TN00025

LPC54608 LCD Dual Frame Buffer with eXecute-In-Place (XIP) from Quad SPI flash

Rev. 1.0 — 08 November 2017 Technical note

Document information

Info	Content
Keywords	LPC54608, LCD, Dual frame Buffer, Quad SPI, XIP, SPIFI, 16bpp, SDRAM
Abstract	This technical note gives an overview of examples that uses LCD with code executing from Quad SPI flash.

Revision history

Rev	Date	Description
1.0	20171108	Initial version.

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

The LPC5460x is a family of ARM Cortex-M4 based microcontrollers for embedded applications. The LPCXpresso Development Board for LPC5460x MCUs is used in this technical note. See the following link for details of the board:

http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus/lpc54000-series-cortex-m4-mcus/lpcxpresso-development-board-for-lpc5460x-mcus:OM13092

Fig 1. LPC54608 LPCXpresso Development Board

This technical note gives an overview of the example developed using the on-board LCD and Quad SPI flash via SPIFI.

2. Description

The example shows the code execution and images loaded from Quad SPI flash through SPIFI direct memory accesses. A simple bootloader is required because the primary boot location is internal flash. Both application code and data images reside in 16 MB Quad SPI flash. The bootloader jumps to SPIFI XIP if it detects an application in Quad SPI flash.

2.1 LCD loaded with 16BPP (RGB565) with image data in Quad SPI flash

The example uses the LPC54608 internal flash, on board 16 MB Quad SPI flash, SDRAM and on-board 480 x 272 LCD screen. The LCD controller is configured for

16 bits per pixel (BPP) RGB565 TFT mode and implements dual frame buffers which reside in the on-board SDRAM.

When the example is executed from Quad SPI flash (XIP) via SPIFI at 0x1000_xxxx, a 480 x 272 image is loaded from SPIFI at 0x1002_0000. When a touch is detected on the LCD panel, the image on the LCD screen is updated with another image with the same dimensions. After displaying 12 images on the LCD screen, the application displays the same 12 images continuously.

The example is available in three tool chains:

- MCUXpresso IDE v10.0
- Keil MDK v5.24
- IAR Workbench v8.11

The Keil and IAR examples are found in:

lpc54608_lcd_spifi_xip_keil_iar\boards\lpcxpresso54608\demo_apps\lcd_spifi_xip

The MCUXpresso example can be found in the zip file:

lpc54608_lcd_spifi_xip_mcux.zip

2.2 Bootloader overview

IAR, Keil, and MCUXpresso use the same bootloader project build. IAR requires the ELF format (bootloader_spifi.out). The ELF format contains the address location for the tool to program to internal flash. Both Keil and MCUXpresso use binary format (bootloader_spifi.bin) where the address location is defined in the linker file.

The bootloader enables debug console and SPIFI IOs. Additional modules can be added as necessary. However, for this example two modules are adequate. The bootloader will setup the core and bus clock to maximum speed before setting up the SPIFI clock to speed up the execution, both in internal flash and XIP in Quad SPI flash. Then, it enables the Quad SPI to quad and read mode for allowing direct access or XIP. Finally, the program checks the value at 0x1000_0000 (stack pointer) and 0x1000_0004 (program counter) before jump to the new program counter value from 0x1000_0004.

