利用 QuadSPI 外扩串行 NOR Flash 的实现

前言

STM32 提供了灵活多样的外扩存储器访问实现。本文中，介绍如何利用 QSPI (QuadSPI) 外扩串行 NOR Flash 存储器。首先对 QSPI 接口功能特性进行介绍，然后分别介绍硬件设计和软件开发。并基于 STM32CubeMX，提供访问 MICRON N25Q128A13EF840F 的实现参考。

一 实现环境

开发板：STM32F469G-DISCO

开发库：STM32CubeF4 v1.16.0

STM32CubeMX: v4.22.0

集成开发环境：IAR v7.70.1.11486

实现过程在 STM32F469I-DISCO 板上展开，利用板上已有的串行 NOR Flash 存储器（MICRON N25Q128A13EF840F）。呈现整个开发涉及环节。在本文中，首先根据 QSPI 接口，介绍 QSPI 与外扩串行存储器硬件连接。另外，Cube 软件包中包含 QSPI 实现例，在本文对库中实现的 QSPI 例不做讨论，读者也可参考这些 QSPI 例进行设计。本文围绕由 STM32CubeMX 生成的工程，介绍如何实现对外扩串行 NOR Flash 存储器的访问。

二 QSPI 介绍

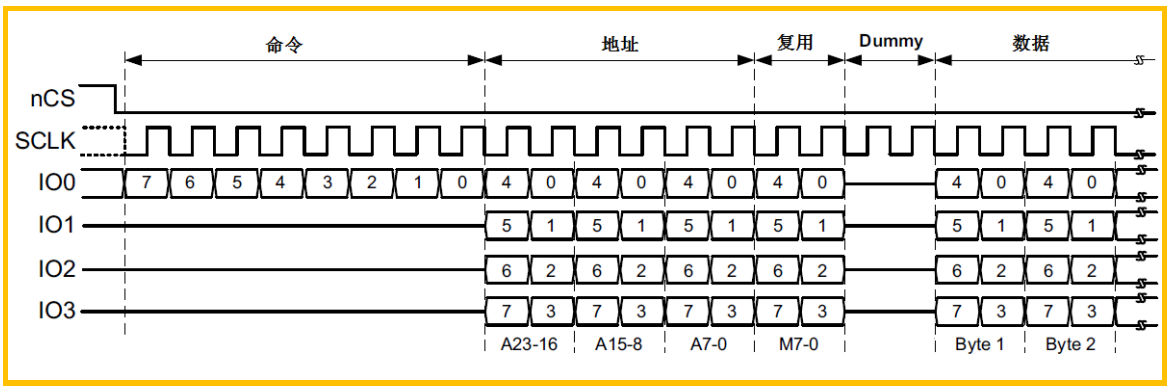
在呈现 QSPI 访问外扩 Flash 的实现例前， 需要对 QSPI 有一定的了解，在此对 QSPI 进行简短的介绍。更多内容请参考 AN4760。

QSPI（Quad-SPI）支持四线串行访问形式。同时，QSPI 支持传统 SPI 和 Dual-SPI 模式，Dual-SPI 模式支持两线串行访问。与 FMC/FSMC 比较，QSPI 支持更低成本、更小封装外部串行 Flash 存储器，更少的 IO 引脚占用，有效减少 PCB 面积，降低 PCB 设计复杂度。

QSPI 在不同系列 STM32 产品线的支持情况（仅部分罗列，未涵盖所有支持型号）。

QSPI 特性	**最大时钟速度（MHz）**		**双 Flash 访问**	**最大地址空间**	
	SDR	DDR	**支持情况**	存储器映射	间接模式
STM32L4x6	60	48	不支持	256MB	4GB
STM32F446	90	60	支持		
STM32F469/F479	90	80	支持		
STM32F7x5/F7x6	108	80	支持		

QSPI 接口提供了灵活可配置的 5 个阶段，如下图所示（仅用于理解阶段构成，时序图根据配置不同存在差异）。分别是命令阶段、地址阶段、复用字节阶段、Dummy 阶段和数据阶段。可以根据外扩 Flash 中命令时序对不同阶段进行配置。后续会以实例进行呈现。更多内容请参考 AN4760。

