL_SPI_CS2	CS2 为 SPI 的 CS 引脚（暂不支持）
QL_SPI_CS3	CS3 为 SPI 的 CS 引脚（暂不支持）

3.3.2.8. ql_spi_clk_delay_e

MISO 延时采样枚举定义如下：

```
typedef enum
{
    QL_SPI_CLK_DELAY_0 = 0,
    QL_SPI_CLK_DELAY_1,
}ql_spi_clk_delay_e;
```

- **参数**

参数	描述
QL_SPI_CLK_DELAY_0	MISO 无延时采样
QL_SPI_CLK_DELAY_1	MISO 延时一个时钟边沿采样

3.3.3. ql_spi_read_follow_write

该函数用于设置通过 SPI 先发送数据、再接收数据。在发送和接收数据的过程中，CS 引脚保持为有效电平。

- **函数原型**

```
ql_errcode_spi_e ql_spi_read_follow_write(ql_spi_port_e port, unsigned char *outbuf, unsigned int outlen, unsigned char *inbuf, unsigned int inlen)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

outbuf：
[In] 发送数据。

outlen：
[In] 发送数据的长度。单位：字节。

inbuf：
[Out] 接收数据。

inlen：
[In] 接收数据的长度。单位：字节。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.4. ql_spi_write_read

该函数用于设置通过 SPI 同时发送和接收数据。

- **函数原型**

```
ql_errcode_spi_e ql_spi_write_read(ql_spi_port_e port, unsigned char *inbuf, unsigned char *outbuf, unsigned int len)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

inbuf：
[Out] 接收数据。

outbuf：
[In] 发送数据。

len：
[In] 发送和接收数据的长度。单位：字节。

- **返回值**

详见***第 3.3.1.4 章***。

备注

在调用该函数前后，需要调用 *ql_spi_cs_low()*、*ql_spi_cs_high()*或 *ql_spi_cs_auto()*自行控制 CS 引脚。

3.3.5. ql_spi_read

该函数用于设置通过 SPI 接收数据。

- **函数原型**

```
ql_errcode_spi_e ql_spi_read(ql_spi_port_e port, unsigned char *buf, unsigned int len)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

buf：
[Out] 接收数据。

len：
[In] 接收数据的长度。单位：字节。

- **返回值**

详见***第 3.3.1.4 章***。

备注

在调用该函数前后，需要调用 *ql_spi_cs_low()*、*ql_spi_cs_high()*或 *ql_spi_cs_auto()*自行控制 CS 引脚。

3.3.6. ql_spi_write

该函数用于设置通过 SPI 发送数据。

- **函数原型**

```
ql_errcode_spi_e ql_spi_write(ql_spi_port_e port, unsigned char *buf, unsigned int len)
```

- **参数**

port：
[In] SPI 总线编号，请参考***第 3.3.1.1 章***。

buf：
[In] 发送数据。

len：
[In] 发送数据的长度。单位：字节。

- **返回值**

详见***第 3.3.1.4 章***。

备注

在调用该函数前后，需要调用 *ql_spi_cs_low()*、*ql_spi_cs_high()*或 *ql_spi_cs_auto()*自行控制 CS 引脚。

3.3.7. ql_spi_release

该函数用于释放 SPI 总线。

- **函数原型**

```
ql_errcode_spi_e ql_spi_release(ql_spi_port_e port)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.8. ql_spi_cs_low

该函数用于强制拉低 SPI 的 CS 引脚。

- **函数原型**

```
ql_errcode_spi_e ql_spi_cs_low(ql_spi_port_e port)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.9. ql_spi_cs_high

该函数用于强制拉高 SPI 的 CS 引脚。

- **函数原型**

```
ql_errcode_spi_e ql_spi_cs_high(ql_spi_port_e port)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.10. ql_spi_cs_auto

该函数用于设置由系统控制 SPI 的 CS 引脚。

- **函数原型**

```
ql_errcode_spi_e ql_spi_cs_auto(ql_spi_port_e port)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.11. ql_spi_set_irq

该函数用于设置 SPI 使用 DMA 中断回调函数，仅在 *QL_SPI_DMA_IRQ* 模式下使用。

- **函数原型**

```
ql_errcode_spi_e ql_spi_set_irq(ql_spi_port_e port, ql_spi_irq_s mask, ql_spi_callback callfunc)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

mask：
[In] 中断源掩码。详见***第 3.3.11.1 章***。

callfunc：
[In] DMA 中断回调函数。详见***第 3.3.11.3 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.11.1. ql_spi_irq_s

中断源掩码结构体定义如下：

```
typedef struct
{
    unsigned int rx_ovf : 1;
    unsigned int tx_th : 1;
    unsigned int tx_dma_done : 1;
    unsigned int rx_th : 1;
    unsigned int rx_dma_done : 1;
    ql_spi_threshold_e tx_threshold;
    ql_spi_threshold_e rx_threshold;
}ql_spi_irq_s;
```

- **参数**

类型	参数	描述
unsigned int	*rx_ovf*	RX 接收溢出。

unsigned int	*tx_th*	TX FIFO 中断使能。
unsigned int	*tx_dma_done*	DMA 发送完成中断。
unsigned int	*rx_th*	RX FIFO 中断。
unsigned int	*rx_dma_done*	DMA 接收完成。
ql_spi_threshold_e	*tx_threshold*	TX FIFO 触发中断级别。详见***第 3.3.11.2 章***。
ql_spi_threshold_e	*rx_threshold*	RX FIFO 触发中断级别。详见***第 3.3.11.2 章***。