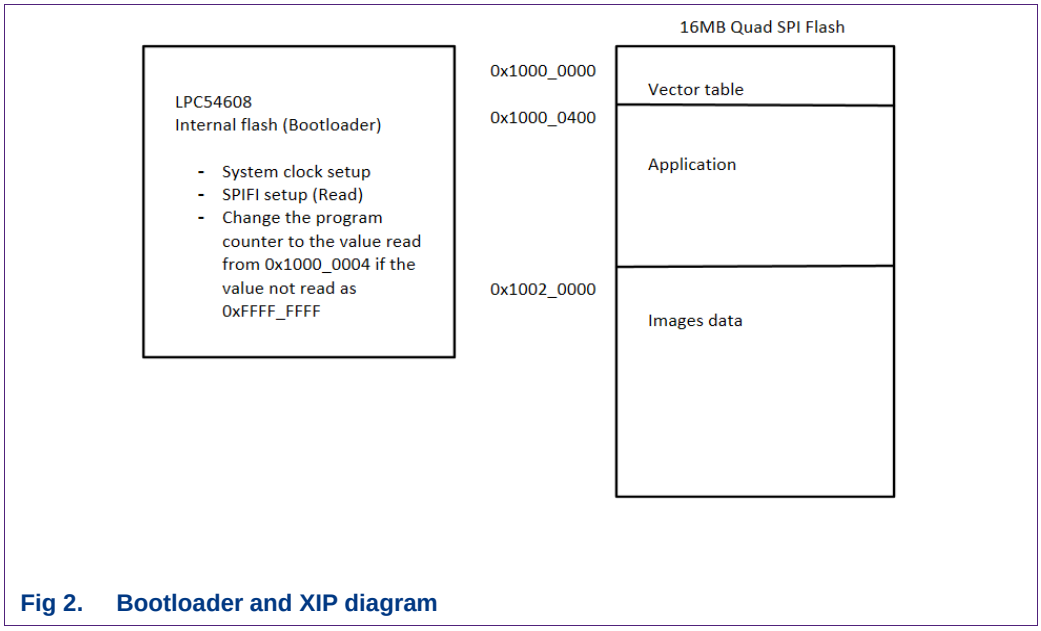

Fig 2. Bootloader and XIP diagram

```
1    /* Board pin, clock */
2    /* attach 12 MHz clock to FLEXCOMM0 (debug console) */
3    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);
4
5    /* attach 12 MHz clock to SPI3 */
6    CLOCK_AttachClk(kFRO12M_to_FLEXCOMM9);
7
8    BOARD_InitPins();
9    BOARD_BootClockFROHF96M();
10
11   /* Set SPIFI clock source */
12   CLOCK_AttachClk(kMAIN_CLK_to_SPIFI_CLK);
13   sourceClockFreq = CLOCK_GetSpifiClkFreq();
14
15   /* Set the clock divider */
16   CLOCK_SetClkDiv(kCLOCK_DivSpifiClk, sourceClockFreq / EXAMPLE_SPI_BAUDRATE,
     false);
17
18   /* Initialize SPIFI */
19   SPIFI_GetDefaultConfig(&config);
20   config.isReadFullClockCycle = 1; // enable this to make SPIFI reach 90MHz clock
     rate
21   SPIFI_Init(EXAMPLE_SPIFI, &config);
22
23   /* Enable Quad mode */
24   enable_quad_mode();
25
```

```
26   /* Setup memory command */
27   SPIFI_SetMemoryCommand(EXAMPLE_SPIFI, &command[READ]);
28
29   // check the program counter from SPIFI, if it is valid jump
30   if (( *((uint32_t *)0x10000000) != 0xFFFFFFFF) || ( *((uint32_t *)0x10000004) |=
     0xFFFFFFFF)) {
31        uint32_t jmp_adr = *((uint32_t *)0x10000004);
32        uint32_t (*fn_jump)(void) = (uint32_t(*)(void))jmp_adr;
33
34        (*fn_jump)();
35   }
36
37   while(1);
```

2.3 Program bootloader and application code to two different flash locations under one project

The following sections shows the user how to program the bootloader separately and allocate images data at fixed location with different tool chains.

The application project in different tool chains has the memory layout mapped at 0x1000_xxxx in linker script. The tool chains will program the application without additional step.

2.3.1 IAR

2.3.1.1 Program Bootloader via IAR built-in tool

From the Project menu, select Download then select Download file. See Fig 3. Navigate the project folder to <drive>\lpc54608_lcd_spifi_xip_keil_iar\boards\src\demo_apps\lcd_spifi_xip, select bootloader_spifi.out (ELF). IAR will start program the bootloader to LPC54608 internal flash.

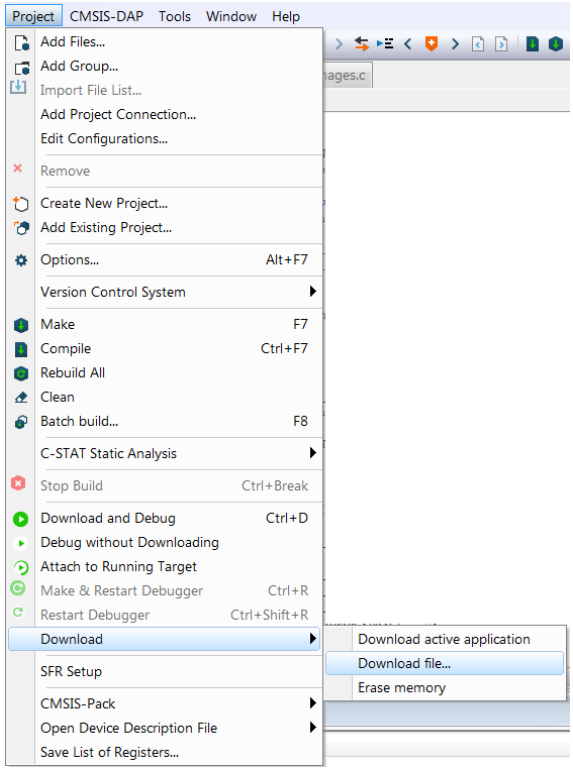

Fig 3. Downloading bootloader in ELF format

2.3.1.2 Include images in application project

The following example defines each address location for each variable in images.c and images.h.

