AN14836

i.MX RT700 Camera Demo using FlexIO through eDMA

Rev. 1.0 — 10 November 2025 Application note

Document information

Information	Content
Keywords	AN14836, i.MX RT700, FlexIO, eDMA, MIMXRT700-EVK
Abstract	This application note contains all the key information of emulating the parallel camera interface using the FlexIO and eDMA of the i.MX RT700 and create a live video demo using the MIMXRT700-EVK.

1 Introduction

The i.MX RT700 chip contains a rich set of peripherals within multiple functional chip domains. These domains include application core complexes, DSPs, on-chip SRAM, and several high-bandwidth interfaces to access off-chip flash. The chip is designed for low power using advanced clock and power-gating techniques, DVFS, and independent power control of specific chip domains.

This application note provides detailed information on utilizing the FlexIO module to emulate the parallel camera interface using the i.MX RT700 Camera Demo that expands the i.MX RT700 capability to capture live video streams from the camera sensor module with a resolution of 640 x 480. The image is rendered on the connected LCD display through the MIPI-DSI interface without any scaling.

This application note provides details about how the camera sensor module integrates to the MIMXRT700-EVK board by configuring the FlexIO module of the i.MX RT700 to function as a parallel digital camera interface to receive the video frames of data from the sensor module into multiple bytes of data accumulated by the shifter buffers of the FlexIO and then pushed into the SRAM linear buffer in the RGB888 format using the eDMA.

The Camera Demo renders the captured live video frames dumped in the frame buffer via the FlexIO onto the LCD display panel. The panel has a resolution of 720p using the LCDIF, MIPI-DSI host controller, and MIPI-DPHY transmitter interface.

2 Hardware and software required

The demo requires the MIMXRT700-EVK Rev. A1/MIMXRT700-EVK Rev B2 board (no hardware rework required).

3 Omnivision Sensor Module OV7670

Figure 1 shows the Omnivision Sensor Module OV7670.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

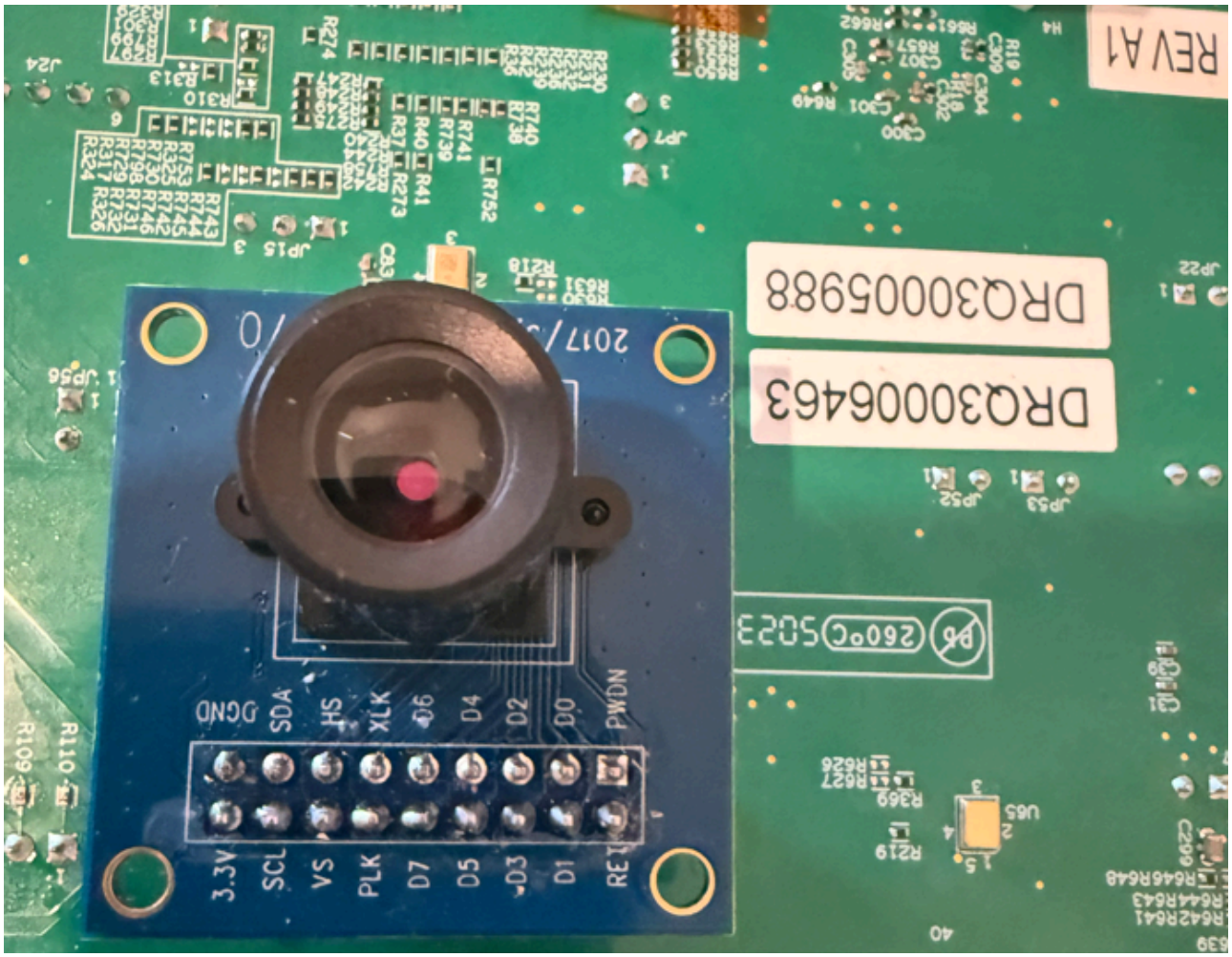

Figure 1. Omnivision Sensor Module OV7670

3.1 Hardware workaround for MIMXRT700-EVK Rev. A1

This module must be mounted to the FlexIO camera socket J53 (CON 2x10) of the MIMXRT700-EVK Rev. A1.

Note: *On the Rev. A1 board, the J53 2x10 connector alignment is directly opposite to the PINs of the OV7670 module. That is, 3.3 V is on PIN 1 of J53, whereas the OV7670 has 3.3 V on PIN 2. The odd pins of J53 are on the even pins of OV7670.*

To properly align the sensor module pins to the J53 camera socket of the MIMXRT700-EVK, the J53 header is removed from the top of the MIMXRT700-EVK and remounted to the bottom of the MIMXRT700-EVK. Refixing the J53 2x10 connector to the back of the MIMXRT700-EVK Rev. A1 version aligns the pins of the OV7670 module with the J53 2x10 CON pins.

Figure 2 shows the J53 2x10 header pinout information as per the MIMXRT700-EVK Rev. A1 schematics.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

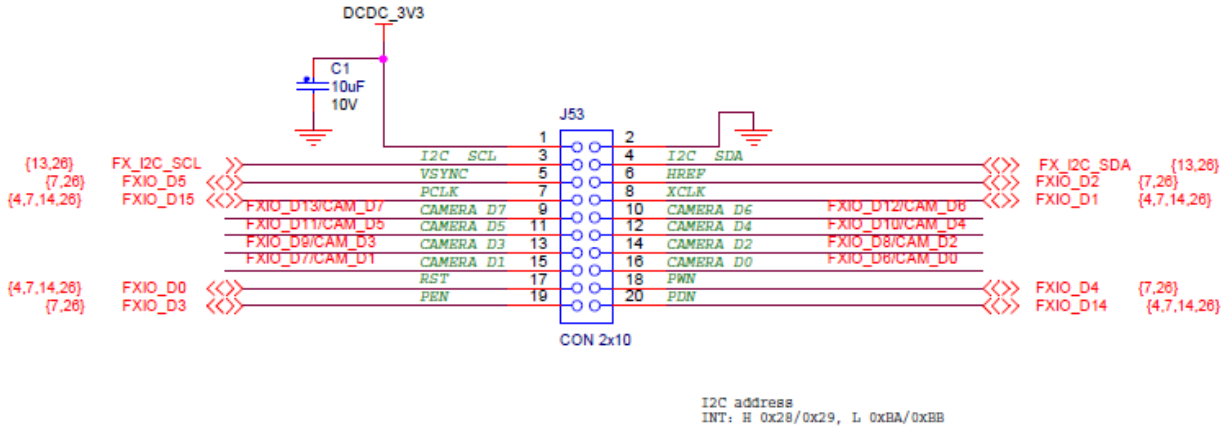

Figure 2. J53 2x10 header pinout

Figure 3 shows the OV7670 module pinout.

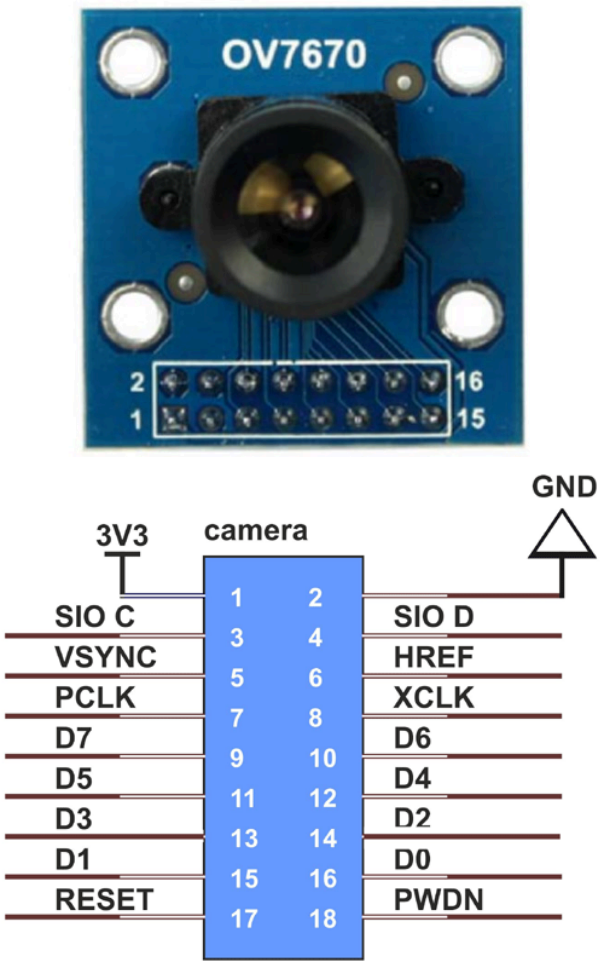

Figure 3. OV7670 module pinout

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Figure 4. J53 header mounted at the bottom of the MIMXRT700-EVK board

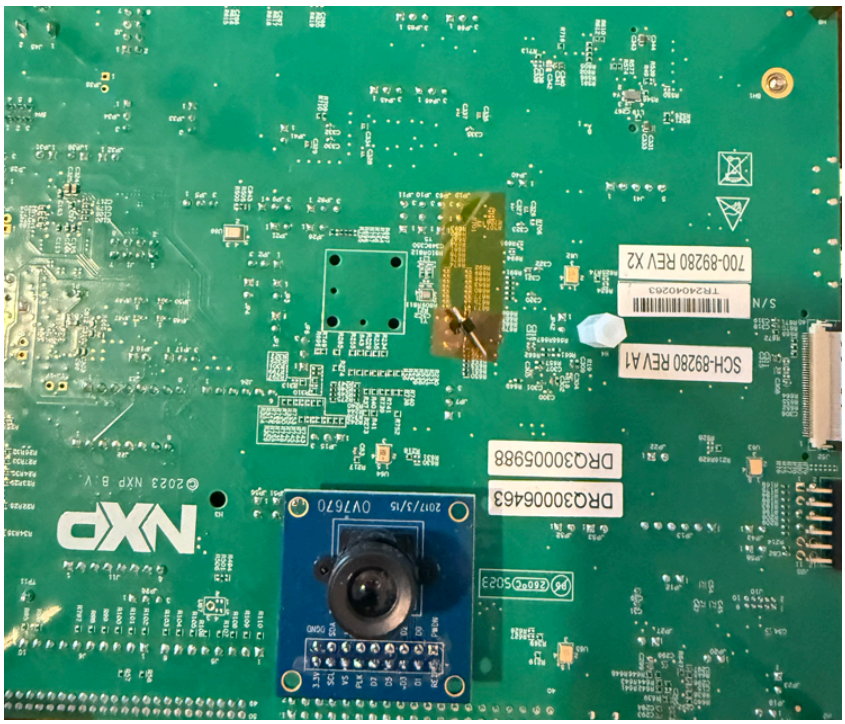

Figure 5. MIMXRT700-EVK bottom side with the OV7670 Camera Sensor Module

The above hardware rework done to the MIMXRT700-EVK Rev. A1 board is not required for mounting the OV7670 camera module to the MIMXRT700-EVK **Rev. B2** board. On the MIMXRT700-EVK Rev. B2, the J53

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

2x10 CON pins are swapped to align with the OV7670 module pins, as shown in Figure 6. The 3.3 V pin is on pin 2 and GND is on pin 1, similar to that of the OV7670 module.

Figure 6 shows the J53 2x10 header pinout as per the MIMXRT700-EVK Rev. B2 schematics.

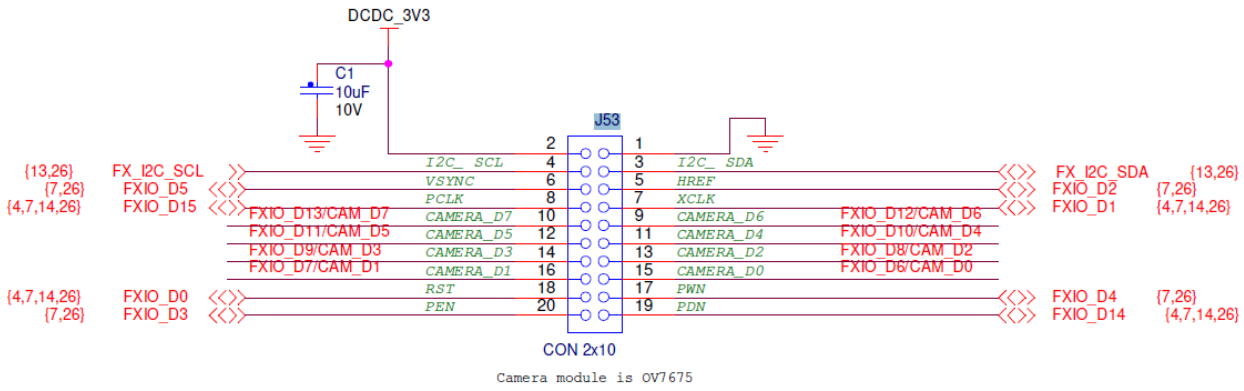

Figure 6. J53 2x10 header pinout as per the MIMXRT700-EVK Rev. B2 schematics

Due to this hardware fix on the MIMXRT700-EVK **Rev. B2**, the hardware rework is not required.

The other hardware and software tools used in the Camera Capture Demo are as follows:

- LCD – 720P LCD – 720x 1280 – RK055HDMIPI4MA0 / RK055HDMIPI4M
- IAR IDE using SDK_2_15_004_MIMXRT700-EVK
- Omnivision OV7670 Sensor Module 640 x 480p @ 30 FPS along with the software driver

4 Bitbucket project source code repositories

The IAR version repository, based on SDK 2_16_000, is located at ssh://git@bitbucket.sw.nxp.com/oemwcse/rt700_camera_demo_using_flexio_iar.git and the URL is https://bitbucket.sw.nxp.com/projects/OEMWCSE/repos/rt700_camera_demo_using_flexio_iar/browse.

The MCUXpresso version repository, based on SDK_25_06_00, is located at:

- Core 0: ssh://git@bitbucket.sw.nxp.com/oemwcse/rt700_camera_capture_using_flexio_power_opt_core0.git, URL: https://bitbucket.sw.nxp.com/projects/OEMWCSE/repos/rt700_camera_capture_using_flexio_power_opt_core0/browse
- Core 1: ssh://git@bitbucket.sw.nxp.com/oemwcse/rt700_camera_capture_using_flexio_power_opt_core1.git, URL: https://bitbucket.sw.nxp.com/projects/OEMWCSE/repos/rt700_camera_capture_using_flexio_power_opt_core1/browse

5 Overview of i.MX RT700 Flexible IO (FlexIO)

The i.MX RT700 MCU has one instance of the FlexIO module.

FlexIO is a highly configurable module that provides:

- Emulation of various serial or parallel communication protocols.
- Flexible 16-bit timers with support for various trigger, reset, enable, and disable conditions.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- Programmable logic blocks that allow the implementation of digital logic functions on the chip and configurable interaction of internal and external modules.
- Programmable state machine for offloading basic system-control functions from the CPU.

FlexIO uses shifters, timers, and external triggers to shift data in or out of FlexIO. As shown in Figure 7, timers control the timing of this data shift. You can configure the timers to use generic timer functions, external triggers, or various other conditions to determine the control.

The FlexIO external trigger inputs come from various sources within the chip and they are selected by the FlexIO peripheral input muxes residing at the INPUTMUX_BASE address plus offsets 0x760h - 0x76Ch. See the FLEXIO_TRIG registers in the INPUTMUX.

Figure 7 provides a high-level overview of the FlexIO timer and shifter configuration.

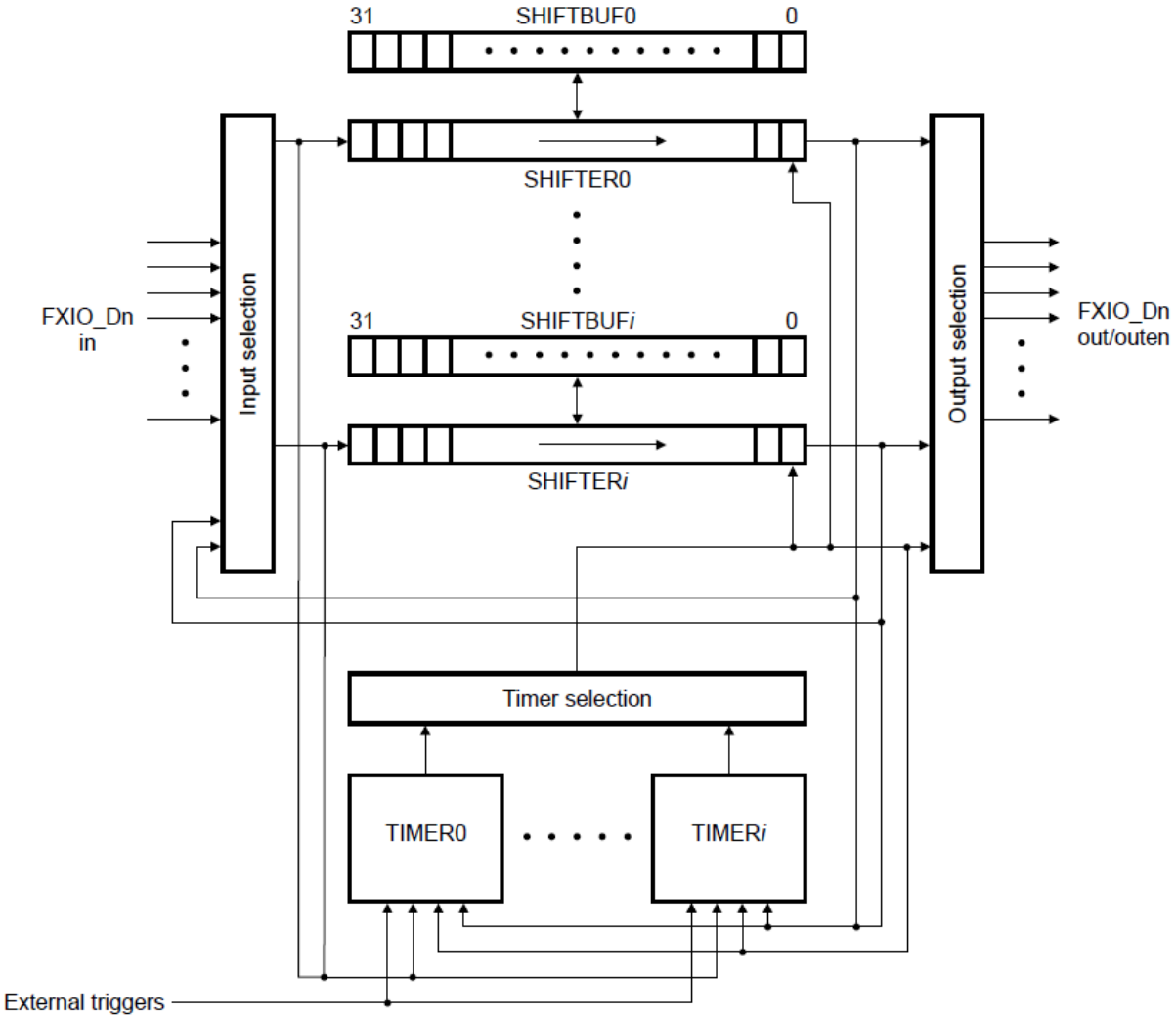

Figure 7. High-level overview of the FLEXIO timer and shifter configuration

5.1 FlexIO features

The FlexIO features are as follows:

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- Array of 32-bit shift registers with transmit, receive, data match, logic, and state modes:
  - Double-buffered shifter operation for continuous data transfer
  - Shifter concatenation to support large transfer sizes
  - Automatic start and stop bit generation
  - 1, 2, 4, 8, 16, or 32 multibit shift widths for parallel interface support
  - Interrupt, DMA, or polled transmit and receive operation
- Highly flexible 16-bit timers with support for various internal or external triggers, reset, enable, and disable conditions:
  - Programmable baud rates independent of bus clock frequency with support for asynchronous operation during the Stop mode
  - Programmable logic mode for integrating external digital-logic functions on the chip or combining pin, shifter, or timer functions to generate complex outputs
  - Programmable state machine for offloading basic system control functions from CPU with support for up to eight states, eight outputs, and three selectable inputs per state
- Integrated general-purpose I/O registers and pin rising or falling edge interrupts to simplify software support
- Support for a wide range of protocols, including but not limited to:
  - UART
  - I^2C
  - SPI
  - I^2S
  - Camera interface
  - Motorola 68K or Intel 8080 bus
  - PWM or waveform generation
  - Input-capture (pulse-edge interval measurement), such as SENT

5.2 Shifters and timers

FlexIO consists of shifters, timers, and pins. The amount of these resources for a given processor can be read from the PARAM register. For example, the FlexIO module on i.MX i.MX RT700 has eight shifters, eight timers, and 32 pins.

The transmit and receive modes are the two basic modes of the shifters. When a shifter is in the transmit mode, it loads data from its buffer register and then shifts the data out to its assigned pin/pins. When a shifter is in the receive mode, it shifts the data from its assigned pin/pins and then stores the data into its buffer register. The loading, storing, and shifting operations are controlled by the shifter’s assigned timer.

The timers can be also configured in different operating modes, including the dual 8-bit counters baud/bit mode, dual 8-bit counters PWM mode, and single 16-bit counter mode.

The dual 8-bit counters' baud/bit mode is used to build a data transmitter. In this mode, the lower eight bits of the 16-bit timer divide the module clock source to generate the desired baud rate and the higher eight bits count the shift bits of a frame. After it is enabled, the timer loads the initial value from its compare register and starts to count down. When the lower eight bits decrement to zero, the timer’s shift clock and its output signal are toggled to generate rising or falling edges. The higher eight bits count down by one. The shift clock drives the shifter. The timer output signal usually drives a pin for the clock output, such as the SCK of a SPI master and the WR of the 8080 bus. After that, the lower eight bits reload the initial value to start another decrement cycle. The two decrement cycles make up a shift cycle, which drives the shifter to shift by one beat. When all 16 bits decrement to zero, all data bits in the shifts are shifted out. The timer is then disabled before another transfer frame.

The dual 8-bit counter PWM mode generates the PWM outputs. The lower eight bits configure the high period of the timer shift clock and the upper eight bits configure the low period of the shift clock.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

The single 16-bit counter mode creates a synchronous communication slave, such as the SPI slave, I^2S slave, and so on. All 16 bits configure the timer shift clock.

5.3 Serial and parallel transfers

The FlexIO on the i.MX RT devices supports both serial and parallel transfers. The data is always shifted from the MSB to the LSB in a shifter for both types of transfers. In the serial transmitter mode, the data is shifted out bit by bit from the LSB (bit zero). In the serial receiver mode, the data is shifted in bit by bit from the MSB (bit 31). The process is shown in Figure 8a.

In the parallel transmitter mode, the data is shifted out from the *n* LSBs of a shifter. In the parallel receiver mode, the data is shifted in from the *n* MSBs, where *n* is the parallel bus width. Figure 8b shows the use case of *n* = 8.

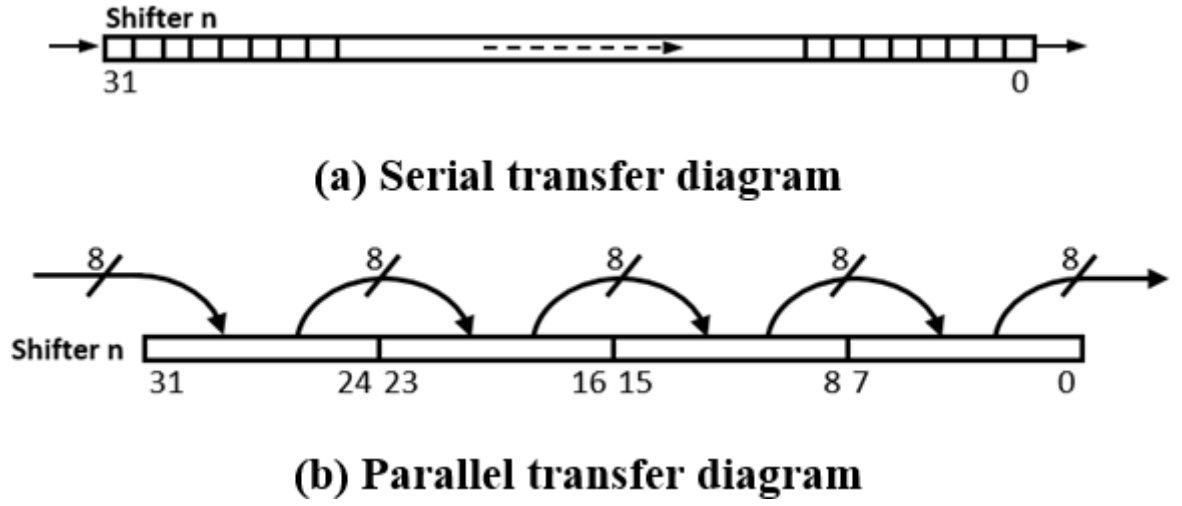

Figure 8. Serial and parallel transfers

The parallel transfer mode works as follows:

- The data is shifted by *n* bits on each shift clock, where *n* is the configured bus width.
- 4, 8, 16, or 32-bit bus widths are supported.
- It combines multiple shifters together for a concatenation to support large transfer sizes and use the DMA method to access the shifter buffer registers for high-speed transfers and low-power operations. Figure 9 shows the shifter concatenation diagram, where the additional shifters work as FIFOs.

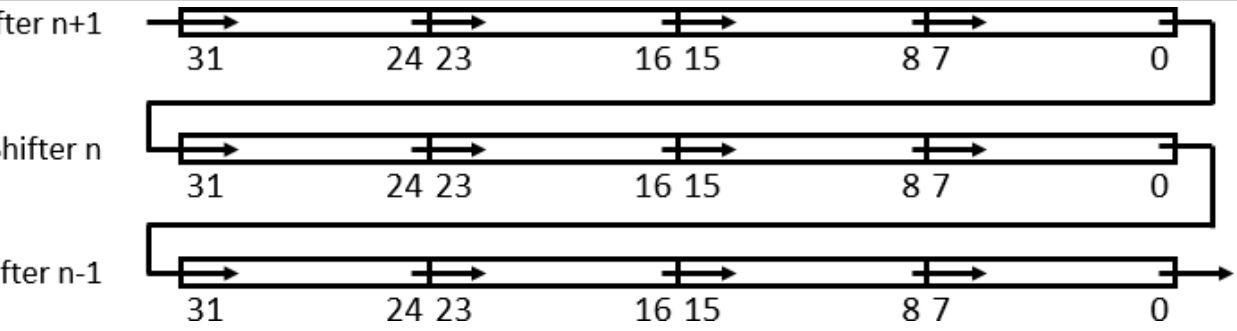

Figure 9. Shifter concatenation diagram

- Only specific shifters (SHIFTER0 and SHIFTER4) support the parallel output to pins. However, all shifters (except for SHIFTER0) support the output to the adjacent low-order shifters.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- Only specific shifters (SHIFTER3 and SHIFTER7) support the parallel input from pins. However, all shifters (except for SHIFTER7) support inputting from the adjacent high-order shifters.
- Any FlexIO pin can be a parallel data output/input pin. However, the pin indexes must be successive for a specific usage, such as pin 0 to pin 7, pin 1 to pin 8, and so on for the 8-bit bus.

5.4 General configurations and operations

The FlexIO can emulate various communication protocols. However, to emulate a dedicated peripheral and to handle the transmit and receive process, the FlexIO must be configured by software.

To implement the master transmitter, the shifter is configured in the transmit mode and the assigned timer is configured in the dual 8-bit counter baud/bit mode. The timer decrement clock originates from the module clock. The timer trigger originates from the shifter flag with a reversed polarity. Filling the shifter buffer via a polling/interrupt/DMA clears the shifter flag, which enables the timer to start counting down. The decrement of the timer drives the shifter to shift the data out and generates the clock output signal.

To implement the receiver, the shifter is configured in the receive mode. The timer is configured in the dual 8-bit counter baud/bit mode for the synchronous master receiver, such as the 8080 bus reading implementation. This timer mode is also used for asynchronous receivers, such as the UART receiver. The receive process is similar to that of the master transmitter, but the data is shifted into the shifter rather than shifted out. For the synchronous receiver, the assigned timer is configured in the single 16-bit counter mode, such as the SPI slave receiver and the parallel camera interface. The decrement clock originates from the pin input, such as the SPI SCK and the camera PCLK signal. The timer trigger originates from another pin, such as the SPI CS and the camera HREF signal. The master device enables the timer and controls the decrement via pins. The decrement of the timer drives the shifter to shift in the data.

For the ease of use, NXP provides API drivers and driver examples in the MCUXpresso SDK. You can also find detailed descriptions and similar APIs of these implementations in various application notes.

6 Parallel camera interface

A parallel camera sensor usually has 8/10/16/24 data lines to output the pixel data and uses an 8-bit interface camera. For such camera, if a pixel has eight bits, one transfer cycle is required. If a pixel has more than eight bits, additional transfer cycles are required. For example, an RGB565 (16-bit) format pixel takes two transfer cycles, and an RGB888 (24-bit) format pixel takes three transfer cycles.

Besides the data lines, there are also timing control outputs: VSYNC/VREF, HREF/HSYNC, and PCLK. Figure 10 shows the timing diagram of the OV7670 camera sensor used in this application example.

Figure 10 shows a frame, which contains 510 rows, including 480 valid data lines during VSYNC high. A data line contains 656 points, including 640 valid pixels during HREF high. Therefore, the resolution is VGA (640 x 480).

Two PCLK cycles are used to output the 16-bit RGB565/YUV422 format pixel data with the higher byte first and the lower byte after it.

The data is changed on the PCLK falling edge and latched on the PCLK rising edge.

The frame resolution is adjustable. Different resolutions have similar signal-timing sequences. Figure 10 illustrates the timing sequence's difference between the VGA, QVGA (320 x 240), and QQVGA (160 x 120) resolutions.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

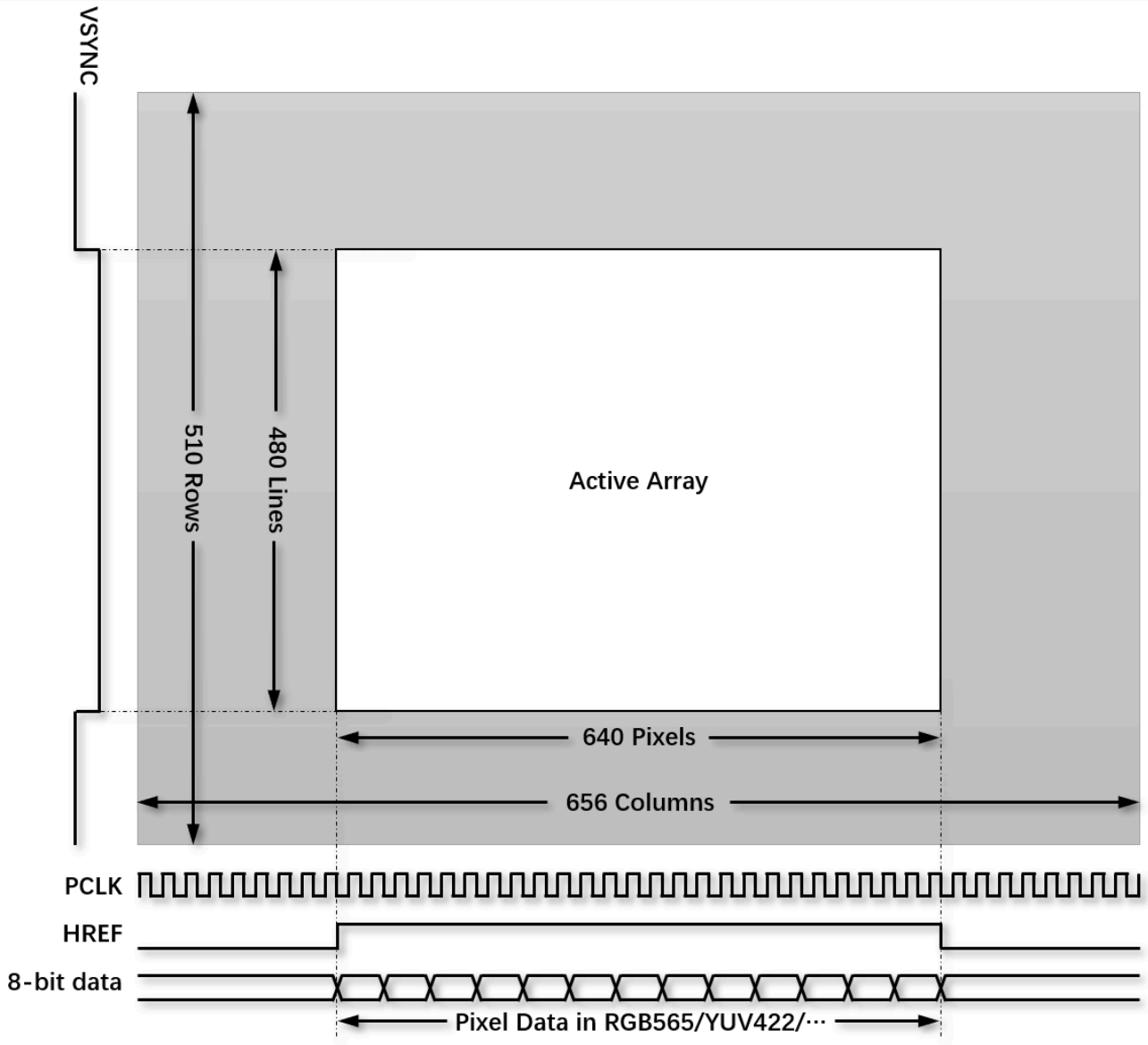

Figure 10. Timing sequences

Figure 11 shows that the lower the frame resolution is, the longer the HREF blanking area is. The sizes of active areas from different resolutions are the same. Other signals' timings are similar for different resolutions.

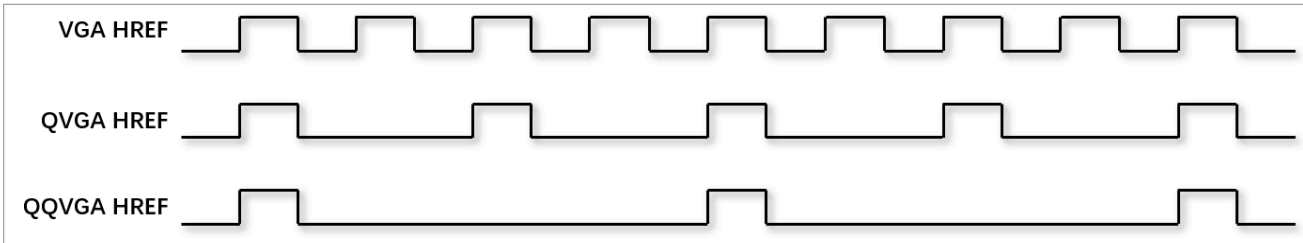

Figure 11. HREF blank area

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7 Using FlexIO to emulate parallel camera interface

This section describes how to use the FlexIO to emulate a parallel camera interface with i.MX RT700.

7.1 Operation principles of FlexIO camera interface

You can configure the FlexIO to emulate a parallel camera interface in different ways, such as with a different data bus width, a number of concatenated shifters, and specific shifters, pins, and timers.

The multibeat transfer supports large transfer sizes. Here, a beat means a shift operation. One transfer sequence requires the timer to generate multiple shift clocks. The number of beats per one transfer sequence is related to the number of concatenated shifters and the bus width. One shifter has 32 bits. One shifter supports one 4-beat transfer for an 8-bit bus. Two shifters support eight beats and so on. This application uses all eight shifters. Therefore, 32 beats are supported for an 8-bit bus.

Figure 12 shows the FlexIO resources' organization for a parallel camera interface.

In the organization, all eight shifters are concatenated together. TIMER0 controls the shifters' shifting. TIMER1 generates the XCLK (24-MHz in this application). D0-D7, HREF, and XCLK are based on the FlexIO pins. An additional GPIO pin receives the VSYNC signal. The SHIFTER0 status flag is used to trigger the DMA request.

The process of the transfer is as follows:

1. Configure the FlexIO, DMA, GPIO, and so on. Detailed configurations are shown in the following sections.
2. In a VSYNC rising-edge ISR, retarget the DMA destination address to a new frame buffer.
3. A new line starts to transfer. TIMER0 is enabled by an HREF rising edge.
4. TIMER0 starts to count down along with the PCLK input. At the same time, TIMER0 generates the shift clock controlling the shifters to shift the data in. The shift clock is generated per one PLCK clock. 8-bit data are shifted per one shift clock.
5. TIMER0 counts down to zero and a compare event occurs after 32 shift clocks.
6. A storing event is signaled by the compare event. Data are stored from SHIFTER0-SHIFTER7 to shifter buffers SHIFTBUF0-SHIFTBUF7.
7. The storing event fills up the shifter buffers, which sets the shifters' status flags and triggers a DMA request.
8. The eDMA copies the data from shifter buffers SHIFTBUF0-SHIFTBUF7 to the SRAM. 32 bytes are copied per one request.
9. After the compare event, TIMER0 loads the initial value from TIMCMP0 again. Then, steps 4-9 repeat.
10. After the line is transferred, HREF goes to low. TIMER0 is disabled until a new line starts to transfer. Then, steps 3-10 repeat.
11. After the frame is transferred, VSYNC goes to low. Then, steps 2-11 repeat.

This is the process of capturing one frame. The CPU is only required to retarget the DMA destination address in the VSYNC ISR. All other operations for capturing the frame are done by FlexIO and eDMA. The display DMA source address configuration is also set in the VSYNC ISR.

7.2 Demo application hardware platform

Figure 12 shows how the camera sensor module pins are connected to the FlexIO module of the i.MX RT700 using the GPIOs configured as input pins that emulate a parallel camera interface. It also shows the captured video frame data dumped into the frame buffer located in the SRAM from the camera module using the FlexIO through the eDMA. It also provides the data flow from the frame buffer to the LCDIF at the output via the LCDIF and MIPI-DSI of the i.MX RT700 to the RK055HDMIPI4MA0 external rectangular LCD display module with the resolution of 720p x 1280.

For this demo, the MIMXRT700-EVK is modified by removing the J53 2x10 connector that is mounted on top of the EVK and placing it at the back of the EVK, in the same board socket where it was initially mounted on the

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

top. This enables the Omnivision OV7670 sensor module pins to perfectly align with the FLEXIO GPIO pins that are configured to emulate the parallel camera interface, without the need for external wires between the sensor module and the EVK board.

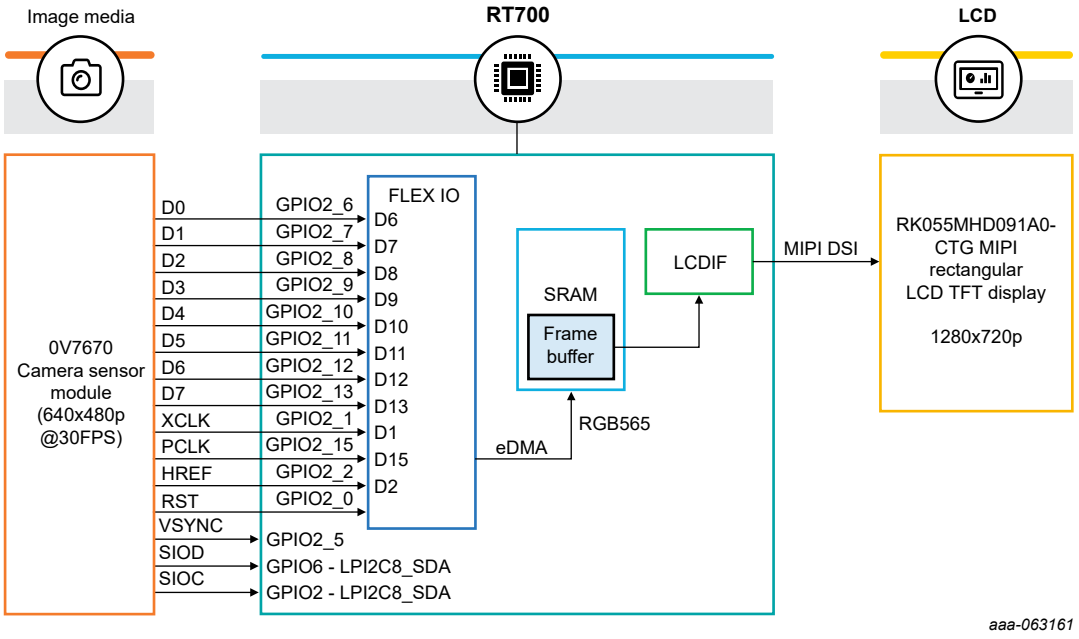

Figure 12. i.MX RT700 camera frame data flow diagram

Table 1 describes the pin mux configurations for the camera signals.

Table 1. i.MX RT700 FLEXIO GPIO pin connections with the camera sensor module

i.MX RT700 pin	Pin mux function	Camera signals
GPIO2_06	FLEXIO_D6 (EVK - J53 CON 2x10 -16)	OV7670_D0
GPIO2_07	FLEXIO_D7 (EVK - J53 CON 2x10 -15)	OV7670_D1
GPIO2_08	FLEXIO_D8 (EVK - J53 CON 2x10 -14)	OV7670_D2
GPIO2_09	FLEXIO_D9 (EVK - J53 CON 2x10 -13)	OV7670_D3
GPIO2_10	FLEXIO_D10 (EVK - J53 CON 2x10 -12)	OV7670_D4
GPIO2_11	FLEXIO_D11 (EVK - J53 CON 2x10 -11)	OV7670_D5
GPIO2_12	FLEXIO_D12 (EVK - J53 CON 2x10 -10)	OV7670_D6
GPIO2_13	FLEXIO_D13 (EVK - J53 CON 2x10 - 9)	OV7670_D7
GPIO2_01	FLEXIO_D1 (EVK - J53 CON 2x10 -8)	OV7670_XCLK
GPIO2_15	FLEXIO_D15 (EVK - J53 CON 2x10 -7)	OV7670_PCLK
GPIO2_02	FLEXIO_D2 (EVK - J53 CON 2x10 -6)	OV7670_HREF
GPIO2_05	FLEXIO_D5 (EVK - J53 CON 2x10 -5)	OV7670_VSYNC
GPIO0_06	LPI2C8_SDA (EVK - J53 CON 2x10 -4)	OV7670_SIOD
GPIO0_07	LPI2C8_SCL (EVK - J53 CON 2x10 -3)	OV7670_SIOC
GPIO2_00	FLEXIO_D0 (EVK - J53 CON 2x10 -17)	OV7670_RST
GPIO2_04	FLEXIO_D4 (EVK - J53 CON 2x10 -18)	OV7670_PWN

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Note the following about the camera pin mux:

- The camera output data pins D0~D7 must be connected to eight successive FlexIO pins. This camera demo uses the FLEXIO_D06~FLEXIO_D13 FlexIO pins.
- XCLK, PCLK, and HREF are connected to any of the three FlexIO pins, in this case to FLEXIO_D1, FLEXIO_D15, and FLEXIO_D2, respectively.
- VSYNC connects to the GPIO2_5 pin, which triggers the interrupt upon a new frame (Vblank interrupt) used to switch buffers.
- SIOD and SIOC connect to the GPIO0_06(SDA) and GPIO0_07(SCL) pins of the i.MX RT700, configured to function as the Flexcomm LPI2C master, which configures the OV7670 module to output video frames in the VGA format (640 x 480p @ 30 FPS).

7.3 Application software pipeline

Figure 13 shows the software pipeline flow. The respective modules enable the camera frame data flow from one end of the pipeline to the other end via the modules of the i.MX RT700, which renders them onto an LCD display capturing live video frames at 640 x 480p @ 30 FPS.

The camera demo application uses the IAR SDK_2_15_004_MIMXRT700-EVK of the i.MX RT700. Since the i.MX RT700 has a lot of SRAM memory, the buffer size requirement of the incoming rectangular MIPI-DSI LCD panel is 1280 x 720, and the pixel format of the incoming video frame is RGB565 (two bytes per pixel), the application allocates a double buffer of 1280 x 720 x 2 bytes.

The i.MX RT700 FlexIO module is configured to emulate a parallel camera interface using the GPIOs, shifters, and timers of the FlexIO module to receive the data from the camera module. The FlexIO XCLK (GPIO2_01) pin must be configured to drive the camera sensor module with a 24-MHz clock using Timer0 and the FlexIO driver API provided in the SDK.

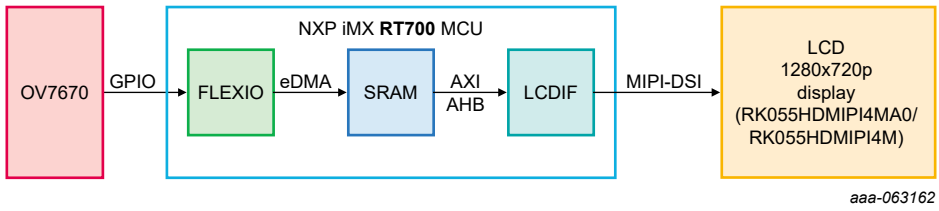

Figure 13. Application software pipeline

The OV7670 camera sensor module is programmed by the main application using the LPI2C8 Flexcomm configured to work as an I^2C master device to enable the application program the OV7670 module to output 640 x 480p @ 30 FPS in the RGB565 pixel format data.

The VSYNC interrupt from the OV7670 module is mapped to GPIO2_5 of the RT700 and configured to generate the corresponding interrupt that triggers the ISR, which manages the switching of the buffer to ensure seamless and flicker-free rendering of the captured video frames.

The FlexIO module emulates the parallel camera interface by programming the shifters and timers according to the operation principles of the FlexIO camera interface, as described in Section 5.1. The eDMA is configured using the latest SDK driver API to capture the incoming video streams and dump them into the SRAM frame buffer memory, which is rendered using the display subsystem of the i.MX RT700.

The display subsystem of the i.MX RT700, including LCDIF and MIPI-DSI, is configured and programmed using the display examples provided as part of the i.MX RT700 SDK.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.4 FlexIO configuration for parallel camera interface

The FlexIO configuration for the camera interface consists of the shifter and timer configurations. The following code, which is clipped from the fsl_flexio_camera.c\FLEXIO_CAMERA_DataPath_Init() API driver, implements the shifter configuration.

The following code shows the shifter configurations:

- Timer selection for shifting clock generation: TIMER0 is selected for this application.
- Shifting polarity: Shift is on the positive edge of the shift clock.
- Shifter pin I/O configuration: The output is disabled, which means for input only.
- The first data pin selection: Pin 06 is selected in this application. The data pins are 06~13 when the parallel width is 8.
- Shifter pin polarity: The pin is active high, which means that the logic is not reversed from the pin to the shifter.
- Shifter mode: Receive mode.
- Parallel width: 8-bit in this application.
- Input source for the shifter: In this application, the input source of SHIFTER0~SHIFTER6 is the output of SHIFTER1~SHIFTER7 (respectively) and the input source of SHIFTER7 is from the data pins. This is the shifter concatenation feature.
- Shifter stop bit: Disabled.
- Shifter start bit: Disabled.

```
/* FLEXIO_CAMERA shifter config */
    shifterConfig.timerSelect = base->timerIdx;
    shifterConfig.timerPolarity = kFLEXIO_ShifterTimerPolarityOnPositive;
    shifterConfig.pinConfig = kFLEXIO_PinConfigOutputDisabled;
    shifterConfig.pinSelect = base->datPinStartIdx;
    shifterConfig.pinPolarity = kFLEXIO_PinActiveHigh;
    shifterConfig.shifterMode = kFLEXIO_ShifterModeReceive;
    shifterConfig.parallelWidth = FLEXIO_CAMERA_PARALLEL_DATA_WIDTH - 1U;
    shifterConfig.inputSource = kFLEXIO_ShifterInputFromNextShifterOutput;
    shifterConfig.shifterStop = kFLEXIO_ShifterStopBitDisable;
    shifterConfig.shifterStart =
 kFLEXIO_ShifterStartBitDisabledLoadDataOnEnable;

    /* Configure the shifters as FIFO buffer. */
    for (i = base->shifterStartIdx; i < ((base->shifterStartIdx + base-
>shifterCount) - 1U); i++)
    {
        FLEXIO_SetShifterConfig(base->flexioBase, i, &shifterConfig);
    }
    shifterConfig.inputSource = kFLEXIO_ShifterInputFromPin;
    FLEXIO_SetShifterConfig(base->flexioBase, i, &shifterConfig);
```

The following code, which is clipped from the fsl_flexio_camera.c\FLEXIO_CAMERA_DataPath_Init() API driver, implements the shifting timer configuration.

```
/* FLEXIO_CAMERA timer config, the PCLK's clk is source of timer to drive the
 shifter, the HREF is the selecting
     * signal for available data. */
    timerConfig.triggerSelect = FLEXIO_TIMER_TRIGGER_SEL_PININPUT(base-
>hrefPinIdx);
    timerConfig.triggerPolarity = kFLEXIO_TimerTriggerPolarityActiveHigh;
    timerConfig.triggerSource = kFLEXIO_TimerTriggerSourceInternal;
    timerConfig.pinConfig = kFLEXIO_PinConfigOutputDisabled;
    timerConfig.pinSelect = base->pclkPinIdx;
    timerConfig.pinPolarity = kFLEXIO_PinActiveHigh;
```


All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

```
    timerConfig.timerMode = kFLEXIO_TimerModeSingle16Bit;
    timerConfig.timerOutput = kFLEXIO_TimerOutputZeroNotAffectedByReset;
    timerConfig.timerDecrement = kFLEXIO_TimerDecSrcOnPinInputShiftPinInput;
    timerConfig.timerReset = kFLEXIO_TimerResetOnTimerTriggerRisingEdge;
    timerConfig.timerDisable = kFLEXIO_TimerDisableOnTriggerFallingEdge;
    timerConfig.timerEnable = kFLEXIO_TimerEnableOnTriggerRisingEdge;
    timerConfig.timerStop = kFLEXIO_TimerStopBitDisabled;
    timerConfig.timerStart = kFLEXIO_TimerStartBitDisabled;
    timerConfig.timerCompare = (8U * base->shifterCount) - 1U;

    FLEXIO_SetTimerConfig(base->flexioBase, base->timerIdx, &timerConfig);
```

The code above shows the timer configurations:

- Timer trigger selection: HREF signal pin.
- Trigger polarity: Trigger is active high.
- Trigger source: Internal trigger, which means trigger from the FlexIO module itself.
- Timer pin I/O configuration: Output is disabled, which means for input only.
- Timer pin selection: PCLK signal pin.
- Timer pin polarity: The pin is active high, which means that the logic is not reversed from the pin to the timer.
- Timer mode: 16-bit counter mode.
- Timer output initial state: The timer outputs logic zero when enabled and not affected by the timer reset.
- Timer decrement condition: Counting down on the pin input, the shift clock equals to the pin input.
- Timer reset condition: On the trigger rising edge.
- Timer disablement condition: On the trigger falling edge.
- Timer enablement condition: On the trigger rising edge.
- Timer Stop bit: Disabled.
- Timer Start bit: Disabled.
- Timer compare (initial value): 63. The number of shifting is (63 + 1) / 2 = 32, which equals the number of bytes in each DMA transfer.

7.5 FlexIO timer configuration for camera XCLK

A camera sensor needs a clock source to work. The clock input is usually called PCLK or MCLK. It can be derived from an oscillator or a clock output from the processor.

The OV7670 camera sensor used in this application requires the clock with the range of 10 ~ 48 MHz and the typical value of 24 MHz. In the application, another FlexIO clock (TIMER1) is used to generate a 24-MHz clock for this requirement. The following code, which is clipped from fsl_flexio_camera.c\FLEXIO_CAMERA_XCLK_Config(), implements the function to configure TIMER1.

```
  /* Configure the timer DEMO_FLEXIO_TIMER_CH for generating PWM */
    timerConfig.triggerSelect   = 0U;           /* any value, ok */
    timerConfig.triggerSource   = kFLEXIO_TimerTriggerSourceInternal;
    timerConfig.triggerPolarity = kFLEXIO_TimerTriggerPolarityActiveHigh;
    timerConfig.pinConfig       = kFLEXIO_PinConfigOutput;
    timerConfig.pinPolarity     = kFLEXIO_PinActiveHigh;
    timerConfig.pinSelect       = BOARD_CAMERA_FLEXIO_XCLK_PIN_INDEX; /* Set pwm
output */
    timerConfig.timerMode       = kFLEXIO_TimerModeDual8BitPWM;
    timerConfig.timerOutput     = kFLEXIO_TimerOutputOneNotAffectedByReset;
    timerConfig.timerDecrement  =
kFLEXIO_TimerDecSrcOnFlexIOClockShiftTimerOutput;
    timerConfig.timerDisable    = kFLEXIO_TimerDisableNever;
    timerConfig.timerEnable     = kFLEXIO_TimerEnabledAlways;
```


All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

```
    timerConfig.timerReset      = kFLEXIO_TimerResetNever;
    timerConfig.timerStart      = kFLEXIO_TimerStartBitDisabled;
    timerConfig.timerStop       = kFLEXIO_TimerStopBitDisabled;

    timerConfig.timerCompare = 0x0201; /* With FLEXIO 120MHz clock source, PWM
 generates 24MHz clock.*/

    FLEXIO_SetTimerConfig(BOARD_CAMERA_FLEXIO_INST,
 BOARD_CAMERA_FLEXIO_PWM_TIMER, &timerConfig);
```

7.6 eDMA software configuration

When the frame data is shifted in and the shifters are full, the data must be read out to the memory in time to avoid overflow. The eDMA is used for the data transfer to lower the CPU load. The following code, which is clipped from flexio_camera_demo_main.c\Start_EDMACameraDataCapture(uint8_t *pCurrentCamBuf), implements the DMA TCD (Transfer Control Descriptor) configuration.

```
/* Configure eDMA */
cameraFlexIOAddr  = FLEXIO_CAMERA_GetRxBufferAddress(&gFlexioCameraDevice);
    cameraDestBufAddr = (uint32_t)pCurrentCamBuf;

    EDMA_PrepareTransfer(&transferConfig, (void *)cameraFlexIOAddr,
 sizeof(cameraFlexIOAddr), (void *)cameraDestBufAddr, sizeof(cameraDestBufAddr),
 DMA_TRANSER_SIZE, OV7670_FRAME_BYTES, kEDMA_MemoryToMemory);

    transferConfig.enableSrcMinorLoopOffset = 1;
    transferConfig.minorLoopOffset = -32;

    EDMA_SubmitTransfer(&g_DMA_Handle, &transferConfig);
    EDMA_StartTransfer(&g_DMA_Handle);
```

The code above shows the DMA TCD configurations using the eDMA API provided by the SDK:

- Source address: FlexIO SHIFTBUF0 address.
- Source address offset: 0, which means no offset.
- Source address modulo: 0, which means that the source address modulo is disabled.
- Source data transfer size for each time access: 5, which means 32 bytes burst.
- Destination address modulo: 0, which means that the destination address modulo is disabled.
- Destination data transfer size for each time access: 5, which means 32 bytes burst.
- Minor loop byte count: 32 bytes.
- Source minor loop offset mapping: Disabled.
- Destination minor loop offset mapping: Disabled.
- Last source address adjustment: 0, which means no adjustment.
- Destination address: RAM buffer address.
- Destination address offset: 0, which means no offset.
- Major loop counter: (OV7670_FRAME_BYTES/32u) = (640 x 480 x 2) / 32 = 19200.
- Last destination address adjustment: 0, which means no adjustment. The new destination address is configured in the VSYNC ISR.
- Disable the DMA request after each major loop. The request is enabled in the VSYNC ISR each time.

8 Power measurement data

One of the key factors to adopt this use case is its power efficiency.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

To get the best power data, the camera capture demo is modified by bringing in the power optimizations of the display and graphics use cases and propagating those changes to this demo. It is done in phases. Table 2 shows the power data based on MCUXpresso SDK 25_06_00 without any optimization and the code is same as the IAR version of SDK_2_16_000, except for the drivers.

Table 2. RT798 (camera demo) @ 30 FPS SDK_25_06_00 tested using MCUXpresso on PSRAM

-	V	mA	mW
VDD1V8 (JP19)	1.80	9.36	16.85
VDD1V8 MIPI (R311)	1.80	0.70	1.26
VDD1 (JP3) @ 0.63 V	0.99	5.67	5.61
VDD2 (JP1) @ 0.90 V	1.09	137.18	149.53
VDDN (JP2) @ 1.1 V	1.10	31.50	34.65
-	-	-	-
Total	-	-	207.90

The first phase of the optimization creates a core 0 and core 1 based project to switch the power source to the PMIC mode only and then boot core 1 from the main of the core 0 using the APP_BootCore1() function. It then switches to core 1 execution, where it is put to the deep-sleep mode to save power. All the unused blocks within the i.MX RT700 MCU have the clocks disabled, as done in the BOARD_InitPowerConfig() function. The critical modules then power down, as done in the BOARD_PowerConfigAfterCPU1Booted () function, and the power mode is switched to PMIC. These three modifications provide a significant power reduction (close to 46 %) and they still use the PLL and FROs as the clock sources. Table 3 shows the power data after the first phase of optimization.

Table 3. RT798 (camera demo) @ 30 FPS SDK_25_06_00 tested using MCUXpresso on PSRAM based on power optimized changes that drives the display and camera using the PLL and FROs

-	V	mA	mW
VDD1V8 (JP19)	1.80	9.28	16.70
VDD1V8 MIPI (R311)	1.80	0.70	1.26
VDD1 (JP3) @ 0.63 V	0.65	0.05	0.03
VDD2 (JP1) @ 0.90 V	0.99	66.65	65.98
VDDN (JP2) @ 1.1 V	1.10	24.42	26.86
-	-	-	-
Total	-	-	110.84

The ONLY_FRO0 compiler flag enables quick switching between the first-phase and second-phase optimizations. The second-phase optimization focuses on switching the clock source from PLLs and FROs to a single FRO0 and powers down all the PLLs and unused FRO1 and FRO2. We must ensure that the memory (PSRAM - XSPI2), display, MIPI, and the media domain clocks are switched to FRO0. The second-phase optimization can be enabled by adding the ONLY_FRO0 compiler flag to the preprocessor settings of core 0. After testing with the second-phase optimization, the power data was decreased by 50 % again, as shown in Table 4.

Table 4. RT798 (camera demo) @ 30 FPS SDK_25_06_00 tested using MCUXpresso on PSRAM based on power optimized changes that drives the display and camera using FRO0 (192 MHz) only as the clock source

-	V	mA	mW
VDD1V8 (JP19)	1.80	0.50	0.90

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 4. RT798 (camera demo) @ 30 FPS SDK_25_06_00 tested using MCUXpresso on PSRAM based on power optimized changes that drives the display and camera using FRO0 (192 MHz) only as the clock source*...continued*

-	V	mA	mW
VDD1V8 MIPI (R311)	1.80	0.70	1.26
VDD1 (JP3) @ 0.63 V	0.65	0.05	0.03
VDD2 (JP1) @ 0.9 V	0.90	50.40	45.36
VDDN (JP2) @ 1.1 V	1.09	6.40	6.98
-	-	-	-
Total	-	-	54.53

After propagating the power optimization changes from other use cases and examples, the camera capture demo has a power number that makes the i.MX RT700 MCU a good candidate for camera-based applications.

9 Conclusion

This application note contains all the key information of emulating the parallel camera interface using the FlexIO and eDMA of the i.MX RT700 and create a live video demo using the MIMXRT700-EVK. It demonstrates the capture of video frames from the OV7670 sensor module and rendering them on a rectangular LCD display connected to the MIPI-DSI of the MIMXRT700-EVK.

10 References

- *i.MX RT700 Reference Manual* (document IMXRT700RM)
- *Omnivision OV7670 Data Sheet* (https://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf)

11 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

12 Revision history

Table 5. Revision history

Document ID	Release date	Description
AN14836 v.1.0	10 November 2025	Initial public release

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Legal information

Definitions

Draft — A draft status on a document indicates that 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 in a draft version of a document and shall have no liability for the consequences of use of such information.

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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

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 and its suppliers accept 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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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 B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

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

NXP — wordmark and logo are trademarks of NXP B.V.

IAR — is a trademark of IAR Systems AB.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Intel, the Intel logo, Intel Core, OpenVINO, and the OpenVINO logo — are trademarks of Intel Corporation or its subsidiaries.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Contents

1 Introduction 2
2 Hardware and software required 2
3 Omnivision Sensor Module OV7670 2
3.1 Hardware workaround for MIMXRT700-EVK Rev. A1 3
4 Bitbucket project source code repositories 6
5 Overview of i.MX RT700 Flexible IO (FlexIO) 6
5.1 FlexIO features 7
5.2 Shifters and timers 8
5.3 Serial and parallel transfers 9
5.4 General configurations and operations 10
6 Parallel camera interface 10
7 Using FlexIO to emulate parallel camera interface 12
7.1 Operation principles of FlexIO camera interface 12
7.2 Demo application hardware platform 12
7.3 Application software pipeline 14
7.4 FlexIO configuration for parallel camera interface 15
7.5 FlexIO timer configuration for camera XCLK ... 16
7.6 eDMA software configuration 17
8 Power measurement data 17
9 Conclusion 19
10 References 19
11 Note about the source code in the document 19
12 Revision history 20
Legal information 21

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2025 NXP B.V. **All rights reserved.**