在 mmWave SDK 增加 QSPI Flash 驱动的示例

Wesley He *Central FAE*

摘要

本应用手册介绍如何在 TI mmWave Radar SDK 中增加 QSPI Flash 的驱动代码并给出了实际示例代码，用户可参考本文的操作步骤，修改 mmWave SDK 的 QSPI flash 驱动，以适配更多的 flash 型号。

修改记录

Version	Date	Author	Notes
1.0	Sept 2022	Wesley He	First release

目录

表

1. QSPI Flash 支持列表

QSPI Flash 用于存储 TI mmWave Radar Sensor 的固件，用户需保证所使用的 QSPI Flash 被 SOC 的 ROM Bootloader 所支持，在使用 QSPI Flash 存储用户信息以及做二级 bootloader 时，也需要保证 mmWave SDK 的 QSPI flash 驱动能够支持所使用 QSPI Flash，Flash 选型的基本要求请参考文档：IWR6x43 Flash Variants Supported by the mmWave Sensor (Rev. C)，对于 TI 已经测试验证过的 Flash 型号如下列表，列表展示了适用于 xWR1642 ES2.0、xWR1842 ES1.0、xWR1443 ES3.0 和 IWR6843 ES1.0 和 ES2.0 设备的闪存部件。 表 1 显示了经过测试可以使用的 QSPI flash 闪存型号。

表 1. 推荐使用的 Flash 型号

Flash Vendor	Variant	Remarks
CYPRESS (SPANSION)	S25FL132K0XNFB01	QE bit set by ROM bootloader in SOP5 while flash programming
	S25FL064LVF01	QE bit set by ROM bootloader in SOP5 while flash programming. This flash variant supports extended addressing mode. All mmWave devices may not be compatible with the extended addressing mode. Refer to Section 1.3 at document
	MX25L3233F	QE bit set by ROM bootloader in SOP5 while flash programming
MACRONIX	MX25R1635FZNIH0	QE bit set by ROM bootloader in SOP5 while flash programming
	MX25V1635FZNQ	QE bit set by ROM bootloader in SOP5 while flash programming
	MX25U1633FZNQ	QE bit set by ROM bootloader in SOP5 while flash programming
	MX25V8035FM1Q	QE bit set by ROM bootloader in SOP5 while flash programming
	MX25U1633FZUI	Industrial grade 1.8-V flash
ISSI	IS25LP080D	QE bit set
WINBOND	W25Q16DVZPIG	By setting QE bit externally once

如果是其他的 Flash 型号，需满足如下几点要求：

- 有关通过 QSPI 接口与 SFLASH 的时序和接口要求的详细信息，请参阅 IWR/AWR 器件的数据手册。
- QSPI flash 器件变体应支持在 40-MHz 下操作所有命令（包括正常读取命令）。

QSPI flash 应支持 SFDP 命令，并可以按 JEDEC 标准的命令集进行响应。关键的指令字段如下表。

表 2. 关键指令

Field	Byte Offset
SFDP signature	[3-0]
JEDEC flash parameter offset in bytes	[0xE-0xC]
(1-1-4) Read support	[JEDEC flash parameter offset in bytes + 0x2] – bit6
(1-1-2) Read support	[JEDEC flash parameter offset in bytes + 0x2] – bit0
(1-1-4) Read command code	[JEDEC flash parameter offset in bytes + 0xB]
(1-1-4) Read dummy cycles	[JEDEC flash parameter offset in bytes + 0xA] – bit[4:0]
(1-1-2) Read command code	[JEDEC flash parameter offset in bytes + 0xD]
(1-1-2) Read dummy cycles	[JEDEC flash parameter offset in bytes + 0xC] – bit[4:0]
• The number of address bytes = 3 (always). • For single data line SPI read – Read Command Code (0xB), Read Dummy cycles (8bit).	