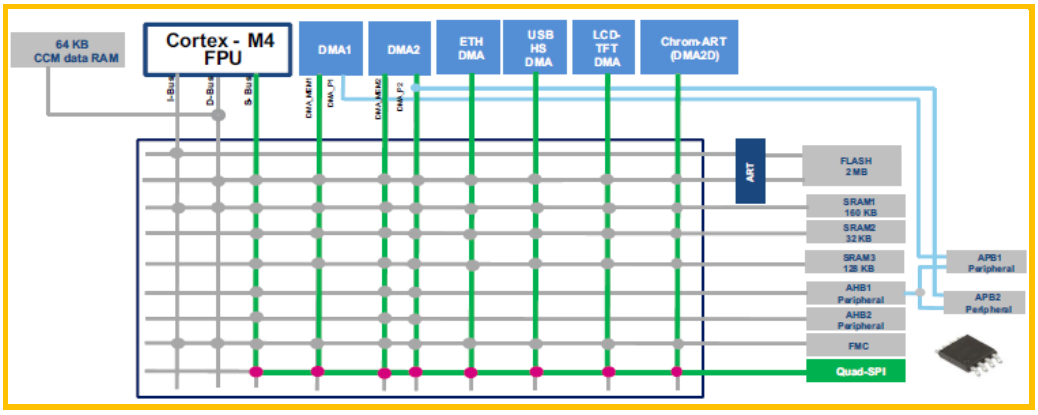


QSPI 支持三种模式，分别是：

间接模式 → 所有操作通过 QSPI 寄存器实现，类似于传统 SPI，可以使用阻塞模式、中断模式或者 DMA 模式进行读写等访问。本文中提供的实现例为间接模式下的实现。

状态轮询模式 → 接口自动轮询指定寄存器，直到回读寄存器内容与指定条件匹配。可应用于状态检测，从而实现忙等待等效果。本文不对此模式进行实现介绍，应用实现可参考 Cube 软件包中 QSPI 例程。

存储器映射模式 → 外扩 Flash 被视为内部存储器，支持 AHB 主器件直接访问，CPU 能够直接运行位于 QSPI 存储器的执行代码。内部系统架构如下图所示（以 STM32F469/F479 为例）。本文不对此模式进行实现介绍，应用实现可参考 Cube 软件包中 QSPI 例程 QSPI_ExecuteInPlace。



三 QSPI 外扩串行 Flash 实现例

3.1 串行 Flash 介绍

以 MICRON N25Q128A13EF840F 为例，更多细节请参考存储器手册。N25Q128A13EF840F 引脚图、时序图和电气参数来源于 N25Q128A13 手册文档。

支持协议： SPI, Dual I/O（对应 Dual-SPI）, Quad I/O（对应 Quad-SPI）

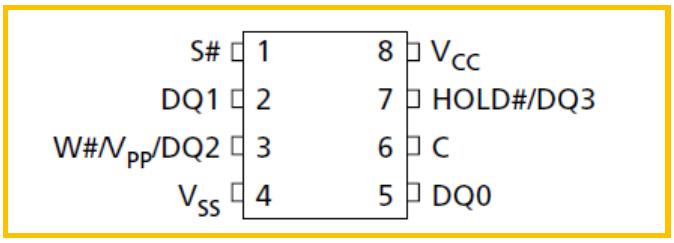
支持访问模式： 单线访问、双线访问、四线访问，得益于 QSPI 接口的灵活可配性，三种访问模式全部支持。

供电电压范围： 2.7 ~ 3.6V

最大时钟频率： 108MHz

存储空间： 128Mb （16MB）

器件引脚示意图如下所示。由两根电源引脚和六根 QSPI 信号线构成。



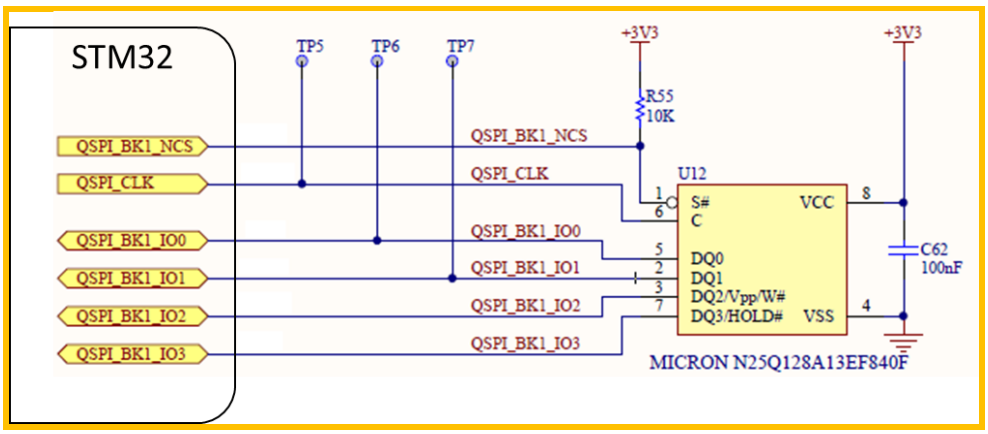
下表为存储器 N25Q128A13xxx 命令（未列出全部命令）。通过下表可知，存储器提供了灵活的访问实现形式，而结合同样灵活可配的 QSPI 接口，能够实现存储器命令全支持。而本文仅为提供设计思路，呈现了部分命令在 QSPI 上的实现。

命令类型	命令	命令码（十六进制）	单线	双线	四线	数据字节
复位	复位使能	66	支持	支持	支持	0
	复位存储器	99	支持	支持	支持	
ID 操作	读 ID	9E/9F	支持	不支持	不支持	1 ~ 20
读操作	读	3	支持	不支持	不支持	1 ~ ∞
	快速读	0B	支持	支持	支持	1 ~ ∞
	四线快速读	6B	支持	不支持	支持	1 ~ ∞
写操作	写使能	06	支持	支持	支持	0
	写失能	04	支持	支持	支持	
寄存器操作	读状态寄存器	05	支持	支持	支持	1 ~ ∞
	写状态寄存器	01	支持	支持	支持	1
烧写操作	页写	02	支持	支持	支持	1 ~ 256
	四线快速写	32	支持	不支持	支持	1 ~ 256
擦操作	子扇区擦	20	支持	支持	支持	0
	扇区擦	D8	支持	支持	支持	
	块擦	C7	支持	支持	支持	

