恩智浦半导体
应用笔记

文档编号：AN12193
第 0 版，2018 年 5 月

使用 S32K148 QuadSPI 模块

作者：恩智浦半导体

1. 介绍

本应用笔记介绍了 S32K148 器件上的 QuadSPI 模块，描述了该模块如何在这些器件上实现，重点讲解如何设置 LUT 序列、使用命令与外部存储器进行通信，以及使用 AHB 接口。有关 QuadSPI 模块的详细信息，请参见相应的器件的参考手册。

本应用笔记有两个软件示例，一个是裸机代码示例，另一个是 SDK 示例。裸机示例可在所附的 zip 文件中找到，而 SDK 示例是 SDK 发布版的一部分。

2. QuadSPI 协议

四串行外设接口（QuadSPI）协议是一种通信协议，用于微控制器与外部闪存之间的通信。QuadSPI 基于流行的串行外设接口（SPI）。SPI 使用多达 4 类连接：数据输入、数据输出、时钟和片选（用于表示发送或接收处于活动状态），QuadSPI 使用时钟、多达 6 个片选通道和多达 4 个双向数据通道。这种额外的连接实现了更快地从闪存读取数据，使 QuadSPI 成为使用额外片外存储器的绝佳选择。

目录

由于引脚数量较少，读/写/擦除请求通过在总线上发送命令来实现。例如，要从闪存中读取数据，需要发送“Read Data（0xEB）”命令，并伴随发送要读取的 24 位地址。然后数据被发送到微控制器。下图显示了使用 4 条数据线的典型读取指令。

