

Using the Chrom-ART Accelerator™ (DMA2D) to refresh an LCD-TFT display on STM32L496xx/L4A6xx microcontrollers

Introduction

The purpose of this application note is to highlight how to refresh a LCD-TFT display via the FSMC interface using the Chrom-ART Accelerator™ (DMA2D) on STM32L496xx/L4A6xx microcontrollers.

The STM32L496xx/L4A6xx microcontrollers implement a Chrom-Art Accelerator™ (DMA2D) that is a specialized DMA dedicated to image manipulation.

It can perform the following operations:

- Filling a part or the whole of a destination image with a specific color
- Copying a part or the whole of a source image into a part or the whole of a destination image with a pixel format conversion
- Blending a part and/or two complete source images with a different pixel format and copying the result into a part or the whole of a destination image with a different color format.

On the STM32L496xx/L4A6xx microcontrollers, the Flexible Static Memory Controller (FSMC) is used to access the LCD-TFT display with a parallel interface.

This application note provides an explanation on:

- How to connect the LCD-TFT display on the FSMC interface
- How to configure the DMA2D for the LCD-TFT display refresh

To fully benefit from this application note, the user should be familiar with the STM32 Chrom-ART Accelerator™ (DMA2D) as described in the *STM32L4x6 advanced ARM®-based 32-bit MCUs* reference manual (RM0351) available from the STMicroelectronics website www.st.com.

Reference documents

- *STM32L4x6 advanced ARM®-based 32-bit MCUs* reference manual (RM0351)
- *Discovery kit with STM32L496AG MCU* user manual (UM2160)
- Embedded software for the STM32L4 Series (STM32CubeL4)

Table 1. Applicable products

Type	Part numbers
Microcontrollers	STM32L496AE, STM32L496AG, STM32L496QE, STM32L496QG, STM32L496RE, STM32L496RG, STM32L496SL, STM32L496VE, STM32L496VG, STM32L496ZE, STM32L496ZG
	STM32L4A6AG, STM32L4A6QG, STM32L4A6RG, STM32L4A6VG, STM32L4A6ZG

Contents

- 1 Chrom-ART Accelerator™ (DMA2D) application use case overview . . 5**

- 2 LCD-TFT display on FSMC 6**
 - 2.1 Hardware interface description 6
 - 2.2 Display Command Set (DCS) software interface 7
 - 2.3 Controlling the D/CX signal with STM32L496xx/4A6xx microcontrollers . . 8

- 3 Chrom-ART Accelerator™ (DMA2D) configuration in STM32CubeL4 10**
 - 3.1 LCD partial refresh 10

- 4 Conclusion 12**

- 5 Revision history 13**

List of tables

Table 1.	Applicable products	1
Table 2.	FSMC signals	6
Table 3.	LCD-TFT signals	6
Table 4.	Minimum usable FSMC address bit depending on image size (16 bit RGB565 access) . . .	9
Table 5.	Document revision history	13

List of figures

Figure 1.	Display application typical use case	5
Figure 2.	Display bus interface specification	7
Figure 3.	Memory map for LCD-TFT display access	8
Figure 4.	Automatic control of LCD-TFT display data/command by FSMC interface	9