其中，默认读写默认 3 字节地址，四线快速读命令默认 Dummy 周期数为 8。

3.2 硬件设计

涉及到的信号线少，硬件设计简单，只需直接将 QSPI 的六根信号线与存储器连接即可。考虑到可测性，可以增加串行电阻或者测试点。硬件电路图如下所示。



QSPI 接口 PCB 设计遵循如下几点，更多硬件设计内容请参考 AN4488。

a. 线阻 50Ω ± 10%

b. 最大线长 < 120mm

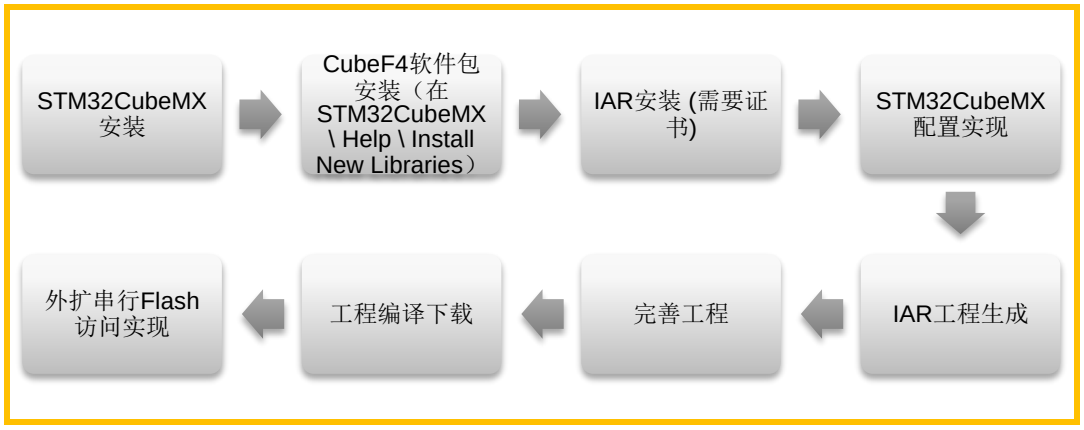
c. 避免在不同信号层走信号线

d. 时钟线至少离其他信号线 3 倍线宽距离

e. 数据信号线长差 ≤ 10mm

f. 避免时钟线采用蛇形走线，同时尽量减少数据线上过孔。

3.3 软件开发流程



3.4 软件实现例

在环境搭建完成后，就可以利用 STM32CubeMX 根据硬件连接情况，进行 QSPI 配置，获取 IAR 工程。具体软件实现流程如下。

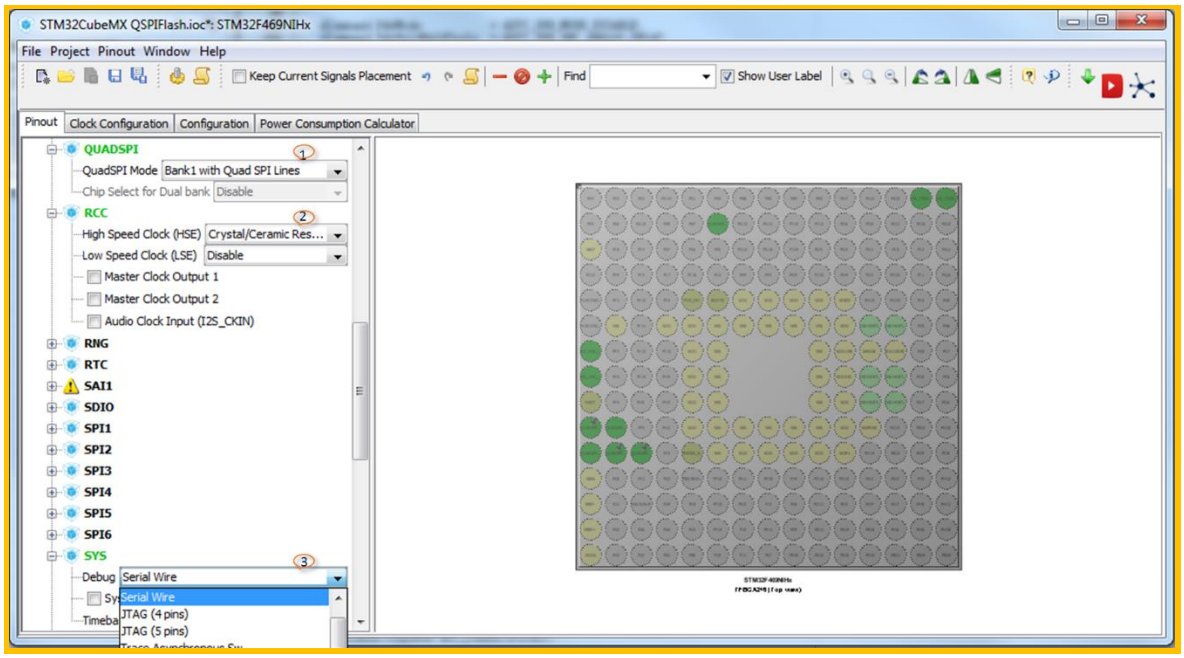
a. 利用 STM32CubeMX 生成 IAR 工程

打开 STM32CubeMX → 点击"New project" → 在"Part Number Search'中输入 STM32F469NI → 点击"MCUs Liast" 中出现的 STM32F469NIHx → 点击"Start Project" ➔ 此时，基于 STM32F469NIHx 的 STM32CubeMX 工程被打开。