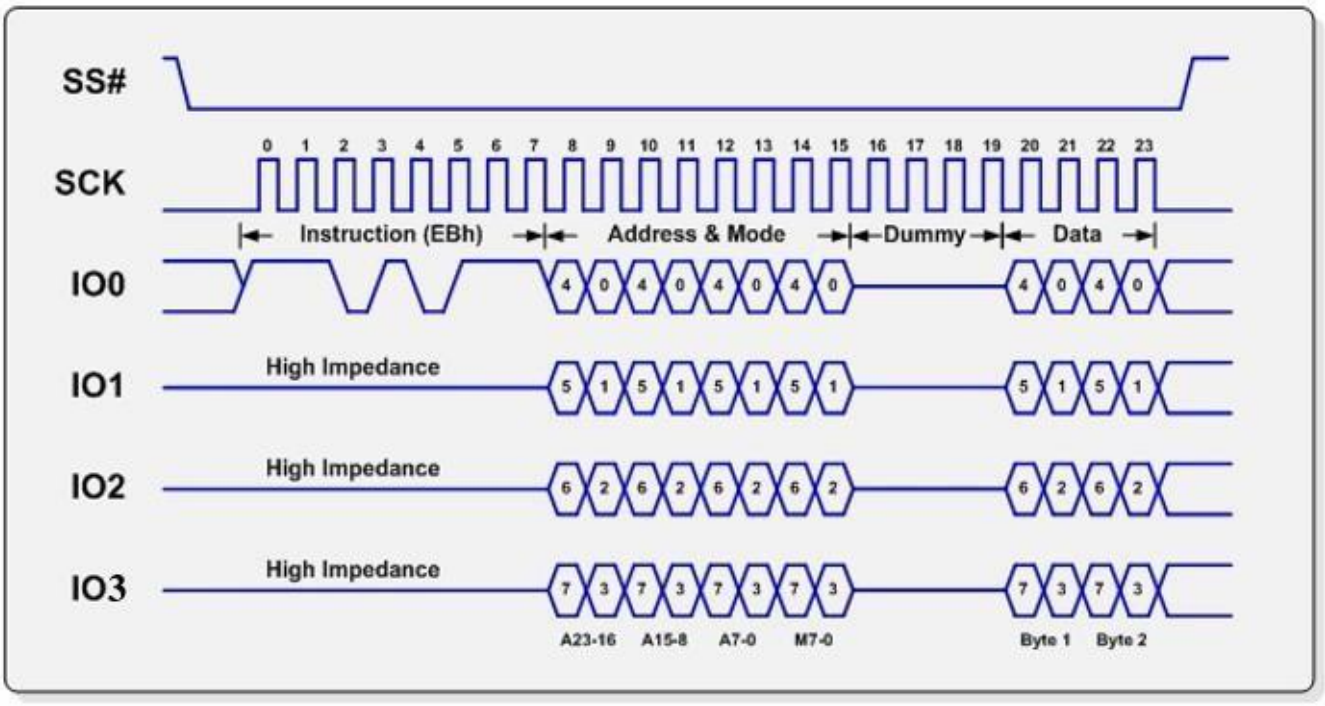

图 1. 读取命令（QuadSPI 帧）

QuadSPI 兼容存储器的供应商有好几家，如 Winbond、Spansion、Macronix 和 Numonyx。本应用笔记中提供的示例将重点介绍 Macronix 器件，因为 S32K148 EVB 上的外部存储器是 Macronix 生产的。QuadSPI 与之前的 SPI 一样，它不遵循一套既定标准，但各家制造商的器件会通过类似的命令集与之进行交互。

3. S32K148 QuadSPI 实现

以下部分将讲述 QuadSPI 模块的一些仅适用于 S32K148 的特性，这归因于 QuadSPI 在 S32K148 上独特的实现方式。

3.1. A 面和 B 面

QuadSPI 模块分为两个“面”（A 面和 B 面），主要原因是器件中可用的高速焊盘数量有限。每一面都有其优缺点，因此这要根据应用来选择。

A 面的主要优势在于速度，它支持高达 80 MHz 的频率。但是，它不支持 DDR，也不支持 Hyperbus 功能。

另一方面，B 面的运行速度较慢，最高为 20 MHz，但它支持 DDR 并支持适用于 HyperRAM 器件的 Hyperbus 协议。

需要澄清的是，尽管 QuadSPI 模块有两个“面”，但这并不意味着它可以像有两个单独的 QuadSPI 模块实例一样来实现。同一时间只能使用一面。

图 2. S32K148 引脚分配：QuadSPI 模块 A 面和 B 面

4. 查找表（LUT）功能

查找表（LUT）是 QuadSPI 模块与外部存储器通信的机制。它被用于发送命令、读取、写入或等待。该器件总共有 64 个 LUT 寄存器，这 64 个寄存器被分成 4 组，形成一个有效的序列。因此，QSPI_LUT[0]、QSPI_LUT[4]、QSPI_LUT[8]……QSPI_LUT[60]是各个有效序列的起始寄存器。

下表列出了一些最常用的 LUT 操作命令。要获取完整列表，参见参考手册中的**表 34-14 指令集**：

表 1. 常用 LUT 命令

命令	代码（6 位）
CMD	0x01
ADDR	0x02
DUMMY	0x03
MODE	0x04

命令	代码（6 位）
READ	0x07
WRITE	0x08
STOP	0x00

作为一种安全机制，默认情况下 LUT 表是锁定的。因此，使用 LUT 的第一步是将其解锁。要解锁它，就必须将密钥写入 LUTKEY 寄存器。密钥值为 0x5AF05AF0，然后必须将值 0x02 写入锁定配置寄存器。此时 LUT 就解锁了。下面的代码片段显示了在 S32K148 器件上 LUT 的解锁操作。

```
//Unlock the LUT
QuadSPI -> LUTKEY = 0x5AF05AF0;
QuadSPI -> LCKCR = 0x2; //UNLOCK the LUT
while(((QuadSPI -> LCKCR)&QuadSPI_LCKCR_UNLOCK_MASK)>>QuadSPI_LCKCR_UNLOCK_SHIFT == 0);
```