1 Chrom-ART Accelerator™ (DMA2D) application use case overview

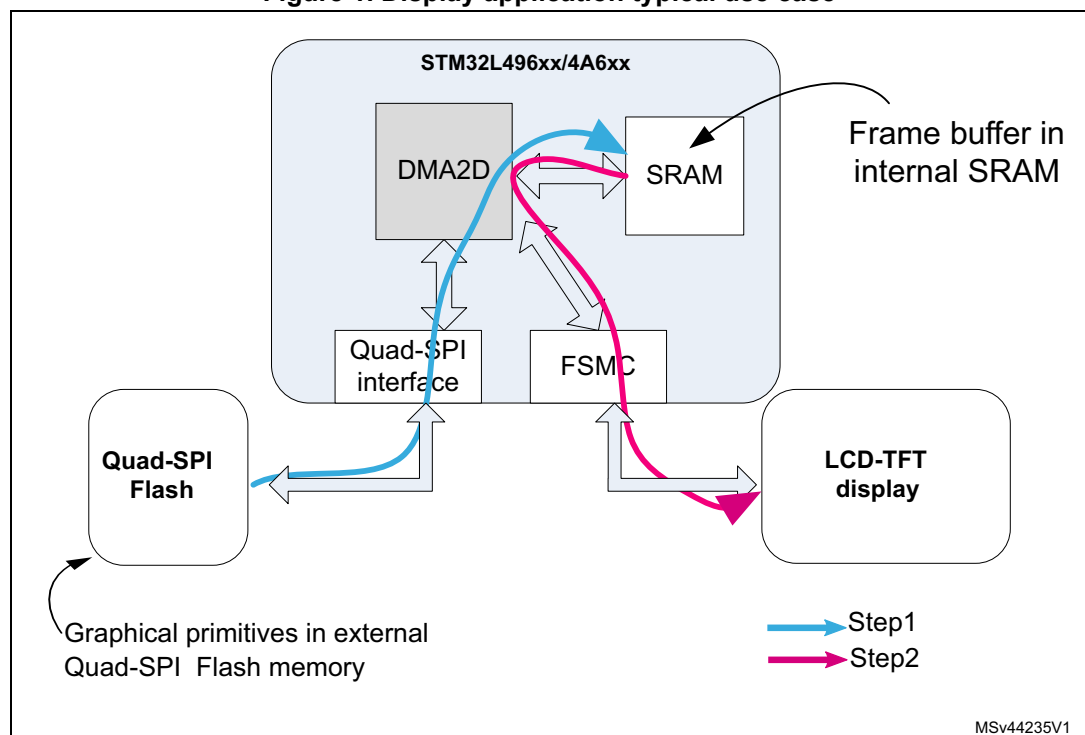
A typical application displaying an image into an LCD-TFT display is divided in 2 steps.

- Step1: creation of the frame buffer content
 - The frame buffer is built by composing graphical primitives like icons, pictures and fonts
 - This operation is done by the CPU running a graphical library software
 - It can be accelerated by a dedicated hardware used with the CPU through the graphical library (Chrom-ART Accelerator™ (DMA2D))
 - The more often the frame buffer is updated, the more fluid are the animations
- Step2: display of the frame buffer onto the LCD-TFT display
 - The frame buffer is transferred to the display through a dedicated hardware interface
 - The transfer can be done using the CPU, the system DMA or using the Chrom-ART Accelerator™ (DMA2D)

In a typical display application example using the STM32L496xx/L4A6xx microcontrollers, the Flexible Static Memory Controller (FSMC) is used as the hardware interface to the LCD-TFT display, the graphical primitives like pictures, icons or fonts are stored in the external Quad-SPI Flash memory and the frame buffer is stored in the internal SRAM. The transfer of the frame buffer to the LCD-TFT display can also be managed by the Chrom-ART Accelerator™ (DMA2D), hence not using the CPU or the DMA resources.

This is showed in [Figure 1: Display application typical use case](#).

Figure 1. Display application typical use case



The Chrom-ART Accelerator™ (DMA2D) can update the whole image on the display (full refresh) or only a part of it (partial refresh).

The configuration of the Chrom-ART Accelerator™ (DMA2D) (full or partial refresh) is done by programming specific registers through the high level HAL library function as shown in [Section 3: Chrom-ART Accelerator™ \(DMA2D\) configuration in STM32CubeL4](#).