如下，根据 STM32F469I-DISCO 板硬件连接情况（QSPI NCS, CLK, Q0, Q1, Q2, Q3 对应 PB6, PF10, PF8, PF9, PF7, PF6；外部高速晶振为 8MHz 无源晶振；调试接口采用 SWD 接口，其中 SWCLK, SWDIO 对应 PA14, PA13）：选择" QuadSPI"为"Bank1 with Quad SPI Lines"（注：也可在开发过程中，先用 STM32CubeMX 查看 QSPI 接口对应的 IO 引脚，进行硬件开发）。

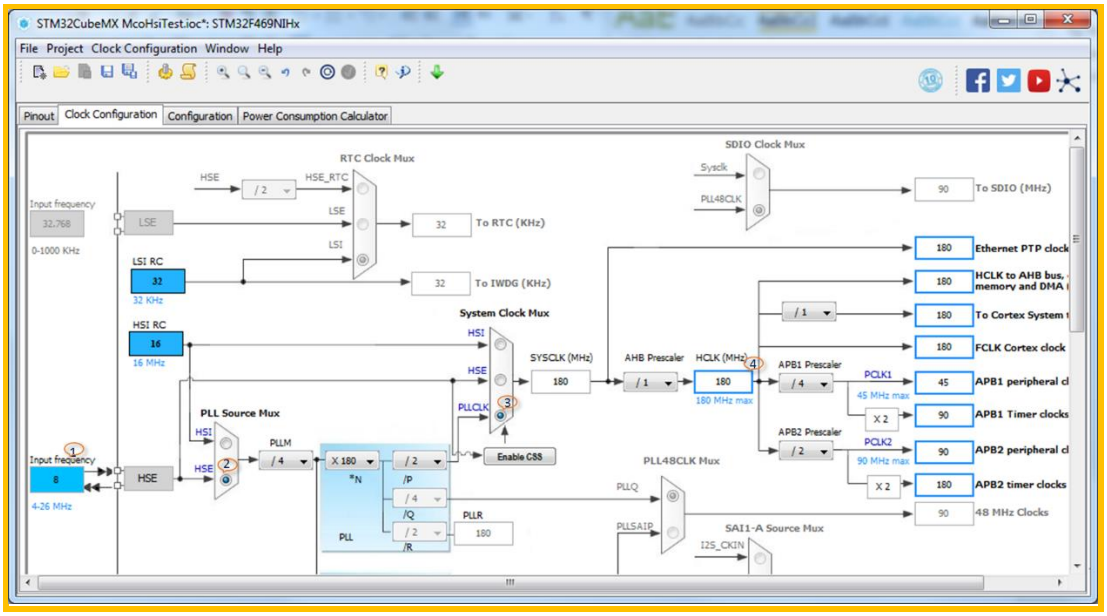
注： 在如上选择后，右侧引脚图中 QSPI 对应的引脚会呈现绿色显示。需要根据电路图中所连接的 QSPI 引脚，进行复用引脚确认。例如，在默认情况下，QSPI IO0 对应到 PC9 引脚，而 STM32F469I-DISCO 板上的 QSPI IO0 与 PF8 连接，并非 PC9。所以，需要在右侧引脚图中，按住 Ctrl 键，左键在 PC9 引脚按下，拖动至 PF8 处，松开左键和 Ctrl 键，实现 IO0 引脚的关联。

选择"High Speed Clock（HSE）"为"Crystal/Ceramic Resonator"。

选择"Debug"为"Serial Wire"