Note: Preprocessor Symbol (IAR) is defined in C/C++ project option.

```
…
_Pragma(location=0x10020000)
GCC_IMGIN_SPIFI static GUI_CONST_STORAGE unsigned short image1[] = {…};
_Pragma(location=0x10060000)
GCC_IMGIN_SPIFI static GUI_CONST_STORAGE unsigned short image2[] = {…};
…
```

2.3.2 Keil

2.3.2.1 Bootloader and application in one project

The following example uses INCBIN directive in bootloader.s, defines a region for bootloader.o in scatter file (linker), and adds --keep=bootloader.o to ensure the binary are not opted out.

bootloader.s:

```
                AREA    BOOTLOADER_BIN, CODE, READONLY
                EXPORT  Bootloader_img
Bootloader_img
INCBIN          .\\bootloader_spifi.bin
Bootloader_img_End
                END
```

LPC54608J512_spifi.scf

```
…
LR_IROM1 0x00000000 0x00080000 {
  ER_IROM1 0x00000000 0x00080000 {
    bootloader.o (+RO-CODE)
  }
}
```

Add --keep=bootloader.o in Misc controls under Linker tab.

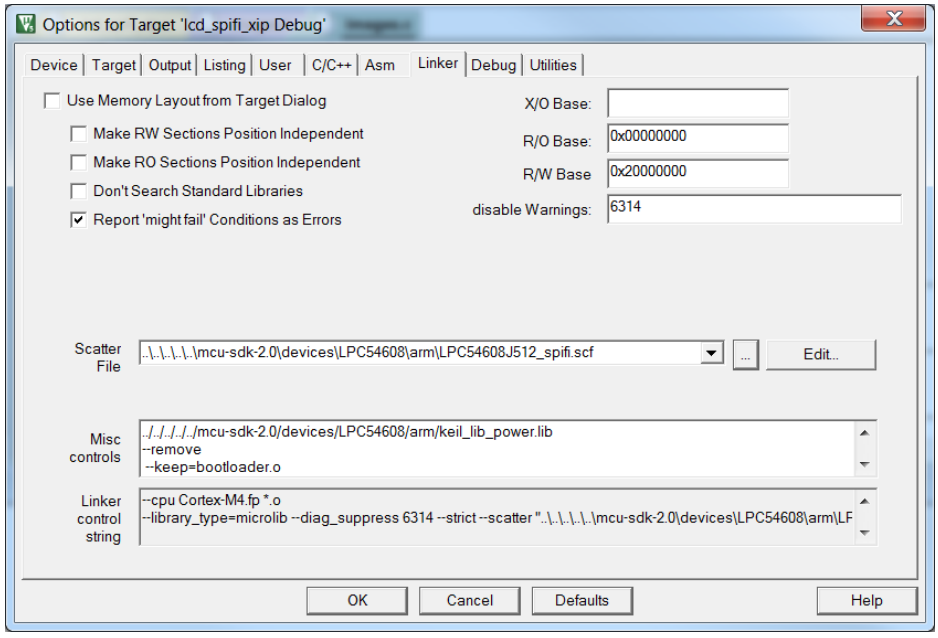

Fig 4. Adding –keep linker option

2.3.2.2 Include images in application project

This example uses the same method as IAR where each address location for each variable in images.c and images.h.

Note: Preprocessor Symbol (MDK) is defined in C/C++ project option.

```
…
__attribute__(( at(0x10020000) ))
GCC_IMGIN_SPIFI static GUI_CONST_STORAGE unsigned short image1[] = {…};
__attribute__(( at(0x10060000) ))
GCC_IMGIN_SPIFI static GUI_CONST_STORAGE unsigned short image2[] = {…};
…
```

2.3.3 MCUXpresso

Uncheck “Manage linker script”, and type ../lpcxpresso54608_lcd_spifi_xip.ld to Linker script edit box.

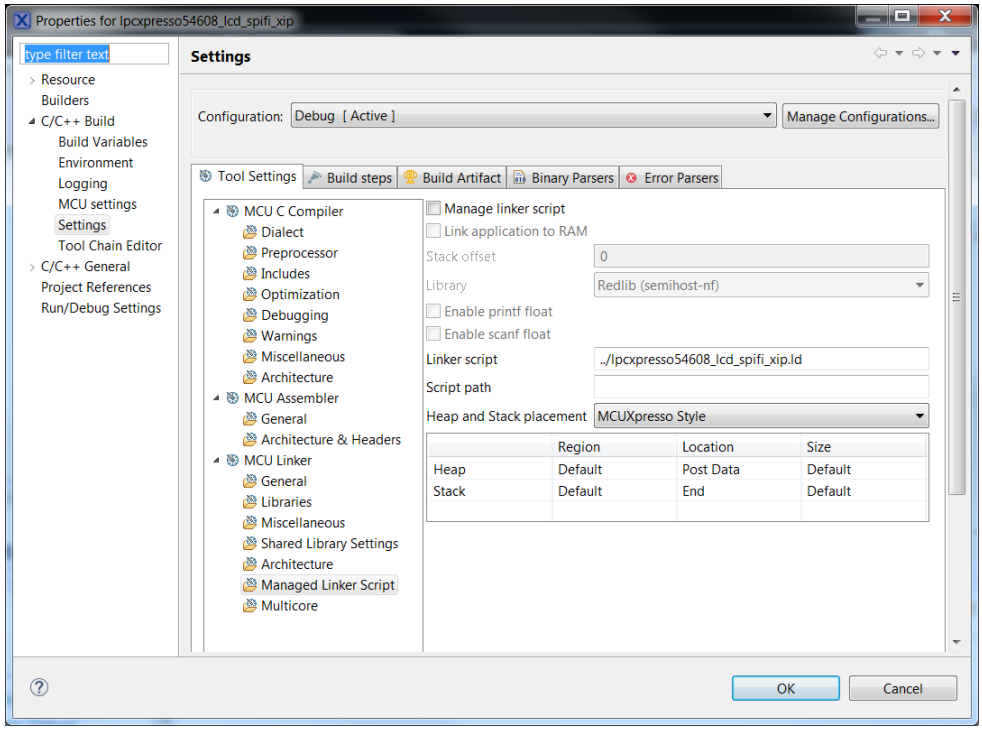

Fig 5. Adding Linker Script in MCUXpresso

2.3.3.1 Bootloader and application in one project

The following example also uses INCBIN in bootloader.s and defines the .data section in gcc linker file under .text_Flash2 for PROGRAM_FLASH (0x0000_0000 – 0x0008_0000).

bootloader.s:

```
          .global bootloader
Bootloader_start:
          .section .data
INCBIN    .\\bootloader_spifi.bin
Bootloader_End
```

lpcxpresso54608_lcd_spifi_xip.ld:

```
…
    .text_Flash2 : ALIGN(4)
    {
          KEEP(*(.data))
    } > PROGRAM_FLASH}
…
```

2.3.3.2 Include images as binary file in application project

The images data does not use array method in images.c, it uses the method as bootloader.s above. The Images_data.bin composes of 12 “C” stream (*.dta) from Bitmap Converter for emWin, and each image is 0x20000 aligned.

images_data.s:

```
          .global images_data

images_data_start:
          .section .imagesdata
          .incbin    "images_data.bin"
images_data_end:
```

lpcxpresso54608_lcd_spifi_xip.ld:

```
…
.text : ALIGN(4)
    {
        *(.text*)
        *(.rodata .rodata.* .constdata .constdata.*)
        . = ALIGN(4);
                . = ALIGN(0x20000);
                KEEP(*(.imagesdata))
    } > BOARD_FLASH
…
```

2.4 Issues

2.4.1 IAR linker warning message

Warning[Lt009]: Inconsistent wchar_t size

board.o and 36 other objects have wchar_t size 16 bits
GUI_DrawBitmap_565.o(libemWin_M4F.a) and 128 other objects have wchar_t size 32 bits

There is an update to wchar_t size from 16 bits to 32 bits in IAR version 8.11. Libraries that compiled using version older than 8.11 will have wchar_t size set to 16-bit. An update to the fsl_power_lib.a library to use 32 bits wchar_t size is planned and no issues with the demo are expected.

http://netstorage.iar.com/SuppDB/Public/UPDINFO/012120/arm/doc/infocenter/iccarm.ENU.html

2.4.2 Keil flash programming progress bar delay

When downloading or debugging the code, the progress bar during flash programming for Quad SPI appears to stop around 1-5%. Wait for ~30 secs up to 2 mins. The progress bar will reach 100% and complete the programming.

2.4.3 Raw binary

Attempt to generate a raw binary will create at least a 256 MB file due to internal flash region at 0x000_0000 and Quad SPI flash region at 0x1000_0000. In contrast to ELF and SREC format , raw binary does not contain header to isolate the two offsets.

3. References

See **LPC54608 LCD Dual Frame Buffer with images in Quad SPI flash** for the following information:

- Converting Bitmap file to image file in C.
- IAR reset type and macro setup.
- Adding 16MB Quad SPI flash to Flash programming in Keil.
- MCUXpresso project bring up.

4. Legal information

4.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

4.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

4.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

5. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 08 November 2017
Document identifier: TN00025