2 LCD-TFT display on FSMC

2.1 Hardware interface description

The below signals are used to connect the Flexible Static Memory interface (FSMC) to the LCD-TFT display:

Table 2. FSMC signals

Signal name	FSMC I/O	Function
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x=1..4
NOE	O	Output enable
NWE	O	Write enable

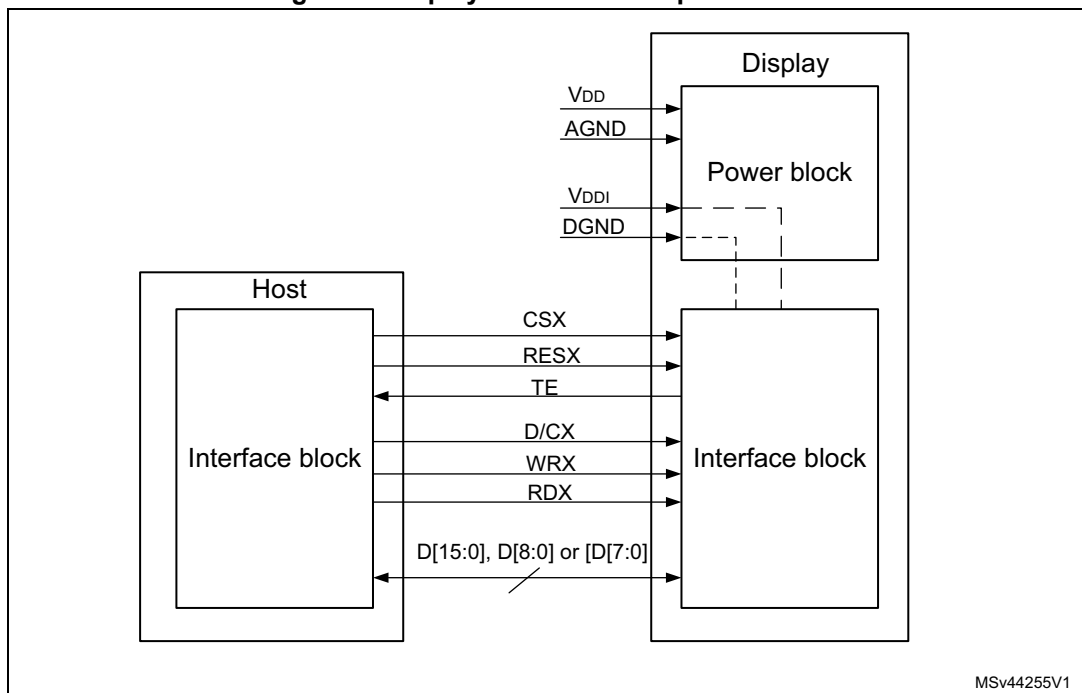
Table 3. LCD-TFT signals

Signal name ⁽¹⁾	LCD-TFT I/O	Function
D/CX	I	Data/command control signal
D[15:0]	I/O	Bidirectional information signals bus
CSX	I	Chip select control signal
RDX	I	Read control signal
WRX	I	Write control signal
TE	O	Tearing effect
RESX	I	Reset

1. The signal names are provided according the Type B Display Bus Interface (DBI) as described in the MIPI Alliance Standard for Display Bus Interface.

A typical connection is showed in [Figure 2](#)

Figure 2. Display bus interface specification



2.2 Display Command Set (DCS) software interface

The LCD-TFT displays can be controlled through the physical interface (here the FSMC bus) using software commands according to the Display Command Set (DCS) as defined in the MIPI Alliance Specification for DCS.

The DCS commands are used to configure the display module and to transfer the frame buffer to the display.