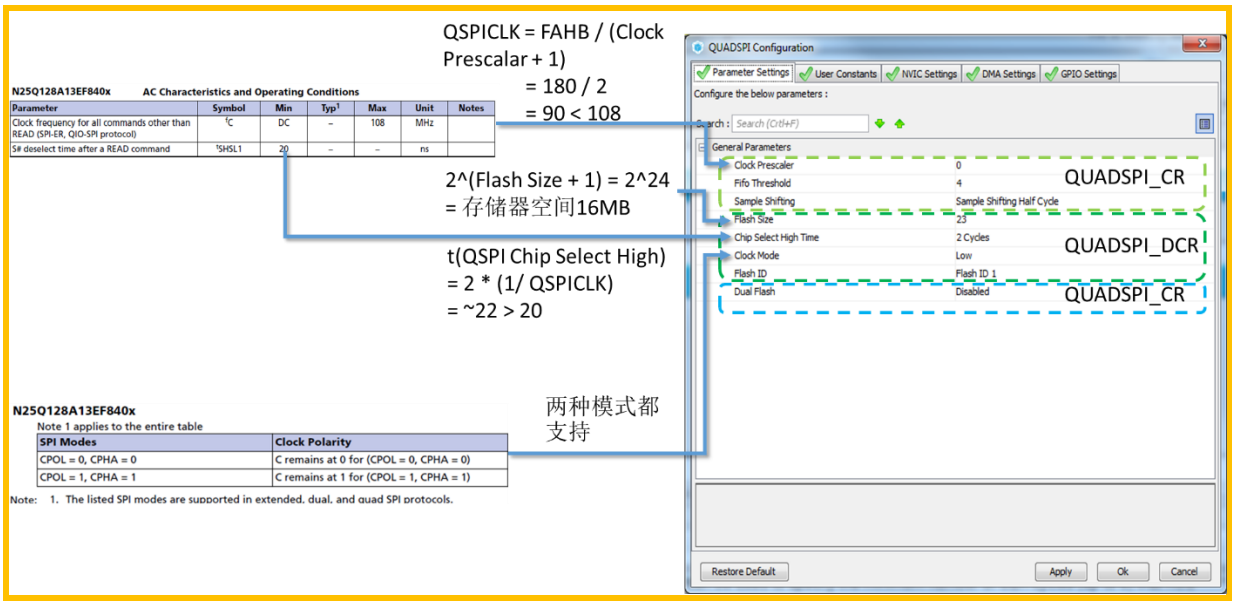
时钟配置如下图所示。设置输入时钟频率为 8MHz → 选择"HSE"做为 PLL 倍频时钟源 → 选择"PLLCLK"做为主频时钟源 → 设置 "HCLK"为 180MHz （FAHB 为 180MHz）→ 点击 Enter 键，自动生成对应主频的时钟参数（仅提供时钟配置参考，并不限制一定要设置 180MHz 主频）。



QSPI 配置如下图。参数的配置需要与存储器参数匹配。

• FIFO Threshold（FIFO 阈值） 配置为 4，并不严格要求。

• Smaple shift 选择“Sample shifting half cycle”。延后半个时钟，获取数据线上数据。可以使用在由于线路设计，数据信号存在较大延迟的场景。



N25Q128A13EF840x AC Characteristics and Operating Conditions

Parameter	Symbol	Min	Typ[1]	Max	Unit	Notes
Clock frequency for all commands other than READ (SPI-ER, QIO-SPI protocol)	f_C	DC	–	108	MHz	
S# deselect time after a READ command	t_{SHSL1}	20	–	–	ns	

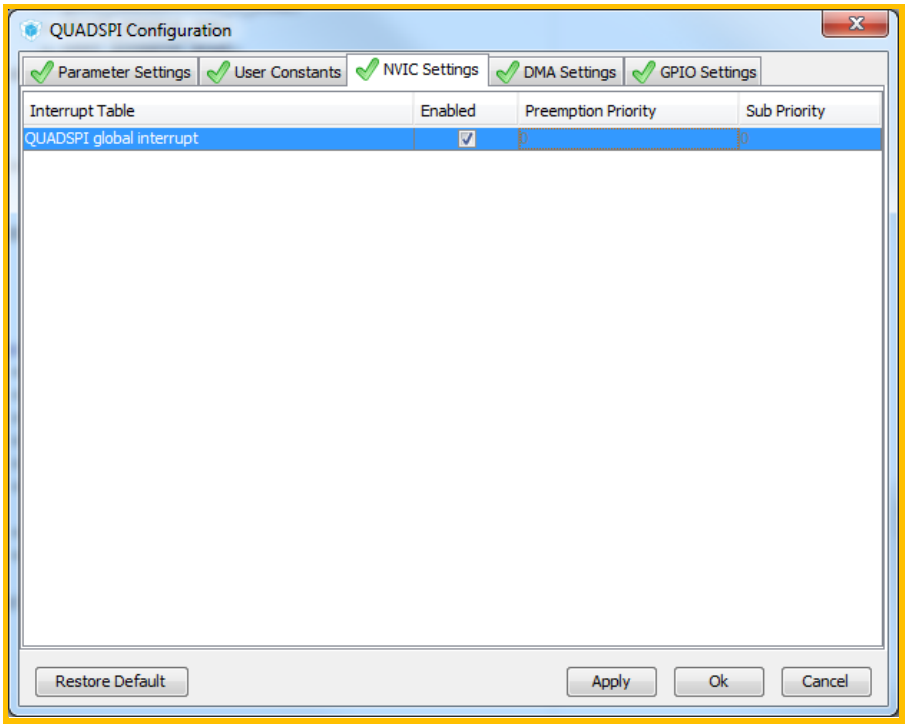
N25Q128A13EF840x

Note 1 applies to the entire table

SPI Modes	Clock Polarity
CPOL = 0, CPHA = 0	C remains at 0 for (CPOL = 0, CPHA = 0)
CPOL = 1, CPHA = 1	C remains at 1 for (CPOL = 1, CPHA = 1)

Note: 1. The listed SPI modes are supported in extended, dual, and quad SPI protocols.