2. 修改 mmWave SDK 的 QSPI Flash 驱动

对于其他型号的 flash，需要从 flash 手册中获取相关的参数及配置信息，并且在 SDK Flash 驱动中进行修改及编译。有关于 Flash 存储器的所有信息，包含快速读取操作码、支持的擦除大小、每条指令所需的时钟数和存储器配置寄存器、如何配置 Qual read 等信息，都可以从 Flash 数据表中获得。

本文以 WINBOND W25Q16JVUXIQ 以及 Gigadevice GD25Q32CSIG 为例介绍如何在 mmWave SDK 中增加此款 Flash 的支持，两者的最大的区别是出厂默认 QE 为 enable 还是 disable。

- W25Q16JVUXIQ 是一款出厂默认 QE 位置为 1 的 Flash(with QE = 1 (fixed) in Status register-2)，存储于 flash 内的 APP BIN 可以被 ROM Bootloader 加载，如果应用中需要使用二级 bootloader 或者从 flash 中读写用户信息，则需要手动的修改 mmWave SDK 的 QSPI Flash 驱动以适配此款 flash 的指令。
- GD25Q32CSIG 是一款出厂默认 QE 位置为 0 的 Flash，ROM Bootloader 无法直接加载 Flash 中存储的 BIN，需要先使用第三方编程器或者 Arm 程序手动将 QE 置为 1（掉电非易失，只需做一次），随后 ROM bootloader 可以加载 flash 中存储的 BIN。Boot 成功后，后续应用中如需要使用二级 bootloader 或者从 flash 中读写用户信息，则需要手动的修改 mmWave SDK 的 QSPI Flash 驱动以适配此款 Flash 的指令。

SDK 驱动程序修改部分如下：

1. 在 qspiflash.h 中增加 flash ID 的支持：

```
#define WINBOND_DEV                     (0xEFU)
#define GIGADEVICE_DEV                  (0xc8U)
```

2. 在 qspiflash.c 中增加基于 flash ID 判断 flash 型号，并调用对应初始化程序对驱动进行初始化：

```
extern QSPIFlash_deviceFxn QSPIFlash_deviceFxnTbl_winbond;
extern QSPIFlash_deviceFxn QSPIFlash_deviceFxnTbl_gigadevice;

QSPIFlash_Handle QSPIFlash_open(QSPI_Handle QSPIHandle, int32_t *errCode)
{
…
    /* Get SPI flash device id */
    QSPIFlashGetDeviceID(QSPIHandle, FLASH_CMD_JEDEC_ID, &devId);

    if(devId.Manufacture == SPANSION_DEV)
    {
        ptrQSPIFlashDrv->devFxnTbl = QSPIFlash_deviceFxnTbl_spansion;
    }
    else if (devId.Manufacture == MACRONIX_DEV)
    {
        ptrQSPIFlashDrv->devFxnTbl = QSPIFlash_deviceFxnTbl_macronix;
    }
    else if (devId.Manufacture == WINBOND_DEV)
    {
        ptrQSPIFlashDrv->devFxnTbl = QSPIFlash_deviceFxnTbl_winbond;
    }
    else if (devId.Manufacture == GIGADEVICE_DEV)
    {
        ptrQSPIFlashDrv->devFxnTbl = QSPIFlash_deviceFxnTbl_gigadevice;
    }
…
}
```