图 3. 解锁 LUT 代码

LUT 被解锁后，用户就可以修改 LUT 序列，同时要考虑到 QSPI_LUT[0]、QSPI_LUT[4]、QSPI_LUT[8] …… QSPI_LUT[60] 是各个有效序列的起始寄存器。LUT 的一些特性如下：

- 每个指令-操作数单元的宽度为 16 位。然而，LUT 寄存器的宽度为 32 位，因此每个 LUT 寄存器中可以放置两个指令。
- 根据 QSPI 事务的复杂程度，一个序列可能由单个或多个指令-操作数集组成。

每个 LUT 指令-操作数都具有以下结构：

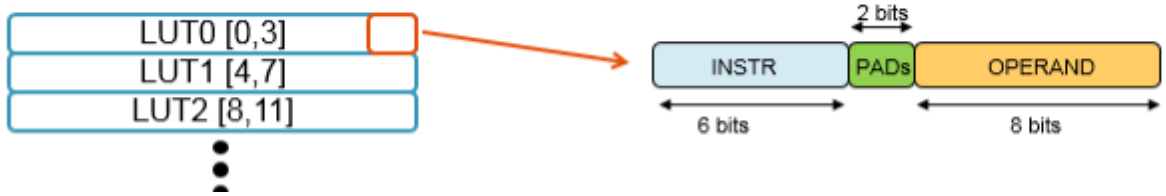

图 4. LUT 操作数结构

INSTR 字段表示前面表 1 列出的 LUT 命令，PADs 字段表示命令使用的数据线数量，操作数字段根据所使用的 INSTR 而变化，详情参见参考手册中的**表 34-14 指令集**。

例如，值 0x1C08。如果看二进制形式，则其值为 0b0001110000001000。如果将其划分为不同的字段：

INSTR						PADs		OPERAND							
0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0

INSTR = 0x07（读取）

PADs = 0x00（1 PAD）

OPERAND = 0x08（8 个字节）

当启动该序列时，模块将通过 1 条数据线读取 8 个字节的数据。

一旦所有的 LUT 序列都被填满，则必须再次锁定 LUT 表。锁定 LUT 的步骤与解锁 LUT 的步骤非常相似。必须将密钥写入 LUTKEY 寄存器。密钥的值为 0x5AF05AF0，然后必须将值 0x01 写入锁定配置寄存器。下面的代码片段显示了在 S32K148 器件上 LUT 的锁定操作。

```
//Lock the LUT
QuadSPI -> LUTKEY = 0x5AF05AF0;
QuadSPI -> LCKCR = 0x1; //LOCK the LUT
while(((QuadSPI -> LCKCR)&QuadSPI_LCKCR_LOCK_MASK)>>QuadSPI_LCKCR_LOCK_SHIFT == 0);
```

