AN13509

Analysis of screen pushing scene based on i.MXRT595 MIPI DSI Controller

Rev. 0 — 09 February 2022 Application Note

Contents

1 Introduction

i.MXRT595 is a dual-core microcontroller combined a graphics engine and a streamlined Cadence® Tensilica® Fusion F1 DSP core with Arm® Cortex®-M33 core. It offers an embedded solution for smart watch or other display device due to the role of GPU.

Power consumption and frame rate are the considerations for performance evaluation during pushing screen. These two points are affected by many factors, such as the number and location of frame buffer and complexity of graphics. This document illustrates graphics drawing methods and performance comparisons under different conditions.

2 System architecture

The system is composed of i.MXRT595 - the main processor, a display module and several on-chip memories. On i.MXRT595 EVK, both PSRAM (APS6408L) and octal NOR flash (MX25UW51345) are connected to FLEXSPI interface as shown in block diagram (Figure 1). Also, the i.MXRT595 EVK board can be connected to display panels of different interface types. This document mainly concerns a 1.2-inch wearable display (G1120B0MIPI) with MIPI DSI interface, and its internal controller is RM67162.

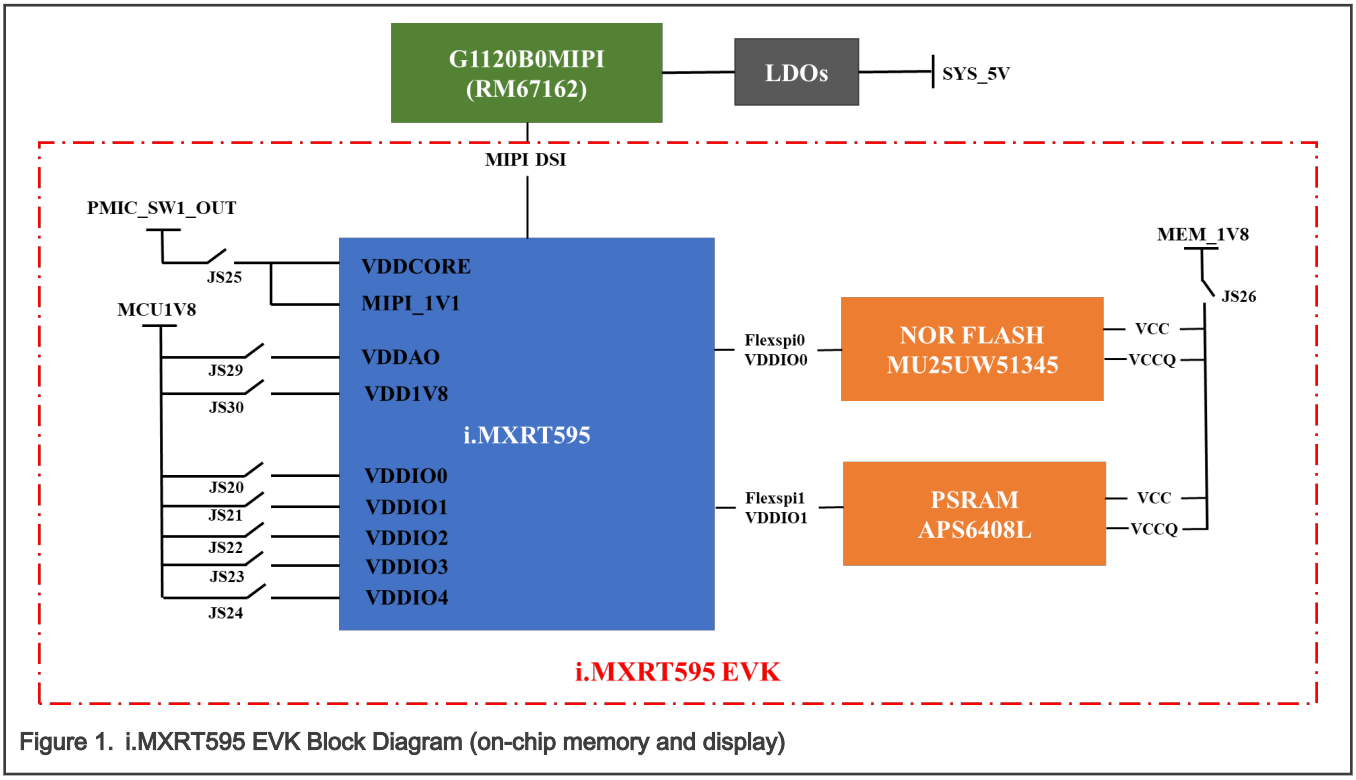

Figure 1. i.MXRT595 EVK Block Diagram (on-chip memory and display)

i.MXRT595 offers several types of peripherals supporting graphics display, including 2D Vector Graphics Processing Unit, LCD Display Interface, FLEXIO configured to 8/10/16/24-bit parallel interface and MIPI DSI Interface. Among them, GPU can render scalable vector graphics and compose and manipulate bitmaps.

APS6408L PSRAM which has octal SPI interface supports maximum 200 MHz DDR access. It is used mostly for data storage, especially those big and infrequent access data.

MU25UM1345G NOR flash is used for code execution and persistent data storage.

G1120B0MIPI is a 1.2 inch wearable display with 1-lane MIPI interface, 392x392 resolution. It is used for data and image displaying with MIPI DSI interface.