使能 QSPI 中断。



点击菜单栏”Project” → “Settings” → 设置”Project Name” , “Project Location” 和 “Toolchain / IDE” 。其中“Toolchain / IDE”设置为 EWARM 以便生成对应 IDE 的工程。其他选项保持默认。

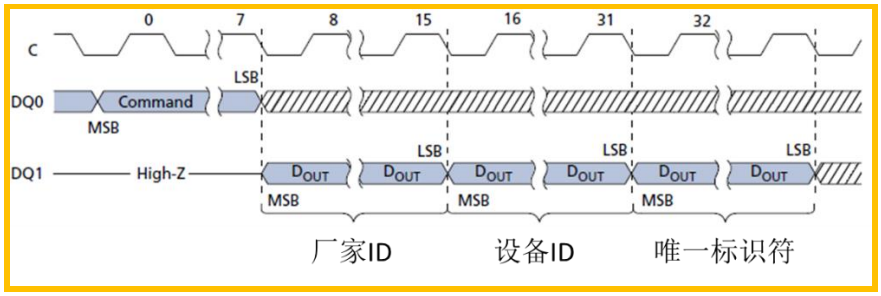
点击菜单栏”Project” → “Generate Code” → 等待 IAR 工程生成，出现”Code Generation”界面 → 点击”Open Project” 打开工程。

b. 完善工程。

由上述步骤获得的 IAR 工程中，包含了时钟配置及 QSPI 接口的初始化。对于外扩 Flash 的操作，还需要 添加外扩 Flash 支持的命令进行操作。N25Q128A13EF840F 支持的部分命令可参见本文 3.1 小结。

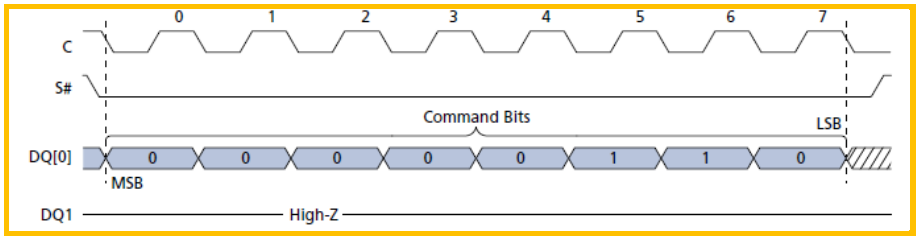
在这里出于简化考虑，仅提供了阻塞式读取 ID，擦除 Flash，块写和快读操作的实现。更多实现模式，可以参考 Cube 软件包中提供的 QSPI 例程。

在 N25Q128A13EF840F 手册中提供了读 ID 命令时序，如下图所示。

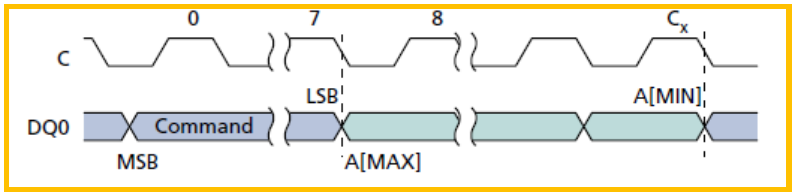


由时序图可知，读 ID 时序构成： 命令阶段 + 数据阶段。命令阶段和数据阶段线宽都为 1，读 ID 命令码为 0x9E 或者 0x9F，ID 数据长度为 17 字节。