图 5. 锁定 LUT 代码

5. 外设总线（命令）接口

QSPI 模块提供两种路径与外部存储器进行通信：外设总线（下图左侧）或 AHB 总线（下图右侧）。本节将详细解释外设总线接口。

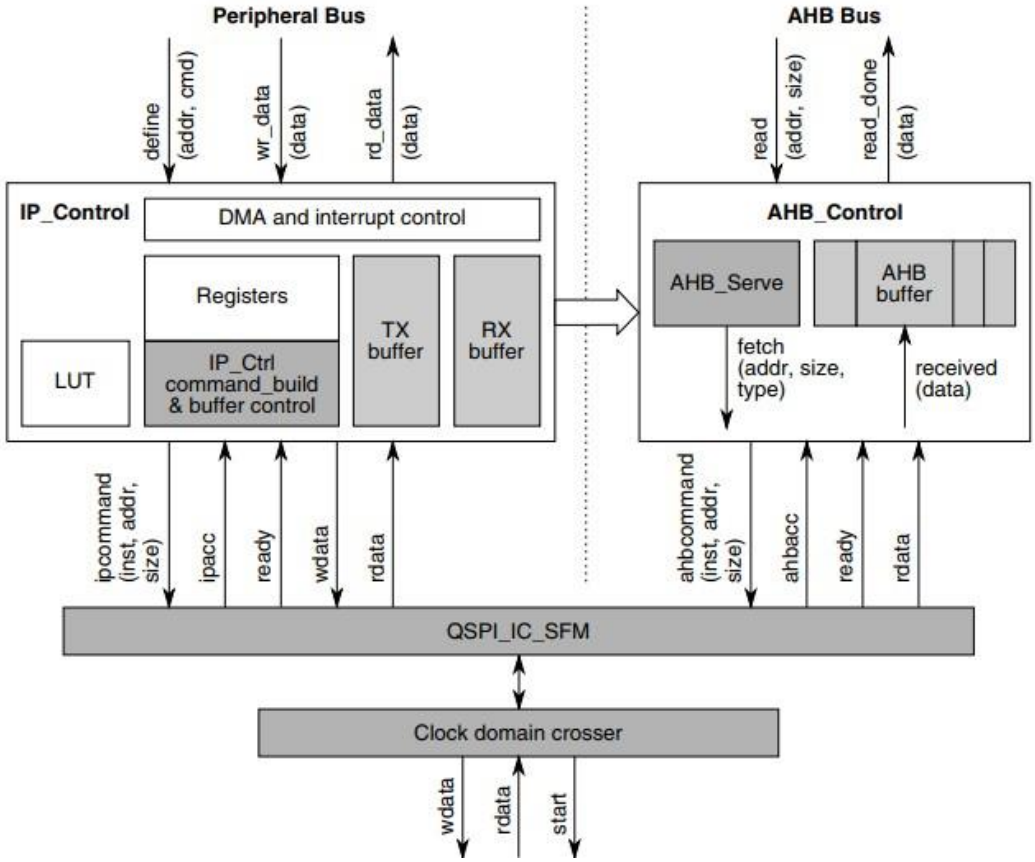

图 6. QSPI 框图

如果用户想写入、擦除或更改外部存储器的配置，唯一的选择是外设总线接口。它使用 LUT 表序列与外部存储器通信。一旦 LUT 表中填写了所需的 LUT 序列，用户就可以直接启动所需的序列。例如，假设 LUT 的填写如下图所示：

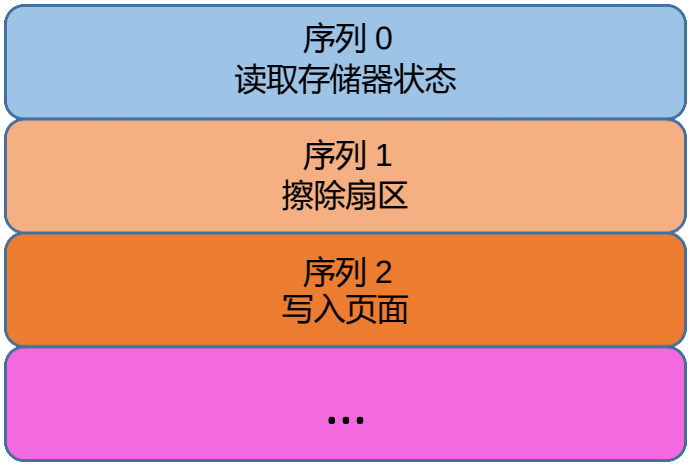

图 7. LUT 示例

序列 0 包含读取存储器状态所需的命令，序列 1 包含擦除所选扇区所需的命令，序列 2 包含写入存储器整个页面所需的命令。例如，如果用户想要擦除存储器的一个扇区，只需根据应用的需要多次调用 LUT 的序列 1 即可。以下代码片段说明了调用 LUT 表的一个序列是非常简单的。

```
void launch_loot(uint8_t loot_index){
    QuadSPI->IPCR = (loot_index>>2) << 24;
    while(QuadSPI->SR & QuadSPI_SR_BUSY_MASK);
}
```