3.3.11.2. ql_spi_threshold_e

FIFO 触发中断级别枚举定义如下：

```
typedef enum
{
        QL_SPI_TRIGGER_1_DATA,
        QL_SPI_TRIGGER_4_DATA,
        QL_SPI_TRIGGER_8_DATA,
        QL_SPI_TRIGGER_12_DATA,
}ql_spi_threshold_e;
```

- **参数**

参数	描述
QL_SPI_TRIGGER_1_DATA	FIFO 触发中断级别为 1 字节/半字
QL_SPI_TRIGGER_4_DATA	FIFO 触发中断级别为 4 字节/半字
QL_SPI_TRIGGER_8_DATA	FIFO 触发中断级别为 8 字节/半字
QL_SPI_TRIGGER_12_DATA	FIFO 触发中断级别为 12 字节/半字

3.3.11.3. ql_spi_callback

该函数为 DMA 中断回调函数。

- **函数原型**

```
typedef void (*ql_spi_callback)(ql_spi_irq_s cause)
```

- **参数**

cause：
[In] 中断源掩码。详见***第 3.3.11.1 章***。

- **返回值**

无

3.3.12. ql_spi_get_tx_fifo_free

该函数用于获取 SPI TX FIFO 中空闲数据数目。

- **函数原型**

```
ql_errcode_spi_e ql_spi_get_tx_fifo_free(ql_spi_port_e port, unsigned int *tx_free)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

tx_free：
[Out] SPI TX FIFO 中空闲数据数目。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.13. ql_spi_request_sys_clk

该函数用于申请不能进入慢时钟。

- **函数原型**

```
ql_errcode_spi_e ql_spi_request_sys_clk(ql_spi_port_e port)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

- **返回值**

详见***第 3.3.1.4 章***。

3.3.14. ql_spi_release_sys_clk

该函数用于释放不能进入慢时钟的申请。

- **函数原型**

```
ql_errcode_spi_e ql_spi_release_sys_clk(ql_spi_port_e port)
```

- **参数**

port：
[In] SPI 总线编号。详见***第 3.3.1.1 章***。

- **返回值**

详见***第 3.3.1.4 章***。

4 SPI 开发示例

本章节主要介绍在 APP 侧如何使用上述 API 进行 SPI 开发以及简单调试。在该示例中，使用示波器或逻辑分析仪分析 SPI 的时序，无需连接 SPI 外设。

4.1. 开发示例

移远通信 ECx00U 系列和 EGx00U QuecOpen SDK 中提供了 SPI API 示例文件 *components\ql-application\spi\spi_demo.c*。

spi_demo.c 主要包含初始化 SPI 资源、通过 SPI 读写数据等设置。入口函数为 *ql_spi_demo_init()*，如下图所示。

```
QlOSStatus ql_spi_demo_init(void)
{
    QlOSStatus err = QL_OSI_SUCCESS;

    err = ql_rtos_task_create(&spi_demo_task, SPI_DEMO_TASK_STACK_SIZE, SPI_DEMO_TASK_PRIO, "ql_spi_demo", ql_spi_demo_task_pthread, NULL, SPI_DEMO_TASK_EVENT_CNT);
    if(err != QL_OSI_SUCCESS)
    {
        QL_SPI_DEMO_LOG("demo_task created failed");
        return err;
    }

    return err;
}
```

该示例已在 *ql_init_demo_thread()*线程中默认不启动。如有需要，取消 *ql_spi_demo_init()*的注释即可运行。如下图所示：

```
#ifdef QL_APP_FEATURE_SPI
    //ql_spi_demo_init();
#endif
```

4.2. 功能调试

模块可通过使用移远通信 LTE OPEN EVB 进行通用 SPI 功能调试，并根据示波器或者逻辑分析仪分析 SPI 的时序。

将编译版本烧录到模块中，使用 USB 线连接 EVB 的 USB 端口和 PC。使用 cooltools 工具抓取 log 后，可通过 USB 的 AP Log 端口查看该示例的调试信息。Log 抓取方法请参考***文档 [3]***。

模块开机后自动启动 *ql_spi_demo_init()*。通过示波器或逻辑分析仪即可查看 SPI 时序是否正确。如函数执行失败，可根据对应 log 信息查看失败原因。

5 附录 参考文档及术语缩写

表 3：参考文档

文档名称
[1] Quectel_ECx00U&EGx00U 系列_QuecOpen_CSDK_快速开发指导
[2] Quectel_ECx00U&EGx00U 系列_QuecOpen_GPIO_API_参考手册
[3] Quectel_ECx00U&EGx00U 系列_QuecOpen_Log_抓取指导

表 4：术语缩写

缩写	英文全称	中文全称
API	Application Programming Interface	应用程序接口
AP	Application Processor	应用处理器
APP	Application	应用
CLK	Clock	时钟
CPHA	Clock Phase	时钟相位
CPOL	Clock Polarity	时钟极性
CS	Chip Select	片选
DI	Data Input	数据输入
DMA	Direct Memory Access	直接存储器访问
DO	Data Output	数据输出
EVB	Evaluation Board	评估板
FIFO	First Input First Output	先进先出
IoT	Internet of Things	物联网

MISO	Master In Slave Out	主机输入从机输出
MOSI	Master Out Slave In	主机输出从机输入
RTOS	Real-Time Operating System	实时操作系统
PC	Personal Computer	个人计算机
SDK	Software Development Kit	软件开发工具包
SPI	Serial Peripheral Interface	串行外设接口
USB	Universal Serial Bus	通用串行总线