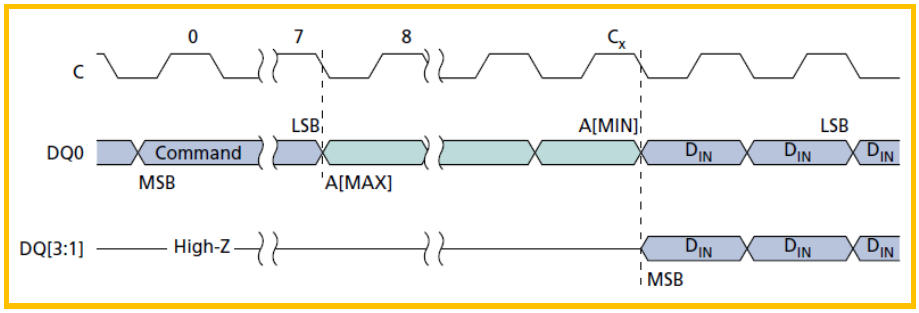
在 N25Q128A13EF840F 手册中提供了写使能命令时序，如下图所示。



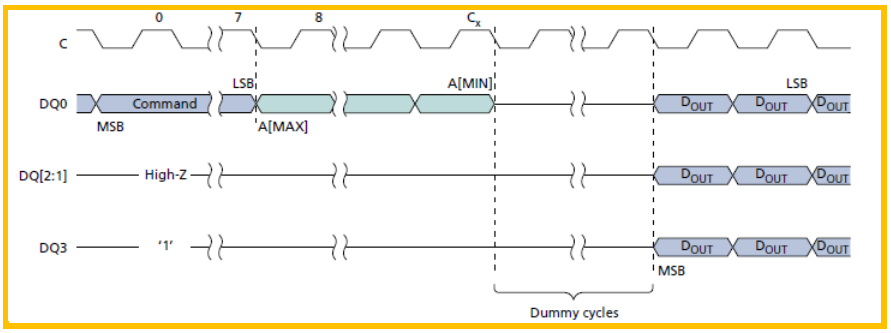
由时序图可知，块擦除时序仅有命令阶段。命令阶段线宽为 1，写使能命令码为 0x06。（注：这里仅呈现了单线命令模式的实现。除此之外，STM32 QSPI 接口和外扩存储器支持双线、四线模式）。
在 N25Q128A13EF840F 手册中提供了扇区擦除命令时序，如下图所示。



由时序图可知，扇区擦除时序构成：命令阶段+地址阶段。命令阶段和地址阶段线宽为 1，扇区擦除命令码为 0xD8。其中地址为 24-bit，任一位于需要进行擦除操作的扇区范围内地址都有效。在此简单选择扇区 0 进行擦除，选择地址为 0。（注：这里仅呈现了单线命令模式的实现。除此之外，STM32 QSPI 接口和外扩存储器支持双线、四线模式）。
在 N25Q128A13EF840F 手册中提供了四线快速写命令时序，如下图所示。



由时序图可知，四线快速写命令时序构成：命令阶段+地址阶段+数据阶段。命令阶段和地址阶段线宽为 1，数据阶段线宽为 4，四线快速写命令码为 0x32。其中地址为 24-bit，对应写入起始地址。（注：这里仅呈现了单线命令模式的实现。除此之外，STM32 QSPI 接口和外扩存储器支持双线、四线模式）。
在 N25Q128A13EF840F 手册中提供了四线快速读命令时序，如下图所示。