图 8. 启动 LUT 序列代码

虽然启动一个 LUT 序列就能执行必要的命令（以便与存储器通信），但某些操作（如读取或写入）仍需要使用其他寄存器来正确接收或发送数据。例如，当读取时，MCU 将数据存储在内部数据缓冲区，该缓冲区的数据要通过 RBDR[0-31]寄存器访问。它还要求用户指定 SFAR 寄存器中要读取的地址，并清除 CLR_RXF 标志。下面的代码片段显示了一个函数，它能够从指定地址读取一定数量的字节（其数量可配置）。

```
void quad_quadspi_read(unsigned long address, unsigned long *dest, unsigned long size)
{
    int i,j;

    QuadSPI->SFAR = address;
    size = size/32;
    for(i = 0; i<size; i++)
    {
        QuadSPI->MCR |= QuadSPI_MCR_CLR_RXF_MASK;
        QuadSPI->FR = 0x10000;

        /* launch quad read command */
        launch_loot(QUAD_READ);

        while(((QuadSPI->RBSR & QuadSPI_RBSR_RDBFL_MASK)>>QuadSPI_RBSR_RDBFL_SHIFT)!=32);
        //RX buffer size is 32 words
        for(j = 0; j<32; j++)
        {
            *dest++ = QuadSPI->RBDR[j];
        }
        QuadSPI->SFAR = QuadSPI->SFAR + (32*4);
    }
    QuadSPI->MCR |= QuadSPI_MCR_CLR_RXF_MASK;
```

图 9. QuadSPI 读取代码

写入数据时也需要类似的考虑。用户必须首先填满一个数据缓冲区，然后根据命令中指定的数据量将其发送到外部存储器。缓冲区通过 TBDR[0-31]寄存器填写。与读取序列一样，用户必须指定在 SFAR 寄存器中要写入的地址。下面的代码片段显示了一个写入例程。

```
for(i=0;i<page_iterations;i++){
    quadspi_write_enable();
    QuadSPI -> MCR |= QuadSPI_MCR_CLR_TXF_MASK;
    QuadSPI -> FR = 0x08000000;
    m = remain_bytes/4; /* TBDR buffer is 4 bytes long */
    for(j = 0; j<m; j++)
    {
        QuadSPI->TBDR = *data++;
    }

    //set address
    QuadSPI->SFAR = base;

    /* Launch Page Program commmand */
    launch_loot(PAGE_PROGRAM);

    quadspi_wait_while_flash_busy();        //check status, wait to be done
    base += FLASH_PGSZ;
    if(size > FLASH_PGSZ){
        remain_bytes = FLASH_PGSZ;
        size = size - FLASH_PGSZ;
    }
    else{
        remain_bytes = size;
    }
}
```

图 10. QuadSPI 页面写入编程代码

6. AHB 接口

AHB 框图如图 6 右侧所示。AHB 接口与外设总线访问不同，AHB 接口只允许读取操作。然而，AHB 访问的主要优点是，它允许查看外部存储器，就像它被映射到器件的内部存储器地址一样，这意味着用户不需要执行任何 LUT 序列启动。在 S32K148 中，QuadSPI AHB 区域长度为 128 MB，映射到起始地址 0x68000000。例如，如果用户尝试使用外设总线访问存储器的地址 0x000000，则需要启动某个 LUT 序列，并且用户需要读取 QuadSPI 的 Rx 缓冲区以获得数据。另一方面，当使用 AHB 时，用户只需访问内存地址 0x68000000（QuadSPI 的起始地址）即可获得数据。下图显示了 QuadSPI 编入一些数据后，调试器访问存储器的情况，可以看出，数据是作为内部存储器读取的。