3. 增加 qspiflash_device_gigadevice.c，增加对 gigadevice flash 的初始化代码及 Qual read mode 设置代码：

```
static void QSPIFlash_gigadevice_init(SPIFLASH_Params     *ptrFlashParams)
{
    if(ptrFlashParams->devID.Manufacture != GIGADEVICE_DEV)
    {
        DebugP_assert(0U);
    }
    /* Read Status commands */
    ptrFlashParams->readSR2Cmd = 0X35;
    ptrFlashParams->readSR3Cmd = 0X15;
    ptrFlashParams->writeStatusRegCmd = 0x31u;//FLASH_CMD_WR_SR;
    ptrFlashParams->readDataCmd = FLASH_CMD_FAST_SINGLE_READ;

    /* Protection bit mask in status register */
    ptrFlashParams->protectBitsMask = (uint8_t)0x7CU;
}
static void QSPIFlash_gigadevice_progQuadRead(QSPIFlash_Handle QSPIFlashHandle)
{
    uint32_t            configReg, newConfigReg;
    uint32_t            status;
    QSPI_Handle         QSPIHandle;
    SPIFLASH_Params     *ptrFlashParams;
```

```
    QSPIFLASH_Driver    *ptrQSPIFlashDrv;

    DebugP_assert(QSPIFlashHandle != (QSPIFlash_Handle)NULL);

    ptrQSPIFlashDrv = (QSPIFLASH_Driver *)QSPIFlashHandle;

    QSPIHandle =ptrQSPIFlashDrv->ptrQSPIHandle;

    /* Get flash parameter pointer */
    ptrFlashParams  = &ptrQSPIFlashDrv->flashParams;

    if(ptrFlashParams->devID.Manufacture != GIGADEVICE_DEV)
    {
        DebugP_assert(0U);
    }

    /* Read status register2 */
    configReg = QSPIFlashGetStatus( QSPIHandle, ptrFlashParams->readSR2Cmd);

    newConfigReg = configReg | (FLASH_CONFIG_QUADREAD_ENABLE);

    if(configReg != newConfigReg)
    {
        /* Make write enable flash before write or erase */
        QSPIFlash_writeEnable(ptrQSPIFlashDrv);

        /* Send write register command*/
        QSPIFlashWriteCmd(QSPIHandle, ptrFlashParams->writeStatusRegCmd, 2U);

        /*send configuratin register byte*/
        QSPIFlashWriteCmd( QSPIHandle, newConfigReg, 1U);

        do
        {
            status = QSPIFlashGetStatus(QSPIHandle, ptrFlashParams->readSR1Cmd);
        }while((status & (FLASH_STATUS_BUSY |FLASH_STATUS_WRENABLE )) != 0x0U);
    }
}
QSPIFlash_deviceFxn QSPIFlash_deviceFxnTbl_gigadevice =
{
    &QSPIFlash_gigadevice_init,
    &QSPIFlash_gigadevice_progQuadRead
};
```

4. 增加 qspiflash_device_winbond.c，增加对 flash 的初始化代码及 Qual read mode 设置代码。

```
static void QSPIFlash_winbond_init(SPIFLASH_Params     *ptrFlashParams)
{
    if(ptrFlashParams->devID.Manufacture != WINBOND_DEV)
    {
        DebugP_assert(0U);
    }
    /* Read Status commands */
    ptrFlashParams->readSR2Cmd = 0X35;
    ptrFlashParams->readSR3Cmd = 0X15;
    ptrFlashParams->writeStatusRegCmd = FLASH_CMD_WR_SR;
    ptrFlashParams->readDataCmd = FLASH_CMD_FAST_SINGLE_READ;
    /* Protection bit mask in status register */
    ptrFlashParams->protectBitsMask = (uint8_t)0x1CU;
}
static void QSPIFlash_winbond_progQuadRead(QSPIFlash_Handle QSPIFlashHandle)
{
    uint32_t            statusReg, newStatusReg;
    uint32_t            configReg;
    uint32_t            status;
    QSPI_Handle         QSPIHandle;
    SPIFLASH_Params     *ptrFlashParams;
```

```
    QSPIFLASH_Driver    *ptrQSPIFlashDrv;

    DebugP_assert(QSPIFlashHandle != (QSPIFlash_Handle)NULL);

    ptrQSPIFlashDrv = (QSPIFLASH_Driver *)QSPIFlashHandle;

    QSPIHandle =ptrQSPIFlashDrv->ptrQSPIHandle;

    /* Get flash parameter pointer */
    ptrFlashParams  = &ptrQSPIFlashDrv->flashParams;

    if(ptrFlashParams->devID.Manufacture != WINBOND_DEV)
    {
        DebugP_assert(0U);
    }

    /* Read status register1 */
    statusReg = QSPIFlashGetStatus(QSPIHandle, ptrFlashParams->readSR1Cmd);

    configReg = QSPIFlashGetStatus( QSPIHandle, ptrFlashParams->readSR2Cmd);

    newStatusReg = statusReg | FLASH_CONFIG_QUADREAD_ENABLE;

    if(statusReg != newStatusReg)
    {
        /* Make write enable flash before write or erase */
        QSPIFlash_writeEnable(ptrQSPIFlashDrv);

        /*send write register command*/
        QSPIFlashWriteCmd( QSPIHandle, ptrFlashParams->writeStatusRegCmd, 3U);

        /*send status register byte*/
        QSPIFlashWriteCmd(QSPIHandle, newStatusReg, 2U);

        /*send configuratin register byte*/
        QSPIFlashWriteCmd(QSPIHandle, configReg, 1U);

        do
        {
            status = QSPIFlashGetStatus(QSPIHandle, ptrFlashParams->readSR1Cmd);
        }while((status & (FLASH_STATUS_BUSY |FLASH_STATUS_WRENABLE )) != 0x0U);
    }
}

QSPIFlash_deviceFxn QSPIFlash_deviceFxnTbl_winbond =
{
    &QSPIFlash_winbond_init,
    &QSPIFlash_winbond_progQuadRead
};
```

5. 在 qspiflashlib.mak 中增加 qspiflash_device_winbond.c 及 qspiflash_device_gigadevice.c 的编译，确定其可被加入驱动库的编译。

```
QSPIFLASH_DRV_SOURCES += qspiflash_device_winbond.c
QSPIFLASH_DRV_SOURCES += qspiflash_device_gigadevice.c
```

3. 编译 QSPI FLASH 驱动库

完成 QSPI flash 驱动库修改后，需要重新对 flash 驱动库进行编译，可参考《mmwave sdk user's guide》文档，位于"C:\ti\mmwave_sdk_<ver>\docs\mmwave_sdk_user_guide.pdf" 章节 Building drivers/control/alg components 进行操作编译驱动库，打开 Windows 命令行，几条操作指令如下：

cd C:\ti\mmwave_sdk_<ver>\packages\scripts\windows\
call setenv.bat
cd C:\ti\mmwave_sdk_<ver>\packages\ti\drivers\qspiflash\

gmake all
即可编译得到 libqspiflash_<device>.aer4f，即是已添加新 flash 支持的 QSPI flash 驱动库。

4. 手动设置 Flash QE 位

上文章节提到，对于 Gigadevice GD25Q32CSIG 这种 QE enable bit 默认为 0 的 flash，需要首先将 flash 内部这一寄存器比特位设置为 1，这个比特位是掉电非易失的，只需设置一次。通常 QE 位的设置是借助第三方 QSPI flash 编程器，直接对 flash 的状态寄存器进行编程，即可完成设置。若用户手里没有合适的编程器，可以考虑使用 mmWave radar 软件代码对其进行配置，工作流程是：PC 编程&编译得到一个设置 QE=1 的 APP 程序 → 修改 uniflash 工作模式，使 APP 程序可通过 UART load 到 mmWave Radar 器件的 RAM 里运行 → APP 运行起来后将 flash 的 QE 位设置位 1 → 测试 flash 是否成功被 ROM bootloader 支持 →正常使用。

对于方法一，借助第三方 QSPI flash 编程器，直接对 flash 的状态寄存器进行编程，可参考第三方 QSPI flash 编程器的使用及说明文档进行操作。

对于方法二，借助 uniflash Load to RAM 的功能，使用 mmWave Radar SOC 软件对 QSPI flash 的配置方法如下：

1. 使用 HEX 编辑器修改编译好的 APP/SBL BIN，手动删除最后 4 位 CRC 结果。保存为新的 BIN，这个 BIN 需要包含将 GD25Q32 的 QE 位手动设置为 1 的功能。
2. 打开 UNIFLASH 目录：
   "C:\ti\uniflash_6.4.0\deskdb\content\TICloudAgent\win\ccs_base\mmWave\FlashPython.py"
3. 将 60 行去除注释。 propertiesMap['MemSelectRadio'] = 'SRAM'
   a. 此步骤位强制设置 uniflash load 到目标的 RAM。
4. 正常打开 uniflash 程序，接下来的步骤与常规 uniflash 烧写 QSPI flash 的操作步骤一致。在 SOP=101 模式下，将去除 CRC 校验的 BIN load 到 IWR6843 的 RAM 中，再次强调，此 BIN 需要手动包含设置 QE 位的功能。
5. 经过此操作后，GD25Q32 的 QE 将被设置为 1，ROM bootloader 可以成功 boot SBL 及 APP。
6. 使用完后，注意将 uniflash 的改动恢复为默认，否则无法正常烧写 BIN 到 flash 中。