由时序图可知，四线快速读命令时序构成：命令阶段+地址阶段+数据阶段。命令阶段和地址阶段线宽为 1，数据阶段线宽为 4，四线快速读命令码为 0x6B。其中地址为 24-bit，对应写入起始地址。四线快速读命令默认 Dummy cycles 为 8。（注：这里仅呈现了单线命令模式的实现。除此之外，STM32 QSPI 接口和外扩存储器支持双线、四线模式）。
在 main.c \ main 函数中，增加代码如下。

```
…                   //系统、时钟、IO 和 QSPI 初始化
/* USER CODE BEGIN 2 */
QSPI_CommandTypeDef      sCommand;
static uint8_t Buf_ID[17] = {0};
static uint8_t TxBuf[0x10] = "Ext Flash", RxBuf[0x10] = {0};

sCommand.DdrMode            = QSPI_DDR_MODE_DISABLE;          //DDR 模式失能
sCommand.DdrHoldHalfCycle   = QSPI_DDR_HHC_ANALOG_DELAY;      //DDR 模式下，数据延迟输出
sCommand.SIOOMode           = QSPI_SIOO_INST_EVERY_CMD;       //每次发送都包含命令阶段
```

```
/***** 读 ID 操作 *****/
sCommand.Instruction       = 0x9F;    //READ ID 命令码
sCommand.InstructionMode   = QSPI_INSTRUCTION_1_LINE; //命令线宽
sCommand.AddressMode       = QSPI_ADDRESS_NONE;       //地址线宽。无地址阶段
sCommand.DataMode          = QSPI_DATA_1_LINE;        //数据线宽
sCommand.NbData            = 17;                      //读取数据长度。ID 长度为 17 字节
sCommand.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE; //无复用字节阶段
sCommand.DummyCycles       = 0;                         //无 Dummy 阶段
//配置命令（在有数据阶段时，命令在后续发送/接收 API 调用时发送）
if (HAL_QSPI_Command(&hqspi, &sCommand, 5000) != HAL_OK)
{
  Error_Handler();
}
//执行 QSPI 接收
if (HAL_QSPI_Receive(&hqspi, Buf_ID,5000) != HAL_OK)
{
  Error_Handler();
}
HAL_Delay(1);          //延时 1ms. 单位为 SysTick 定时中断周期

/***** 写使能操作（需要在块擦除之前，使外扩存储器处于写使能状态） *****/
sCommand.Instruction       = 0x06;                    //写使能 命令码
sCommand.InstructionMode   = QSPI_INSTRUCTION_1_LINE; //命令线宽
sCommand.AddressMode       = QSPI_ADDRESS_NONE;    //地址线宽。无地址阶段
sCommand.DataMode          = QSPI_DATA_NONE;          //数据线宽。无数据阶段
sCommand.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE; //无复用字节阶段
sCommand.DummyCycles       = 0;                       //无 Dummy 阶段
//配置发送命令
if (HAL_QSPI_Command(&hqspi, &sCommand, 5000) != HAL_OK)
{
  Error_Handler();
}

/***** 块擦除操作 *****/
sCommand.Instruction       = 0xD8;                    //扇区擦除 命令码
sCommand.InstructionMode   = QSPI_INSTRUCTION_1_LINE; //命令线宽
sCommand.AddressMode       = QSPI_ADDRESS_1_LINE;     //地址线宽。无地址阶段
sCommand.AddressSize       = QSPI_ADDRESS_24_BITS;    //地址长度
sCommand.Address           = 0;                       //位于所需擦除扇区内的任一地址。
sCommand.DataMode          = QSPI_DATA_NONE;          //数据线宽。无数据阶段
sCommand.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE; //无复用字节阶段
sCommand.DummyCycles       = 0;                         //无 Dummy 阶段
//配置发送命令
if (HAL_QSPI_Command(&hqspi, &sCommand, 5000) != HAL_OK)
{
```

```
  Error_Handler();
}
HAL_Delay(3000);      //延时 3s. 单位为 SysTick 定时中断周期

/***** 写使能操作（需要在块擦除之前，使外扩存储器处于写使能状态） *****/
sCommand.Instruction       = 0x06;                    //写使能 命令码
sCommand.InstructionMode   = QSPI_INSTRUCTION_1_LINE; //命令线宽
sCommand.AddressMode       = QSPI_ADDRESS_NONE;    //地址线宽。无地址阶段
sCommand.DataMode          = QSPI_DATA_NONE;          //数据线宽。无数据阶段
sCommand.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE; //无复用字节阶段
sCommand.DummyCycles       = 0;                       //无 Dummy 阶段
//配置发送命令
if (HAL_QSPI_Command(&hqspi, &sCommand, 5000) != HAL_OK)
{
  Error_Handler();
}

/***** 四线快速写操作 *****/
sCommand.Instruction       = 0x32;    //四线快速写 命令码
sCommand.InstructionMode   = QSPI_INSTRUCTION_1_LINE; //命令线宽
sCommand.AddressMode       = QSPI_ADDRESS_1_LINE;     //地址线宽
sCommand.AddressSize       = QSPI_ADDRESS_24_BITS;    //地址长度
sCommand.Address           = 0;                       //写入起始地址
sCommand.DataMode          = QSPI_DATA_4_LINES;        //数据线宽
sCommand.NbData            = 10;                      //写入数据长度
sCommand.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE; //无复用字节阶段
sCommand.DummyCycles       = 0;                        //无 Dummy 阶段
//配置命令（在有数据阶段时，命令在后续发送/接收 API 调用时发送）
if (HAL_QSPI_Command(&hqspi, &sCommand, 5000) != HAL_OK)
{
  Error_Handler();
}
//执行 QSPI 接收
if (HAL_QSPI_Transmit(&hqspi, TxBuf ,5000) != HAL_OK)
{
  Error_Handler();
}
HAL_Delay(5);         //延时 5ms. 单位为 SysTick 定时中断周期

/***** 四线快速读操作 *****/
sCommand.Instruction       = 0x6B;    //四线快速读 命令码
sCommand.InstructionMode   = QSPI_INSTRUCTION_1_LINE; //命令线宽
sCommand.AddressMode       = QSPI_ADDRESS_1_LINE;    //地址线宽
sCommand.AddressSize       = QSPI_ADDRESS_24_BITS;    //地址长度
sCommand.Address           = 0;                       //起始地址
sCommand.DataMode          = QSPI_DATA_4_LINES;        //数据线宽
```

```
  sCommand.NbData            = 10;                       //读取数据长度
  sCommand.AlternateByteMode = QSPI_ALTERNATE_BYTES_NONE; //无复用字节阶段
  sCommand.DummyCycles       = 8;                        //Dummy 阶段。N25Q128A13EF840F
Dummy cycles 默认为 15
  //配置命令（在有数据阶段时，命令在后续发送/接收 API 调用时发送）
  if (HAL_QSPI_Command(&hqspi, &sCommand, 5000) != HAL_OK)
  {
    Error_Handler();
  }
  //执行 QSPI 接收
  if (HAL_QSPI_Receive(&hqspi, RxBuf,5000) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE END 2 */
```

四 小结

STM32 的 QuadSPI 接口灵活可配，对于命令阶段、地址阶段、复用字节阶段、Dummy 阶段和数据阶段都可以进行配置。基于这种灵活性，能够实现市面上 SPI、Dual IO、Quad IO 的串行 Flash 支持。但出于简化考虑，QSPI 支持的中断访问及 DMA 访问等更多功能没有在本文进行介绍，更多实现可以参考 ST 提供的 Cube 软件包中的 QSPI 例程。另外，不同厂家的串行 Flash 命令及操作实现略有差异，具体以采用的 Flash 文档描述为准。

相关文档

AN4760 Quad-SPI (QSPI) interface on STM32 microcontrollers

AN4488 Getting started with STM32F4xxxx MCU hardware development

RM0386 STM32F469xx and STM32F479xx advanced ARM®-based 32-bit MCUs

相关工具&链接

STM32CubeMX http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html

STM32CubeF4 http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-embedded-software/stm32cubef4.html

重要通知 – 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。