Address	0 - 3	4 - 7	8 - B	C - F
68000000	72B60B49	0A4A0A4B	094C094D	084E084F
68000010	B846B946	BA46BB46	BC460648	85460648
68000020	80470648	804762B6	00F0B4F8	FEE70000
68000030	00000000	00F00120	45000068	75000068
68000040	FFF7FEBF	80B400AF	084B094A	5A60074B
68000050	5B68064B	42F22012	1A60044B	4FF6FF72
68000060	9A60BD46	5DF8047B	704700BF	00200540
68000070	20C528D9	80B48BB0	00AF304B	3B62304B
68000080	7B61304B	FB60304B	FB61304B	3B61304B
68000090	BB60304B	BB61304B	7B60304A	304B9A42
680000A0	18D00023	7B620AE0	2D4A7B6A	52F82320
680000B0	2A497B6A	41F82320	7B6A0133	7B62294B
680000C0	9B087A6A	9A42EFD3	4FF0E023	234AC3F8
680000D0	082D04E0	4FF0E023	214AC3F8	082D09E0
680000E0	7B691A78	3B6A1A70	3B6A0133	3B627B69
680000F0	01337B61	FA687B69	9A42F1D1	09E03B69
68000100	1A78FB69	1A70FB69	0133FB61	3B690133
68000110	3B61BA68	3B699A42	F1D105E0	BB690022
68000120	1A70BB69	0133BB61	7A68BB69	9A42F5D1
68000130	2C37BD46	5DF8047B	704700BF	0004FE1F
68000140	1C020068	1C020068	0004FE1F	1C020068
68000150	1C020068	00000020	00000020	0000FE1F
68000160	00000000	00040000	80B400AF	074B084A
68000170	5A60064B	4FF6FF72	9A60044B	4FF40452
68000180	1A60BD46	5DF8047B	704700BF	00200540
68000190	20C528D9	80B582B0	00AF0023	7B60FFF7
680001A0	E3FF0D4B	4FF08042	C3F83421	0B4A0B4B
680001B0	5B6D43F4	80735365	094A094B	5B6943F4

图 11. QuadSPI 内存区域

AHB 访问使用 LUT 序列 0 作为默认读序列。因此，用户在尝试使用 AHB 访问之前，必须确保使用有效的读取命令对序列 0 进行编程，默认情况下，序列 0 使用典型值编程，进行简单（一条数据线）读取。

需要注意的一点是，由于以下因素，从外部存储器读取数据的速度要比从内部存储器读取数据的速度慢得多：

- 以 QuadSPI 时钟频率从外部存储器获取数据，而内部数据是以内核频率获取。
- 内部存储器数据在高速缓存范围内，而 QuadSPI 区域数据不在高速缓存范围内。
- QuadSPI AHB 缓冲区最多可配置为 4KB，访问 4KB 以外的数据将需要 QuadSPI 从外部存储器获取数据，这就会增加延迟。

使用 AHB 访问的另一个好处是，它允许从外部存储器执行代码，但如上所述，它的速度将显着变慢。

7. 软件示例

本应用笔记附带软件。该软件工程可以在 S32DS 中打开，并在 S32K148 EVB 上运行，使用板上可用的 MX25L6433F 外部存储器。该示例使用 AHB 和外设总线两种访问类型的例程，它使用预编译的应用程序对外部存储器进行编程，并通过读取它来验证它是否编写正确，这两个操作都使用外设总线访问。验证通过后，程序就使用 AHB 访问执行应用。该应用是一个简单的红色 LED 亮灭切换。

8. 参考资料

- AN5412，四串行外设接口（QuadSPI）模块更新
- AN4186，在 MPC56XXS 上使用 QuadSPI 模块
- AN5244，如何在 KL8x 系列上使用 QuadSPI

How to Reach Us:

Home Page:
nxp.com.cn

Web Support:
nxp.com.cn/support

Document Number: AN12193
Rev. 0
05/2018