3 Graphics drawing and display

3.1 MIPI DSI and G1120B0MIPI module

3.1.1 Feature and layer

The MIPI DSI module in i.MXRT595 contains the MIPI DSI Controller and D-PHY. It provides a high-speed serial interface between the host processor and the display module. And it implements all layers and protocol functions defined in MIPI DSI specification. The features of MIPI DSI module are shown in Table 1.

Table 1. MIPI DSI module feature

Feature	Support		Remark
Operation Mode	Command Mode	Video Mode	Actual mode depends on display module
Data Lane Support	One or Two Lane(s)		Lane 0 - bidirectional or unidirectional Lane 1 - unidirectional
Transfer Mode	High-speed Mode	Low-power Mode	Video mode only matches with High-speed mode
Flexible Packet	APB interface option		Status and control
	DPI-2 option		Display Pixel Interface Core

Command Mode refers to that the host processor sends pixel information or instructions to a display module which incorporates a display controller in a serial manner, and reads the status or information from the display controller. The information is transmitted in a packet format and it requires a bidirectional interface. Video Mode refers to operation in which transfers from the host processor to the peripheral take the form of a real-time pixel stream.

MIPI DSI protocol contains below four layers in order from top to bottom layer, corresponding to D-PHY, DSI, DCS specifications.

1. Application
2. Low-Level Protocol
3. Lane Management
4. PHY Layer

Among them, MIPI DSI controller core realizes the upper three layers and MIPI DSI D-PHY provides a physical layer implementation for DSI.

G1120B0MIPI module incorporates a display controller which contains local registers and internal buffer. It is 1.2 inch circular AMOLED display, 392 * 392 pixels, with a 1-lane MIPI interface. And, its driver IC is RM67162, touch panel IC is FT3267.

3.1.2 Clock distribution

The display module used in this document is G1120B0MIPI. DPHY on i.MXRT595 supports up to 895.1 MHz bit clock. When choosing the clock frequency, consider the panel frame rate and resolution. From the RM67162 data sheet, the controller maximum bit rate is 320 Mbps when using 16-bit data format.

G1120B0MIPI module uses one pair of MIPI lanes. To get better performance, the MIPI clock source can be set to AUX1 PLL. Moreover, the clock source can use FRO for power optimization. See section 3.1.2.3 for details.

3.1.2.1 DSI host clock

The DSI Host Controller requires clocks as shown in Table 2.

Table 2. DSI Host Clock

Clock type	Clock alias	Description
Reference clock	ref_clk	Input clock of the D-PHY PLL module to generate the High-Speed MIPI clock and High-Speed Data Lane signaling
Byte clock	clk_byte	Generated by the DPHY DLL module, used for packet generation and transfer to the DPHY
LP mode clock	clk_tx_esc	For internal LPDT state machines and low-power transmissions
	clk_rx_esc	

Note:

1. $CLOCK_{DPHY\ PLL}$ = High-speed bit of the data lanes
2. $CLOCK_{clk\_byte}$ = 1/8 * (DPHY High-Speed data lane)
3. In this document, the DPHY PLL module is not used, so there is no need to consider ref_clk and the internal parameters of the DPHY PLL module. This is equivalent to distributing the clock to the output part of the DPHY PLL module, namely $CLOCK_{DPHY\ PLL}$.

3.1.2.2 D-PHY clock

The clock bus and data bus of D-PHY are differential structures. It supports two transmission modes, namely High-speed(HS) and Low-power(LP) mode. The application scenarios of different modes are shown in Table 3.

Table 3. D-PHY transmission mode

Transmission Mode	Application scenarios
HS mode	High-speed synchronous data transmission
LP mode	Control command transmission
	Low-speed asynchronous data transmission

Table 4 lists the clock signals of the D-PHY module.

Table 4. D-PHY clock

Clock type	Description
clk_byte	HS mode byte transmission clock. All data lanes share this signal
clk_tx_esc	Transmit clock in LP mode, 12 MHz-20 MHz
clk_rx_esc	Receive clock in LP mode, Maximum is 60 MHz

3.1.2.3 Clock distribution example

1. AUX1 PLL as MIPI clock source

   Example 1 shows the results of each clock distribution with AUX1 PLL as the clock source.

   - DPHY PLL(Input clock) = 316.8 M
   - BitClock = 316.8 M
   - RxClkEsc = 39.6 M
   - TxClkEsc = 19.8 M

   Example 1: AUX1 PLL as clock source

```
POWER_DisablePD(kPDRUNCFG_APD_MIPIDSI_SRAM);
POWER_DisablePD(kPDRUNCFG_PPD_MIPIDSI_SRAM);
POWER_DisablePD(kPDRUNCFG_PD_MIPIDSI);
POWER_ApplyPD();
CLOCK_AttachClk(kAUX1_PLL_to_MIPI_DPHY_CLK);
/* PFD value is in the range of 12~35. */
CLOCK_InitSysPfd(kCLOCK_Pfd3, 30);
/*DPHY PLL = 316.8MHz. */
CLOCK_SetClkDiv(kCLOCK_DivDphyClk, 1);
/* RxClkEsc max 60MHz, TxClkEsc 12 to 20MHz. */
CLOCK_AttachClk(kAUX1_PLL_to_MIPI_DPHYESC_CLK);
/* RxClkEsc = 316.8MHz / 8 = 39.6MHz. */
CLOCK_SetClkDiv(kCLOCK_DivDphyEscRxClk, 8);
/* TxClkEsc = 316.8MHz / 8 / 2 = 19.8MHz. */
CLOCK_SetClkDiv(kCLOCK_DivDphyEscTxClk, 2);
/* BitClk = 8 Byteclk = 8 * 1/8 * DPHY PLL = 316.8MHz. */
mipiDsiDphyBitClkFreq_Hz = CLOCK_GetMipiDphyClkFreq();
```

2. FRO as MIPI clock source

   Example 2 shows the results of each clock distribution with FRO as the clock source.