2.3 Controlling the D/CX signal with STM32L496xx/4A6xx microcontrollers

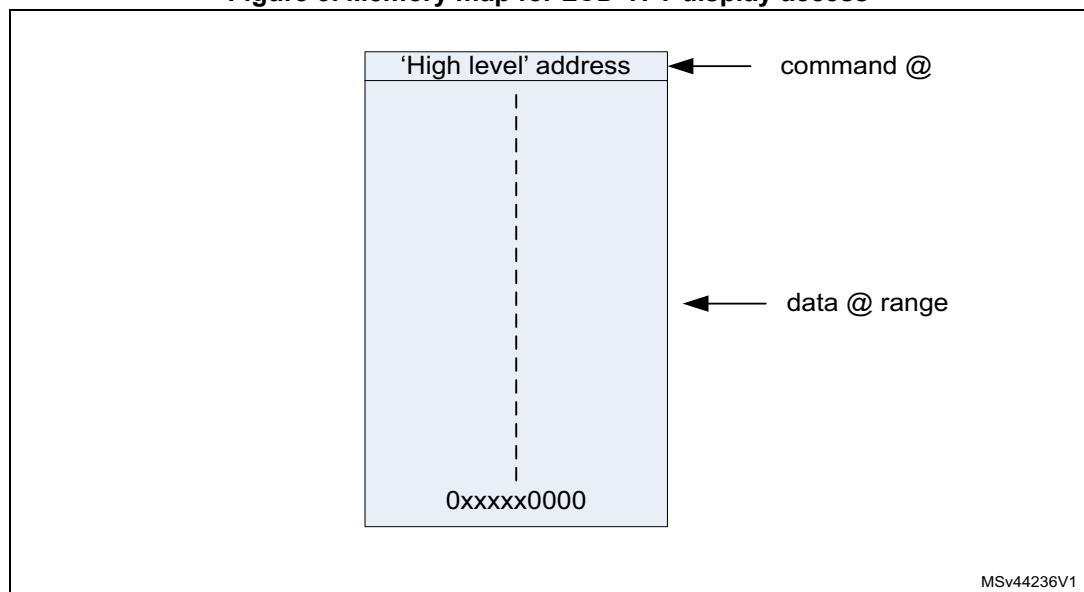
The D/CX signal of the DBI protocol is used to distinguish the commands (when D/CX = 0) from the data (when D/CX = 1) transfers.

There are 2 ways to control the 'Data/Command control' (D/CX) signal:

1. **By using a dedicated GPIO:**
 - Setting the 'Data/Command control' signal in "command mode" (setting the GPIO connected to the D/CX signal to '0' by software)
 - Sending the command
 - Setting the 'Data/Command control' signal in "data mode" (setting the GPIO connected to the D/CX signal to '1' by software)
 - Sending the data (frame buffer)
2. **By using an address bit of the FSMC address bus:**
 - Reserving a "low level" address in the memory map for the command transfer
 - Reserving the higher memory map range for the data transfer

When using the DMA2D to access the LCD-TFT display on FSMC interface, it is important to remember that even if the LCD-TFT display target is at a fixed address, the Chrom-ART Accelerator™ (DMA2D) increments the address bus of the transmitted data at each access (like a memory to memory access). Thus the FSMC address bus is incremented to cover the full data range address in the memory map.