5. 测试

完成全部驱动层代码修改，以及 QSPI flash 的 QE 位成功设置为 1 后，可对 SDK 中的 SBL 及 out-of-box demo 进行重新编译，测试 flash 驱动是否正常运行，可参考《mmwave sdk user's guide》文档，位于 "C:\ti\mmwave_sdk_<ver>\docs\mmwave_sdk_user_guide.pdf" 章节 Building demo 进行操作编译 SBL 及 out-of-box demo APP 的操作

- 重新编译 SBL
  - 打开 Windows 命令行，编译 SBL 的几条操作指令如下：
    cd C:\ti\mmwave_sdk_<ver>\packages\scripts\windows\
    call setenv.bat
    cd C:\ti\mmwave_sdk_<ver>\packages\ti\utils\sbl
    gmake all
- 重新编译 out-of-box demo APP
  - 打开 Windows 命令行，编译 SBL 几条操作指令如下：
    cd C:\ti\mmwave_sdk_<ver>\packages\scripts\windows\
    call setenv.bat
    cd C:\ti\mmwave_sdk_<ver>\packages\ti\demo\xwr68xx\mmw
    gmake all
- 烧写 xwr68xx_sbl.bin 及 xwr68xx_mmw_demo.bin 到 QSPI Flash 进行测试
  - 参考 SBL 的文档"C:\ti\mmwave_sdk_<Ver>\packages\ti\utils\sbl\docs\SBL_design.pdf"进行操作及测试

5.1. 测试结果

烧写编译出来的 SBL 及 APP 到 flash 后，直接加载并启动，打印信息如下，SBL 成功加载 APP 并跳转是成功的。

```
**********************************************
Debug: Secondary Bootloader Application Start
**********************************************
Press CR key or Space key to stop auto boot and Update Meta Image...
Loading existing Meta Image from Flash in   5  4  3  2  1
Debug: Device info: Manufacturer: c8, Device type = 40, Capacity = 16

Debug: Loading application metaImage from Flash address: c0040000
Debug: Parsing completed
****************************************
xWR68xx MMW Demo 03.02.01.02
****************************************
mmwDemo:/>
```

通过 SBL 更新 APP 固件并加载跳转 APP，打印信息如下，SBL 成功更新 flash 并加载 APP 跳转， flash 驱动的修改是成功的。

```
**********************************************
Debug: Secondary Bootloader Application Start
**********************************************
Press CR key or Space key to stop auto boot and Update Meta Image...
Loading existing Meta Image from Flash in   5
Debug: Update Meta Image selected

Debug: Device info: Manufacturer: c8, Device type = 40, Capacity = 16

Debug: Loading application metaImage from Flash address: c0040000
Debug: Erasing SFlash...
Debug: Total data written = 0x79480
Debug: Parsing completed
****************************************
xWR68xx MMW Demo 03.02.01.02
****************************************
mmwDemo:/>
```

由以上两项测试可知，QSPI flash 的基本读写功能已经被 mmWave SDK 所支持，用户可以自由根据使用情况，增加更多的功能。本应用手册介绍如何在 TI mmWave Radar SDK 中增加 QSPI Flash 的驱动代码并给出了实际示例代码，用户可参考本文的操作步骤，修改 mmWave SDK 的 QSPI flash 驱动，适配更多的 flash 型号。

6. 参考文献

1. IWR6843 Datasheet
2. mmWave Software Development Kit Ver 3.5.0.4
3. IWR14xx/16xx/18xx/68xx/64xx Industrial Radar Family Technical Reference Manual
4. IWR6x43 Flash Variants Supported by the mmWave Sensor (Rev. C)
5. 60GHz mmWave Sensor EVMs (Rev. D)
6. GD25Q32C datasheet
7. W25Q16JV datasheet

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265