   - DPHY PLL(Input clock) = 192 M
   - BitClock = 192 M
   - RxClkEsc = 48 M
   - TxClkEsc = 16 M

   Example 2: FRO as clock source

```
POWER_DisablePD(kPDRUNCFG_APD_MIPIDSI_SRAM);
POWER_DisablePD(kPDRUNCFG_PPD_MIPIDSI_SRAM);
POWER_DisablePD(kPDRUNCFG_PD_MIPIDSI);
POWER_ApplyPD();
CLOCK_AttachClk(kFRO_DIV1_to_MIPI_DPHY_CLK);
```

```
CLOCK_SetClkDiv(kCLOCK_DivDphyClk, 1);
/* RxClkEsc max 60MHz, TxClkEsc 12 to 20MHz. */
CLOCK_AttachClk(kFRO_DIV1_to_MIPI_DPHYESC_CLK);
/* RxClkEsc = 192MHz / 4 = 48MHz. */
CLOCK_SetClkDiv(kCLOCK_DivDphyEscRxClk, 4);
/* TxClkEsc = 192MHz / 4 / 3 = 16MHz. */
CLOCK_SetClkDiv(kCLOCK_DivDphyEscTxClk, 3);
/* BitClk = 8 Byteclk = 8 * 1/8 * DPHY PLL = 192MHz. */
mipiDsiDphyBitClkFreq_Hz = CLOCK_GetMipiDphyClkFreq();
```

3.2 Utilize GPU for graphics drawing

i.MXRT595 contains a 2D Vector Graphics Processing Unit(GPU). To draw graphics, first the CPU sends instructions to the GPU. And, after receiving the instructions, the GPU performs the corresponding operations. During the period, *vg_lite_task( )* API is suspended, waiting for the instruction to complete. Various instructions are encapsulated in the API.

i.MXRT595 supports OpenVG1.0 API and VGLite Graphics API. Among them, OpenVG provides low-level hardware acceleration interfaces for vector graphics libraries, such as Flash and SVG based on standard protocols. VGLite is a lightweight API whose feature set is smaller than OpenVG, mainly designed for MCU. It has smaller memory footprint and lower CPU overhead. In this document, we call the VGlite API to draw scalar and vector graphics.

For detailed content of VGLite API, refer to *i.MX RT VGLite API Reference Manual* and *VGLite®Graphics API for Vivante GCNanoLiteV Vector Graphics Core*. Among them, the VGLite API has two main rendering function API as mentioned below.

- *Vg_lite_draw*. Render vector graphics primitives. No distortion after zooming in. Graphic elements are objects, and each object is a self-contained entity with attributes, such as color, shape, outline, size, and screen position.
- *Vg_lite_blit*. Render bitmap / raster image. Consists of pixels. These pixels can be sorted and colored in different ways to form a pattern.

3.3 Frame transfer and display

3.3.1 Tearing effect output signal

The G1120B0MIPI module has a tearing effect output line (TE pin) to synchronize MCU to frame writing . The screen refresh frequency refers to the frequency at which the device refreshes the screen, which is constant for a specific device. The refresh process is from left to right (row refresh), and then from top to bottom (column refresh). When completed, the signal can be issued to synchronize internal VSYNC.

The TELOM (Tearing effect line mode) has two types, Mode 1 and Mode 2. Mode 1 refers to the tearing effect output signal consist of V-sync information only. And Mode 2 refers to the signal consist of V-sync and H-sync information. The signal can be enabled or disabled by the set_tear_off (34h) and set_tear_on (35h) commands. The mode of the tearing effect signal is defined by the parameter of the set_tear_on (35h) and set_tear_scanline(44h) commands.

The tearing effect output signal is configured to Mode 1, as shown in Figure 2.

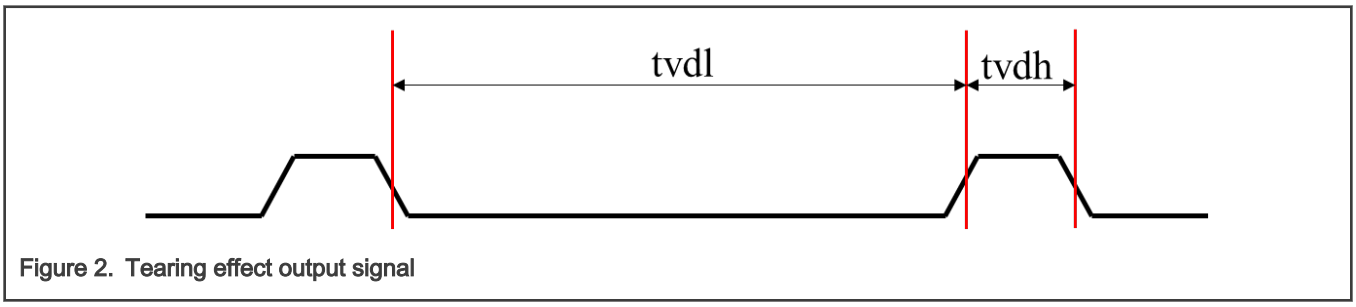

Figure 2. Tearing effect output signal

tvdh = The display is not updated from the frame memory.

tvdl = The display is updated from the frame memory.

The TE pin configuration is briefly summarized in Table 5.

Table 5. TE pin setting instructions

Scene	TE pin settings
1frame send time < 1 frame refresh time	set interrupt at the start of VSYNC
1 frame refresh time< 1frame send time < 2 frame refresh time	set interrupt at the end of VSYNC

In this document, the specific situation is in line with scenario one. So the TE pin can be set to rising edge trigger interrupt as shown in Example 3.

Example 3: TE pin setting

```
static void BOARD_InitMipiPanelTEPin(void)
{
    const gpio_pin_config_t tePinConfig = {
        .pinDirection = kGPIO_DigitalInput,
        .outputLogic  = 0,
    };
    gpio_interrupt_config_t tePinIntConfig = {kGPIO_PinIntEnableEdge, kGPIO_PinIntEnableHighOrRise};
    GPIO_PinInit(GPIO, BOARD_MIPI_TE_PORT, BOARD_MIPI_TE_PIN, &tePinConfig);
    GPIO_SetPinInterruptConfig(GPIO, BOARD_MIPI_TE_PORT, BOARD_MIPI_TE_PIN, &tePinIntConfig);
    GPIO_PinEnableInterrupt(GPIO, BOARD_MIPI_TE_PORT, BOARD_MIPI_TE_PIN, 0);
    NVIC_SetPriority(GPIO_INTA_IRQn, 3);
    NVIC_EnableIRQ(GPIO_INTA_IRQn);
}
```

3.3.2 Functional analysis

During the graphics generation and display, the connection and access means of each part are as shown in Figure 3.

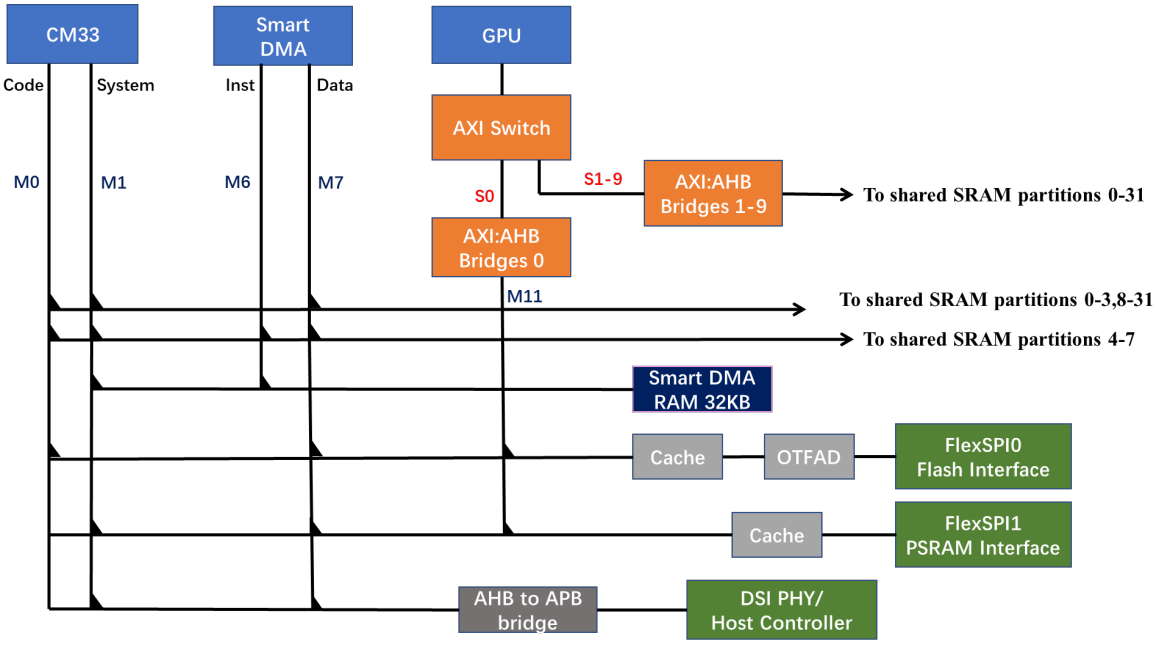

Figure 3. Overall connection and access means

Below points need attention.

1. CM33 and Smart DMA access SRAM via the AHB matrix, and Smart DMA data bus can access all SRAM.
2. Smart DMA code is in Smart DMA RAM 32 KB.
3. GPU access SRAM via AXI:AHB Bridges 1 -9. Need to enable AXI Switch clock.
4. CM33's system bus and Smart DMA's data bus can access DSI PHY/Host Controller register via AHB to APB bridge.

Analyze the main function of each module in this process as follows.

- CPU: Host controller. Assign clock to each peripheral. Send commands to GPU. Monitor TE pin signal. Control and configure display device.
- GPU: Receive CPU commands and execute them for image processing and rendering.
- SMART DMA: Transport graphics data, responsible for frame sending(From frame buffer at SRAM or PSRAM to MIPI DSI FIFO).
- SRAM/PSRAM: Store temporary frame buffer.
- Octal Flash: code execution and persistent data storage.
- Display Module: Refresh the screen and display graphical data.

3.3.3 Frame generation transfer processing

3.3.3.1 Round panel processing

Normally, the display panel is rectangular or square. However, G1120B0MIPI is a round panel with 392x392 display buffer. As shown in Figure 4, the displayable area in the panel is a circle with a diameter of 392. The coordinates of the center of the circle are (392/2,392/2).

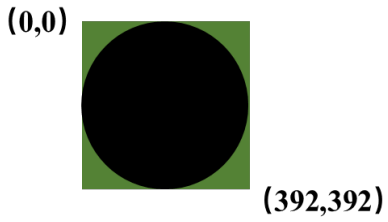

Figure 4. G1120B0MIPI display area

During the drawing, the position of the graphics can be planned according to the coordinates of the circle.

3.3.3.2 One frame buffer used

When there is only one frame buffer, graphical frame generation and transfer process are shown in Figure 5.

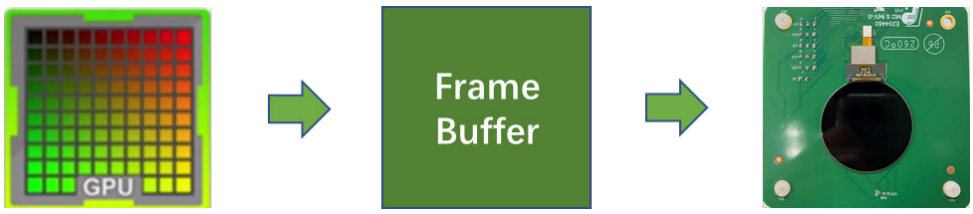

Figure 5. Graphical frame generation and transfer process under one frame buffer

The actual process can be divided into two parts.

Part one: GPU generates graphics to Frame Buffer.

The GPU executes a series of commands which is encapsulated in API sent by CPU to generate graphics.

Create a semaphore *fbdev->semaFramePending*, the semaphore will be given when no frame is pending namely after receiving TE interrupt and triggering Smart DMA transfer.

And, after the GPU generates graphics, take the semaphore to set Frame Buffer. The process is shown in Example 4.

Example 4: GPU generates graphics to Frame Buffer

```
status_t FBDEV_SetFrameBuffer(fbdev_t *fbdev, void *frameBuffer, uint32_t flags)
{
    TickType_t tick;
    const dc_fb_t *dc = fbdev->dc;
    tick = ((flags & (uint32_t)kFBDEV_NoWait) != 0U) ? 0U : portMAX_DELAY;
    if (pdTRUE == xSemaphoreTake(fbdev->semaFramePending, tick))
    {
        return dc->ops->setFrameBuffer(dc, fbdev->layer, frameBuffer);
    }
    else
    {
        return kStatus_Fail;
    }
}
```

Part two: Screen takes graphics from Frame Buffer.

When the TE interrupt comes, check whether the new frame buffer is ready. If done, trigger the Smart DMA transmission to transfer the new buffer to MIPI DSI FIFO. Example 5 shows the process.

Example 5: Data transfer from frame buffer to display module

```
void DC_FB_DSI_CMD_TE_IRQHandler(const dc_fb_t *dc)
{
    dc_fb_dsi_cmd_handle_t *dcHandle;
    dc_fb_dsi_cmd_layer_t *layer;
    dc_fb_info_t *fbInfo;
    void *newFB;
    dcHandle = (dc_fb_dsi_cmd_handle_t *)dc->prvData;
    /* Currently only support one layer, so the layer index is always 0. */
    layer = &(dcHandle->layers[0]);
    newFB = layer->fbWaitTE;
    if (NULL != newFB)
    {
        fbInfo = &(layer->fbInfo);
        layer->fbWaitTE = NULL;
        layer->frameBuffer = newFB;
        (void)MIPI_DSI_WriteMemory(dcHandle->dsiDevice, newFB,
                                   (uint32_t)fbInfo->height * (uint32_t)fbInfo->strideBytes);
    }
}
```

Moreover, the member variables *fbWaitTE* is assigned at function *DC_FB_DSI_CMD_SetFrameBuffer().* When GPU completes generating graphics to Frame Buffer, assign the new Frame Buffer to *fbWaitTE.*

In summary, these two parts complete the synchronization process through a semaphore *fbdev->semaFramePending* and the TE interrupt.

3.3.3.3 Two frame buffer used

If one frame buffer is used, the frame rate may be reduced due to the following reasons.

1. We cannot read and write to a buffer at the same time

2. When the TE interrupts come, the new frame buffer is not yet ready

Therefore, two frame buffers can be used to solve it. Moreover, if two frame buffers are used, allocate them in two different partitions.

The transfer process is shown in Figure 6.

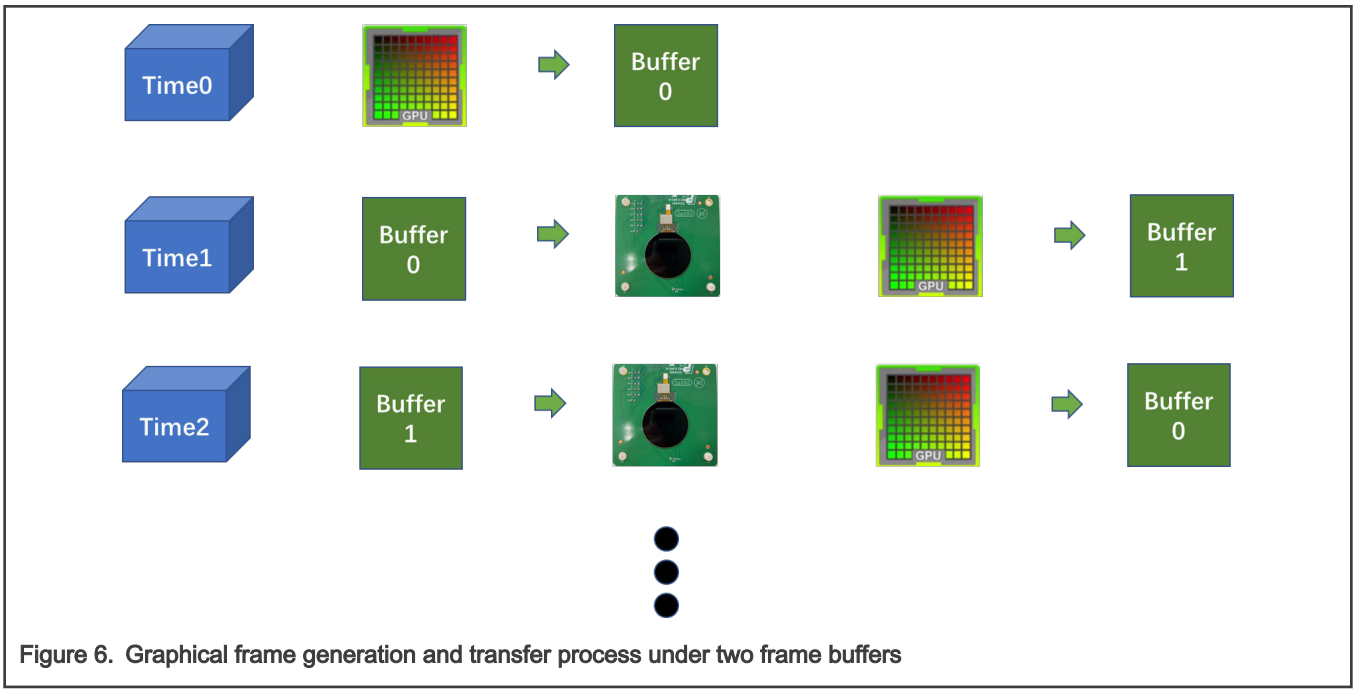

Figure 6. Graphical frame generation and transfer process under two frame buffers

The switching of the two buffers is carried out through the MEMPOOL and the semaphore *fbdev->semaFbManager*.

4 Performance evaluation

Power consumption and frame rate are considerations for performance evaluation during frame rendering. Set different conditions to measure these two points.

4.1 System power domain

There are various power domains in i.MXRT595 and the power domains are supplied by different power rails as illustrated in AN13162 (i.MX RT500 Power Management).

In low-power applications, power management IC is commonly used. On i.MXRT595 EVK board, PCA9420 supplies the power for i.MXRT595 when LDO_Enable is pulled to low (Set JS28.2 and JS28.3).

Among them, the most noteworthy power domains are summarized below in Table 6.

Table 6. Noteworthy power domain

Power Domain	Power supply	Power sink
VDD Core	PMIC_SW1_OUT	CM33, DSP, SRAM, Peripherals IP
VDD1V8	PMIC_SW2_OUT	Analog(PLL, OSC, ADC), OTP, PMC
VDDAO	PMIC_LDO1_OUT	Always-on pad and logic, RTC, POR
VDDIO0	MCU_1V8 (PMIC_SW2_OUT)	FlexSPI0 NOR Flash IOs

Table continues on the next page...

Table 6. Noteworthy power domain (continued)

Power Domain	Power supply	Power sink
VDDIO1	MCU_1V8 (PMIC_SW2_OUT)	FlexSPI1 PSRAM IOs
VDDIO2	MCU_1V8 (PMIC_SW2_OUT)	eMMC Flash IOs
VDDIO3	MCU_3V3 (PMIC_LDO2_OUT)	-
VDDIO4	MCU_1V8 (PMIC_SW2_OUT)	-

4.2 Experimental scene setting

To better analyze the performance, measure the power consumption of each power domain and the corresponding frame rate in below scenes.

- Simple graphics (shown in Figure 7) or complex graphics (shown in Figure 8)
- Signal frame buffer or double frame buffer
- Frame buffer(s) is(are) placed in SRAM or PSRAM

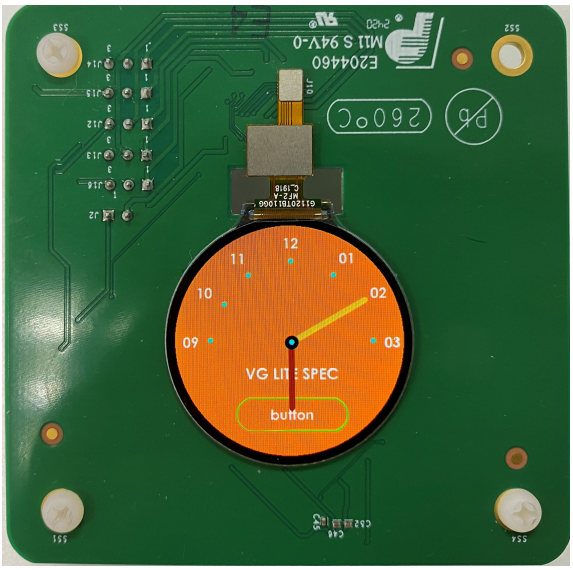

Figure 7. Simple graphics

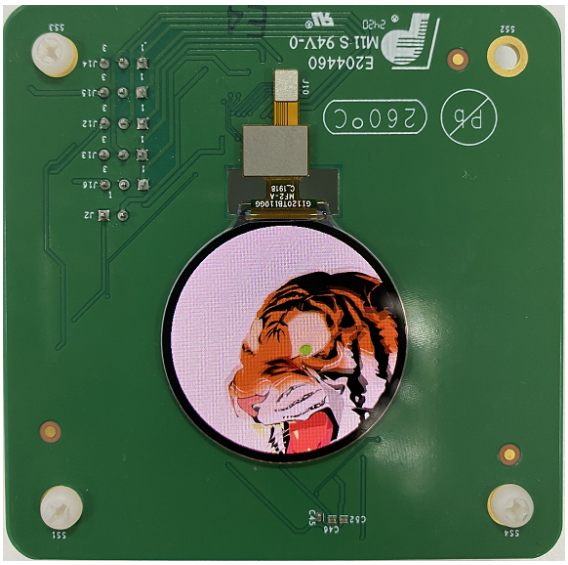

Figure 8. Complex graphics

In order to avoid other influences, all scenes are set according to Table 7.

Table 7. Experimental scene setting

Main Clock	192 M
Flexspi0 Clock	192 M
Flexspi1 Clock (if used)	198 M
GPU Clock	192 M
MIPI DSI Clock	316.8 M
Vddcore	0.9 V
SRAM Rention	5 M RAM Enabled

4.3 Power consumption and frame rate

On i.MXRT595 EVK, jumpers are reserved for power measurement in different domains. Digital multimeter is placed in series instead of jumpers. The power measurement location at jumpers of board is listed in Table 8.

Table 8. Power measurement location

Power Domain	Jumper
VDD Core and MIPI 1V1 (0.9V@active 0.6V@deepsleep)	JS25
VDD1V8	JS30
VDDAO_1V8	JS29

Table continues on the next page...

Table 8. Power measurement location (continued)

Power Domain	Jumper
VDDIO0_1V8	JS20
VDDIO1_1V8	JS21
VDDIO2_1V8	JS22
VDDIO3_3V3	JS23
VDDIO4_1V8	JS24

Figure 9 is the frame rate measurement process.

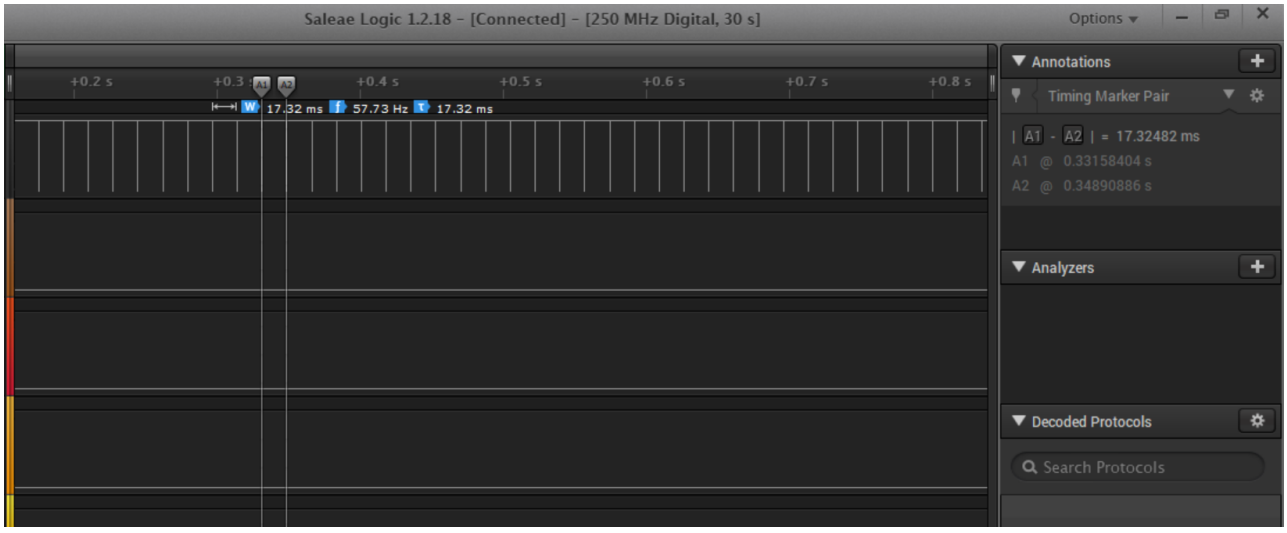

Figure 9. Frame rate measurement process

During the measurement, both graphics rotate 5 degrees counterclockwise every frame.

4.3.1 Scenario one: Frame buffer number as a variable

The first scenario is set as signal or double frame buffer(s) in PSRAM, and displaying simple graphics. The result is shown in Table 9.

Table 9. Current measurement result

Power Domain	Signal frame buffer - Current/mA	Double frame buffers - Current/mA
VDD Core and MIPI 1V1 (0.9V@active 0.6V@deepsleep)	20.25 mA	35.40 mA
VDD1V8	2.43 mA	3.96 mA
VDDAO_1V8	1.0 µA	1.0 µA
VDDIO0_1V8	0.25 mA	0.39 mA

Table continues on the next page...

Table 9. Current measurement result (continued)

Power Domain	Signal frame buffer - Current/mA	Double frame buffers - Current/mA
VDDIO1_1V8	1.57 mA	2.88 mA
VDDIO2_1V8	0.1 μA	0.1 μA
VDDIO3_3V3	0.1 μA	0.1 μA
VDDIO4_1V8	0.1 μA	0.1 μA

Table 10. Frame rate measurement result

/	Signal frame buffer	Double frame buffers
Frame rate	28.68 fps	57.5 fps

4.3.2 Scenario two: Frame buffer location as a variable

The second scenario is set as double frame buffers in SRAM or PSRAM, and displaying simple graphics. The result is shown in Table 11.

Table 11. Current measurement result

Power Domain	SRAM - Current/mA	PSRAM - Current/mA
VDD Core and MIPI 1V1 (0.9V@active 0.6V@deepsleep)	29.25 mA	35.40 mA
VDD1V8	3.86 mA	3.96 mA
VDDAO_1V8	1.0 μA	1.0 μA
VDDIO0_1V8	0.33 mA	0.39 mA
VDDIO1_1V8	0.1 μA	2.88 mA
VDDIO2_1V8	0.1 μA	0.1 μA
VDDIO3_3V3	0.1 μA	0.1 μA
VDDIO4_1V8	0.1 μA	0.1 μA

Table 12. Frame rate measurement result

/	SRAM	PSRAM
Frame rate	57.7 fps	57.5 fps

4.3.3 Scenario three: Graphics complexity as a variable

4.3.3.1 Frame buffers in SRAM

The third scenario is set as double frame buffers in SRAM, and displaying simple or complex graphics. The result is shown in Table 13.

Table 13. Current measurement result

Power Domain	Simple graphics - Current/mA	Complex graphics - Current/mA
VDD Core and MIPI 1V1 (0.9V@active 0.6V@deepsleep)	29.25 mA	33.91 mA
VDD1V8	3.86 mA	4.06 mA
VDDAO_1V8	1.0 µA	1.0 µA
VDDIO0_1V8	0.33 mA	0.34 mA
VDDIO1_1V8	0.1 µA	0.1 µA
VDDIO2_1V8	0.1 µA	0.1 µA
VDDIO3_3V3	0.1 µA	0.1 µA
VDDIO4_1V8	0.1 µA	0.1 µA

Table 14. Frame rate measurement result

/	Simple graphics	Complex graphics
Frame rate	57.7 fps	57.5 fps

4.3.3.2 Frame buffers in PSRAM

The fourth scenario is set as double frame buffers in PSRAM, and displaying simple or complex graphics. The result is shown as Table 15.

Table 15. Current measurement result

Power Domain	Simple graphics - Current/mA	Complex graphics - Current/mA
VDD Core and MIPI 1V1 (0.9V@active 0.6V@deepsleep)	35.40 mA	38.1 mA
VDD1V8	3.96 mA	4.08 mA
VDDAO_1V8	1.0 µA	1.0 µA
VDDIO0_1V8	0.39 mA	0.41 mA
VDDIO1_1V8	2.88 mA	3.5 mA
VDDIO2_1V8	0.1 µA	0.1 µA

Table continues on the next page...

Table 15. Current measurement result (continued)

Power Domain	Simple graphics - Current/mA	Complex graphics - Current/mA
VDDIO3_3V3	0.1 μA	0.1 μA
VDDIO4_1V8	0.1 μA	0.1 μA

Table 16. Frame rate measurement result

/	Simple graphics	Complex graphics
Frame rate	57.5 fps	37 fps

5 Conclusion

This document briefly describes the principle of drawing using GPU and MIPI DSI, and measures the power consumption and frame rate of drawing and pushing screens under different conditions.

1. In some cases, using a single frame buffer as an example, i.MXRT595 has the opportunity to enter a low-power state.
   - When a frame is refreshed and the next TE interrupt has not yet arrived, CPU can enter the deep sleep state and wait for the TE interrupt to wake up.
   - After CPU sends the command to GPU, CPU can enter the WFI state during the execution of GPU.
2. Use on-chip SRAM for frame buffer rather than PSRAM can save power in VDD Core domain and VDDIO1 domain. In addition, the frame rate is limited by the bandwidth of PSRAM.
3. Because the size of the frame buffer is a bit large, when it is placed in PSRAM, the cache is almost useless. Close Cache may get better results.

6 References

1. iMXRT500RM Rev 0.1
2. MIMXRT595-EVK Schematic diagram
3. RM67162 User Guide V0.0
4. RM67162 Data Sheet V0.5

If you are unable to download any of the above listed documents, you will need to request a copy of the document.

7 Revision history

Table 17 is the revision history table.

Table 17. Revision history

Revision number	Date	Substantive changes
0	09 February 2022	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

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.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX,EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

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

Date of release: 09 February 2022
Document identifier: AN13509