Application Report
SWRA583–January 2017

Adding Flash Read and Write to an Existing mmWave Project

Dave Woodall

ABSTRACT

This application report describes the required steps to integrate usage of the QSPI flash on mmWave devices.

Contents

Trademarks

Code Composer Studio is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Initializing QSPIFlash Driver

The first step is to add code to include and initialize the QSPI and QSPIflash mmWave drivers. Both drivers are required for reading or writing to the flash. The following is C code that initializes both QSPI and QSPIFlash drivers. This tested code may be copied into the project.

```
#include <ti/drivers/qspiflash/qspiflash.h>

QSPI_Handle  qspiHandle;
QSPIFlash_Handle qspiflashHandle;

/* This is for the IWR1443. For IWR1642, simply substitute the equivalent pinmux macros. */

int32_t init_qspiflash(void)
{
  int32_t      errCode;
  QSPI_Params  QSPIParams;

  /* Setup the PINMUX to bring out the QSPI */
  Pinmux_Set_OverrideCtrl(SOC_XWR14XX_PINR10_PADAP,
PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_HW_CTRL);
  Pinmux_Set_FuncSel(SOC_XWR14XX_PINR10_PADAP,
SOC_XWR14XX_PINR10_PADAP_QSPI_CLK);
```

Copyright © 2017, Texas Instruments Incorporated

```
  Pinmux_Set_OverrideCtrl(SOC_XWR14XX_PINP8_PADAQ,
PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_HW_CTRL);
  Pinmux_Set_FuncSel(SOC_XWR14XX_PINP8_PADAQ,
SOC_XWR14XX_PINP8_PADAQ_QSPI_CS);

  Pinmux_Set_OverrideCtrl(SOC_XWR14XX_PINR11_PADAL,
PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_HW_CTRL);
  Pinmux_Set_FuncSel(SOC_XWR14XX_PINR11_PADAL,
SOC_XWR14XX_PINR11_PADAL_QSPI_D0);

  Pinmux_Set_OverrideCtrl(SOC_XWR14XX_PINP9_PADAM,
PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_HW_CTRL);
  Pinmux_Set_FuncSel(SOC_XWR14XX_PINP9_PADAM,
SOC_XWR14XX_PINP9_PADAM_QSPI_D1);

  Pinmux_Set_OverrideCtrl(SOC_XWR14XX_PINR12_PADAN,
PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_HW_CTRL);
  Pinmux_Set_FuncSel(SOC_XWR14XX_PINR12_PADAN,
SOC_XWR14XX_PINR12_PADAN_QSPI_D2);

  Pinmux_Set_OverrideCtrl(SOC_XWR14XX_PINP10_PADAO,
PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_HW_CTRL);
  Pinmux_Set_FuncSel(SOC_XWR14XX_PINP10_PADAO,
SOC_XWR14XX_PINP10_PADAO_QSPI_D3);

  /* Initialize the QSPI Driver */
  QSPI_init();

  /* Initialize the QSPI Flash */
  QSPIFlash_init();

  /* Open QSPI driver */
  QSPI_Params_init(&QSPIParams);

  /* Set the QSPI peripheral clock to 200MHz  */
  QSPIParams.qspiClk = 200 * 1000000U;

  QSPIParams.clkMode = QSPI_CLOCK_MODE_0;

  /* Running at 40MHz QSPI bit rate
   * QSPI bit clock rate derives from QSPI peripheral clock(qspiClk)
     and divide clock internally down to bit clock rate
     BitClockRate = qspiClk/divisor(=5, setup by QSPI driver internally)
   */

  QSPIParams.bitRate = 40 * 1000000U;

  qspiHandle = QSPI_open(&QSPIParams, &errCode);

  if (qspiHandle == NULL)
  {
    demo_printf("Error: Unable to open the QSPI Instance\n");
    return(0);
  }

  qspiflashHandle = QSPIFlash_open(qspiHandle, &errCode);
  if (qspiflashHandle == NULL )
  {
    demo_printf("Error: Unable to open the QSPIflash Instance\n");
    return(0);
  }

  return(1);
}
```

Copyright © 2017, Texas Instruments Incorporated

2 Obtaining Flash Address

Whether reading or writing, one must find a valid area or areas in flash to be used.

First, the mmWave SDK function returns the base address of the flash memory (per the current device):

```
  uint32_t   flashAddr;
  flashAddr = QSPIFlash_getExtFlashAddr(qspiflashHandle);
```

As shown in the TRM, for IWR1443 and IWR1642 this is:

```
EXT_FLASH          0xC000_0000        0xC07F_FFFF       8MB        MSS_QSPI (QSPI) flash
memory space
```

The next piece of information required is where the bootloader expects to find the flashed image. This is important so one does not use the same flash sectors.