Figure 3. Memory map for LCD-TFT display access

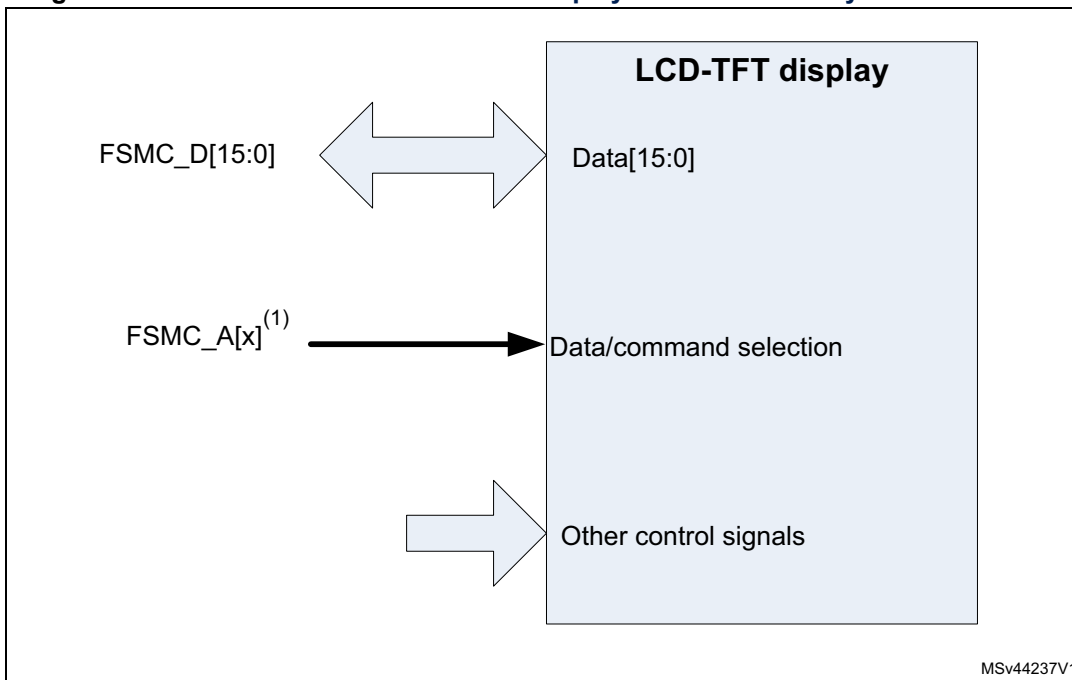


MSv44236V1

Using the 2nd option "an address bit of the FSMC address bus" makes the software simpler than using the 1st option with a dedicated GPIO, but it requires using the "high level" address to control the 'data or command select signal'.

- The user cannot use for example the FSMC address LSB bit (FSMC_A0) to control the 'data or command select signal'
- The user must use a "high enough" FSMC address bit in order to keep for this bit the same value during the whole image frame buffer transfer.

Figure 4. Automatic control of LCD-TFT display data/command by FSMC interface



1. 'x' as high as possible according [Table 4](#).

For example, if the image buffer size is 240x240 pixels and the transfer is done using 16 bits in RGB565 mode (one pixel transferred per access to LCD), the number of accesses are $240 \times 240 = 57600$ accesses and the FSMC address increments from `0x0000 0000` to `0x0000 E0FF`.

Thus the 1st address bit that does not change during the transfer is the bit 16.

In this specific case the `FSMC_A16` or a higher address bit can be used.

[Table 4](#) shows the minimum FSMC address bit that can be used depending on some image size.

Table 4. Minimum usable FSMC address bit depending on image size (16 bit RGB565 access)

Image size	Nb of pixels	Nb of accesses	Max address	Min usable FSMC address bit
VGA	640x480	307200	0x4AFFF	FSMC_A19
HVGA	480x320	153600	0x257FF	FSMC_A18
QVGA	320x240	76800	0x12BFF	FSMC_A17
-	240x240	57600	0x0E0FF	FSMC_A16

3 Chrom-ART Accelerator™ (DMA2D) configuration in STM32CubeL4

3.1 LCD partial refresh

An example configuring the DMA2D for an LCD partial refresh is provided in the STM32Cube examples:

STM32Cube_FW_L4\Firmware\Projects\STM32L496G-Discovery\Examples\DMA2D**DMA2D_MemToMemWithLCD**.

The code used to configure and start the DMA2D is shown below:

```

/* Configure LCD before image display: set first pixel position and image
size */
/* the position of the partial refreshed window is defined here. A rectangle
in the middle of the screen */
LCD_ImagePreparation((ST7789H2_LCD_PIXEL_WIDTH - LAYER_SIZE_X)/2,
(ST7789H2_LCD_PIXEL_HEIGHT - LAYER_SIZE_Y)/2, LAYER_SIZE_X, LAYER_SIZE_Y);
/*##-2- DMA2D configuration
#####*/
DMA2D_Config();
/*##-3- Start DMA2D transfer
#####*/
hal_status = HAL_DMA2D_Start_IT(&Dma2dHandle,
(uint32_t)&RGB565_240x160, /* Source buffer in format RGB565 and size
240x160 */
(uint32_t)&(LCD_ADDR->REG), /* LCD data address */
1, LAYER_SIZE_Y * LAYER_SIZE_X); /* number of pixel to transfer */
OnError_Handler(hal_status != HAL_OK);
...
...
...
/**
 * @brief DMA2D configuration.
 * @note This function Configure the DMA2D peripheral :
 * 1) Configure the transfer mode : memory to memory
 * 2) Configure the output color mode as RGB565
 * 3) Configure the transfer from FLASH to SRAM
 * 4) Configure the data size : 240x160 (pixels)
 * @retval
 * None
 */
static void DMA2D_Config(void)
{
    HAL_StatusTypeDef hal_status = HAL_OK;
    /* Configure the DMA2D Mode, Color Mode and output offset */
    Dma2dHandle.Init.Mode = DMA2D_M2M; /* DMA2D Mode memory to memory */

```

```

Dma2dHandle.Init.ColorMode    = DMA2D_OUTPUT_RGB565; /* Output color mode
is RGB565: 16 bpp */
Dma2dHandle.Init.OutputOffset = 0x0; /* No offset in output */
Dma2dHandle.Init.RedBlueSwap  = DMA2D_RB_REGULAR;    /* No R&B swap for
the output image */
Dma2dHandle.Init.AlphaInverted = DMA2D_REGULAR_ALPHA; /* No alpha
inversion for the output image */

/* DMA2D Callbacks Configuration */
Dma2dHandle.XferCpltCallback  = TransferComplete;
Dma2dHandle.XferErrorCallback = TransferError;

/* Foreground Configuration: Layer 1 */
Dma2dHandle.LayerCfg[1].AlphaMode = DMA2D_NO_MODIF_ALPHA;
Dma2dHandle.LayerCfg[1].InputAlpha = 0xFF; /* Fully opaque */
Dma2dHandle.LayerCfg[1].InputColorMode = DMA2D_INPUT_RGB565; /* Foreground
layer format is RGB565 : 16 bpp */
Dma2dHandle.LayerCfg[1].InputOffset = 0x0; /* No offset in input */
Dma2dHandle.LayerCfg[1].RedBlueSwap  = DMA2D_RB_REGULAR;    /* No R&B
swap for the input foreground image */
Dma2dHandle.LayerCfg[1].AlphaInverted = DMA2D_REGULAR_ALPHA; /* No alpha
inversion for the input foreground image */
Dma2dHandle.Instance = DMA2D;

/* DMA2D Initialization */
hal_status = HAL_DMA2D_Init(&Dma2dHandle);
OnError_Handler(hal_status != HAL_OK);
hal_status = HAL_DMA2D_ConfigLayer(&Dma2dHandle, 1);
OnError_Handler(hal_status != HAL_OK);
}

```

A full refresh is of course done in the same way but initializing the LCD 1st pixel at (0, 0) and the image size to the LCD size.

```

LCD_ImagePreparation(0, 0, ST7789H2_LCD_PIXEL_WIDTH,
ST7789H2_LCD_PIXEL_HEIGHT);

```

And changing the number of pixels to be transferred in the DMA2D start command:

```

hal_status = HAL_DMA2D_Start_IT(&Dma2dHandle,
(uint32_t)&RGB565_240x240, /* Source buffer in format RGB565 and size
240x240 */
(uint32_t)&(LCD_ADDR->REG), /* LCD data address */
1,
ST7789H2_LCD_PIXEL_HEIGHT * ST7789H2_LCD_PIXEL_WIDTH); /* number of pixel
to transfer */
OnError_Handler(hal_status != HAL_OK);

```

4 Conclusion

This application note gives a guideline to easily transfer images to an LCD-TFT display via the FSMC interface using the Chrom-ART Accelerator™ (DMA2D) without using the CPU or the DMA resources. A focus is given to the correct control of the 'Data/command control' signal of the LCD-TFT display. Some code examples have also been provided to setup the Chrom-ART Accelerator™ (DMA2D).

5 Revision history

Table 5. Document revision history

Date	Revision	Changes
27-Jan-2017	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2017 STMicroelectronics – All rights reserved