2.1 Finding Bootloader Image Flash Offset

For Code Composer Studio™ (CCS), the following code is provided in the .projectspec file used to create the project:

```
<buildVariable name="LOAD_ADDRESS" value="0x200000"/>
```

For IWR1443 makefiles, the address is found in the main makefile: (eg. mmwave_sdk.mak for the demos)

```
LOAD_ADDRESS            = 0x200000
```

For IWR1642 projects, the meta image is flashed to offset zero.

You will see this in the link step of a build:

```
C:/ti/mmwave_sdk_01_00_00_05/packages/scripts/ImageCreator/xwr14xx/out2rprc/out2rprc.exe
level_sense_demo.xer4f level_sense_demo.bin 0x200000
```

The “generate bin” portion of the build process will provide you with the size of the image to be flashed:

```
C:/ti/mmwave_sdk_01_00_00_05/packages/scripts/ImageCreator/xwr14xx/append_bin_crc/gen_bincrc32.pl
level_sense_demo.bin

>>>> Binary CRC32 = 5a9b9b17 <<<<
>>>> Total bytes in binary file 71988 <<<<
```

In most cases one must account for the position and size of the flashed code image, or one may overwrite it. For safety, locations (offsets) 0x20_0000 to 0x28_0000 should be considered off limits for 14xx devices and 0x0 to 0x10_0000 for 16xx. Note that these locations and sizes may change in the future with new releases of the bootloader.

Copyright © 2017, Texas Instruments Incorporated

3 Writing to Flash

Before writing, the required portion of flash to use must be erased without erasing anything else. There are three APIs in the mmWave QSPIFlash driver that erase sections of flash memory. Table 1 shows the corresponding memory sizes that are erased.

Table 1. mmWave QPSIFlash Erase APIs

NAME	SIZE
QPSIFlash_sectorErase	4 KB
QPSIFlash_blockErase	64 KB
QPSIFlash_chipErase	Entire flash

The following code is a simple example that writes to the flash. If the size to be written exceeds 4 KB, QSPIFlash_sectorErase() must be called with adjoining 4 KB offsets, or QSPIFlash _blockErase() is used instead for 64 KB or larger areas.

```
#define DEMO_FLASH_OFFSET          0x100000
#define DEMO_FLASH_LEN                 (sizeof(my_struct))

  flashAddr = QSPIFlash_getExtFlashAddr(qspiflashHandle);

  flashAddr = flashAddr + DEMO_FLASH_OFFSET;

  /* Erase the 4KB sector to be written. */
  QSPIFlash_sectorErase(qspiflashHandle, flashAddr);

  QSPIFlash_singleWrite(qspiflashHandle, flashAddr, DEMO_FLASH_LEN,
                       (uint8_t *)my_data);
```

4 Reading From Flash

Reading from flash is easier than writing. After initializing the QSPIFlash driver, simply compute the correct offset into flash and call the read API with the following code:

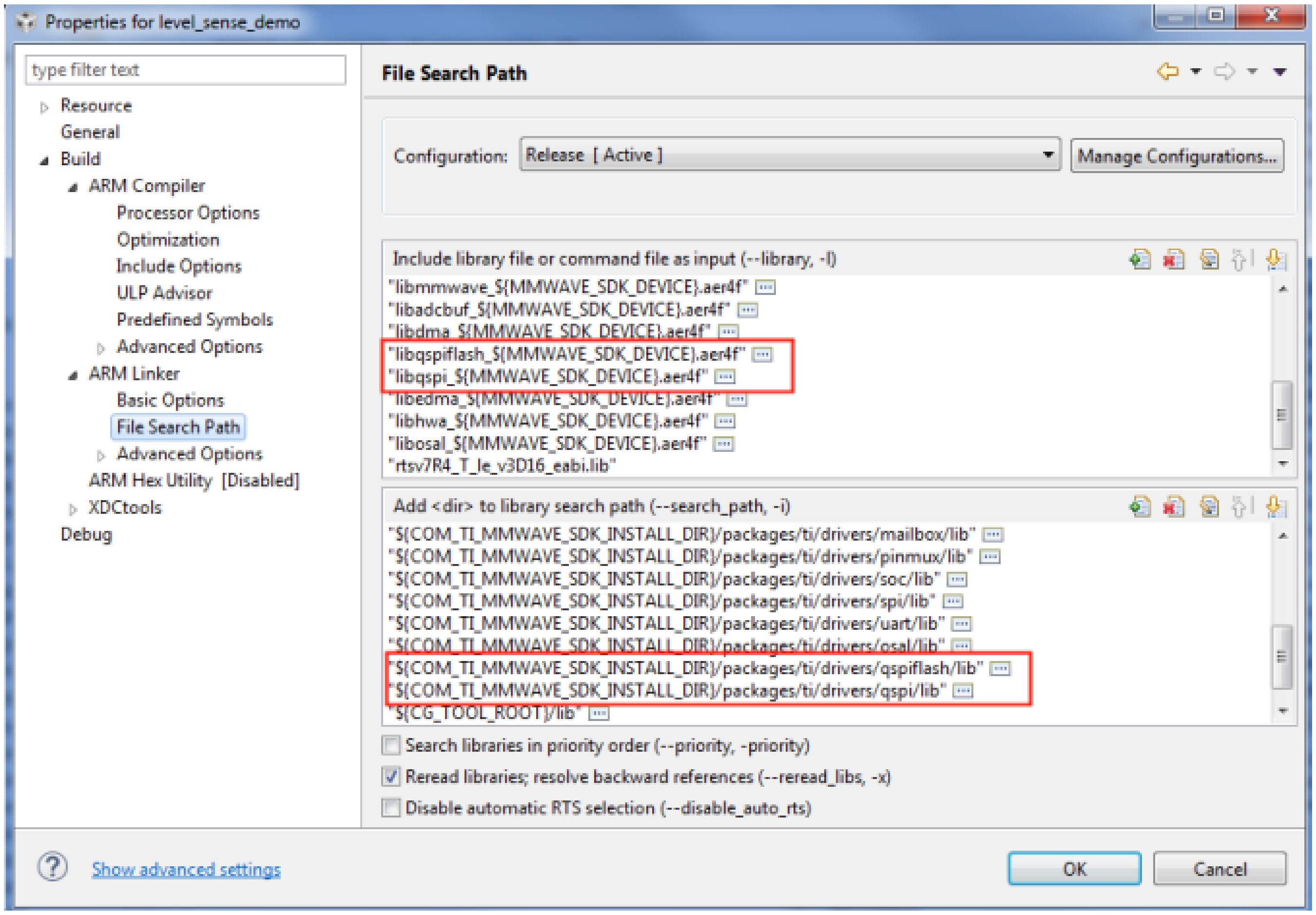
```
  /* Read my structure from flash. */
  flashAddr = QSPIFlash_getExtFlashAddr(qspiflashHandle);
  flashAddr = flashAddr + DEMO_FLASH_OFFSET;

  QSPIFlash_singleRead(qspiflashHandle, flashAddr, DEMO_FLASH_LEN,
                      (uint8_t *)my_data);
```

Copyright © 2017, Texas Instruments Incorporated

5 Linking QSPI Flash Drivers

The last step is to build the executable by linking with the QPSI drivers. If using a CCS project, the QPSI drivers can be added to the project's linker properties as shown in Figure 1.


Properties for level_sense_demo

File Search Path

Configuration: Release [Active]

Include library file or command file as input (--library, -l)

```
"libmmwave_${MMWAVE_SDK_DEVICE}.aer4f"
"libadcbuf_${MMWAVE_SDK_DEVICE}.aer4f"
"libdma_${MMWAVE_SDK_DEVICE}.aer4f"
"libqspiflash_${MMWAVE_SDK_DEVICE}.aer4f"
"libqspi_${MMWAVE_SDK_DEVICE}.aer4f"
"libedma_${MMWAVE_SDK_DEVICE}.aer4f"
"libhwa_${MMWAVE_SDK_DEVICE}.aer4f"
"libosal_${MMWAVE_SDK_DEVICE}.aer4f"
"rtsv7R4_T_le_v3D16_eabi.lib"
```

Add <dir> to library search path (--search_path, -i)

```
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/mailbox/lib"
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/pinmux/lib"
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/soc/lib"
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/spi/lib"
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/uart/lib"
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/osal/lib"
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/qspiflash/lib"
"${COM_TI_MMWAVE_SDK_INSTALL_DIR}/packages/ti/drivers/qspi/lib"
"${CG_TOOL_ROOT}/lib"
```

- [ ] Search libraries in priority order (--priority, -priority)
- [✓] Reread libraries; resolve backward references (--reread_libs, -x)
- [ ] Disable automatic RTS selection (--disable_auto_rts)

Figure 1. CCS Project Linker Properties

If using makefiles, do the same procedure there.

```
###############################################################################
# Additional libraries which are required to build the executable:
###############################################################################
STD_LIBS = $(R4F_COMMON_STD_LIB)                             \
                  -llibpinmux_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)       \
                  -llibdma_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)    \
                  -llibadcbuf_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)       \
                  -llibhwa_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)    \
                  -llibmailbox_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)      \
                  -llibedma_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)   \
                  -llibmmwave_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)       \
                  -llibmmwavelink_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)   \
                  -llibcrc_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)    \
                  -llibspi_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)    \
                  -llibqspi_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)   \
                  -llibqspiflash_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)    \
                  -llibuart_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)
```

Copyright © 2017, Texas Instruments Incorporated

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated (‘TI”) technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, “TI Resources”) are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI’s provision of TI Resources does not expand or otherwise alter TI’s applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED “AS IS” AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI’s standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated