AN14191

How to Use SmartDMA to Implement Camera Interface in MCXN MCU

Rev. 1 — 18 January 2024 Application note

Document information

Information	Content
Keywords	AN14191, MCXN947, SmartDMA
Abstract	This application note describes the parallel interface for the camera solution in MCXN947. It includes the introduction of camera interface, features, API routines, and demo.

1 Introduction

This application note describes the parallel interface for the camera solution in MCXN947. It includes the introduction of camera interface, features, API routines, and demo. MCXN947 contains a coprocessor SmartDMA, which can be used to implement the camera interface.

2 Target applications

The camera interface can be used as an important part of camera usage for the following:

• Object detection

• Gesture recognition

• Color recognition

• QR code scanning

3 Introduction of camera interfaces

A typical camera interface supports at least one parallel interface, although nowadays many camera interfaces begin to support the MIPI CSI interface.

The camera parallel interface consists of the following lines:

- Data line (D[0:7]): These parallel data lines carry pixel data. The data transmitted on these lines change with every Pixel Clock (PCLK).
- Horizontal Sync (HSYNC): This is a special signal that goes from the camera sensor. An HSYNC indicates that one line of the frame is transmitted.
- Vertical Sync (VSYNC): This signal is transmitted after the entire frame is transferred. This signal is often a way to indicate that one entire frame is transmitted.
- Pixel Clock (PCLK): This pixel clock changes on every pixel.

This application note only focuses on the Digital-Video-Port (DVP) interface which is a parallel interface.

4 Introduction of MCX Nx4x MCU

The MCX Nx4x MCU has up to 2 MB flash, up to 512 kB SRAM, 150 MHz system clock, SmartDMA FlexIO, QSPI interface, and I2C interface. The SmartDMA can be used to transfer data from the camera interface to the internal RAM. The FlexIO can be used to transfer the data in RAM to an LCD interface. The QSPI can be used to extend the memory to store the frame data. The internal SRAM can be used to store the temporary frame data. The I2C interface can be used to configure the camera module.

The application note focuses on camera implementation. For the FlexIO implemented LCD interface, see *Using FlexIO to Drive 8080 Bus Interface LCD Module* (document AN5313).

5 Features of camera interface

The supported formats are:

- RGB565.
- The maximum image transfer rate is 30 fps for VGA (640x480). For small RAM parts, reduce the image size and frame rate.
- OV7670 is the tested camera module.

- Other camera modules can be supported if they provide the same signal timing.

6 Function description

This chapter describes the functions.

6.1 How to work for SmartDMA

The SmartDMA can access the GPIO in a single system cycle. It reads the data from the camera and stores the data in the RAM. After that, the FlexIO can send the data to an LCD screen.

Some configurations must be made before using the SmartDMA. They include the pin configuration, clock enable, dedicated processor enable, interrupt enable, and so on.

The SmartDMA shares the system clock with the Arm core. To speed up the processing time, the system clock has better to be configured to 150 MHz.

6.2 Camera clock source

The camera needs a roughly 6-MHz clock source, which is provided by the CLKOUT signal from the MCU. Different clock sources can get a different frame-rate output.

6.3 MCU8080 LCD interface

The FlexIO can implement an LCD interface. See *Using FlexIO to Drive 8080 Bus Interface LCD Module* (document AN5313).

6.4 I2C interface

The camera is configured through an I2C interface in the MCU. Before the camera runs, it must be configured by the Arm core through the I2C peripheral.

6.5 Memory usage

In this application note, the resolution of the LCD screen is 480x320. The OV7670 camera module output resolution is 640x480.The SmartDMA can crop the size of the image output and only store the frame size with a resolution of 480x320 in the RAM to adapt to the size of the LCD. This is the flexibility of the SmartDMA. Therefore, the required memory space is 300 kB of RAM.

To implement the sampling of different sizes, use different API functions implemented by the SmartDMA.

A ping-pong buffer can be used to store the partial frame data. This requires timely processing of data to not affect the subsequent data storage.

Additionally, the instruction code of the camera engine must run in the RAM for high performance. This solution uses the SRAMX to store the camera engine code. Because the implementation of SmartDMA is related to the RAM address where the running code is located, the running address must be fixed at the start position of SRAMX. If running SmartDMA code at other RAM addresses, regenerate the code array.

6.6 Other supported camera modules

Other camera modules can be supported if they provide the same signal timing.

1. The camera module must be configured in the RGB565 mode with the timing diagram shown in Figure 1.

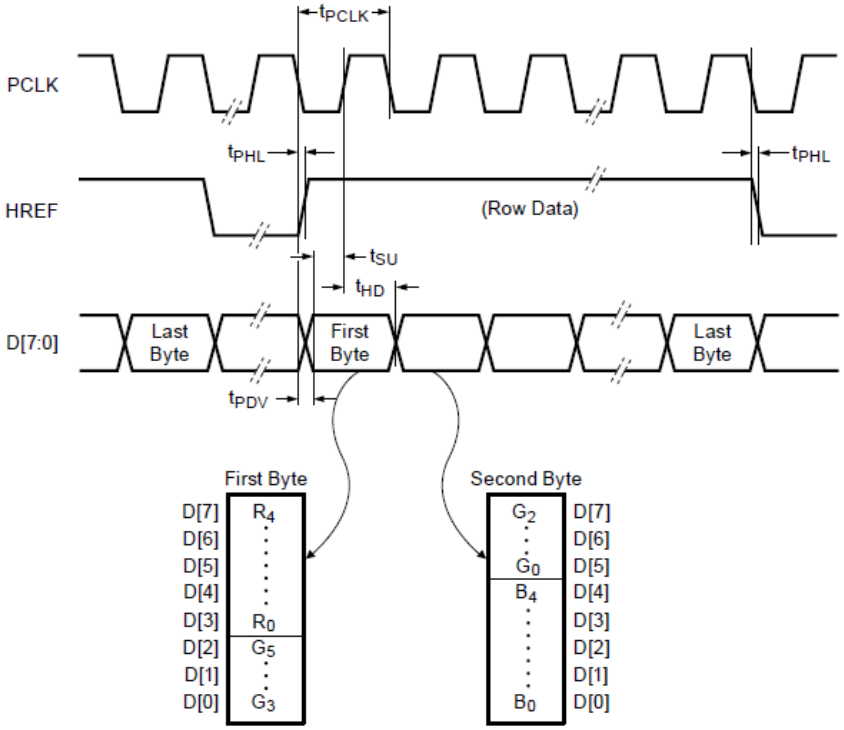

Figure 1. Timing diagram

2. The resolution can be configured as VGA (640x480), QVGA (320x240), QQVGA (160x120), and so on. The SmartDMA code must be modified to adapt to different resolutions.

7 Pin description

This chapter describes the pins.

7.1 Interface connection

The SmartDMA can access 32 GPIO pins in MCXN947 and use 8 GPIO pins in parallel to read the camera data. The MCU uses I2C to configure the camera. The VSYNC, HREF, PCLK, and PWDN signals can be controlled by SmartDMA. The camera module can be powered by the MCU board with 3.3 V.

7.2 Interface requirements

The D0-D7 should be connected to SmartDMA D0-D7 for the byte reading of data. SIOC and SIOD should be connected to the I2C interface of the MCU for configuration. VSYNC, HREF, and PCLK should be connected to the pins of Port0 to trigger SmartDMA. XCLK should be connected to the clock output pin of the MCU.

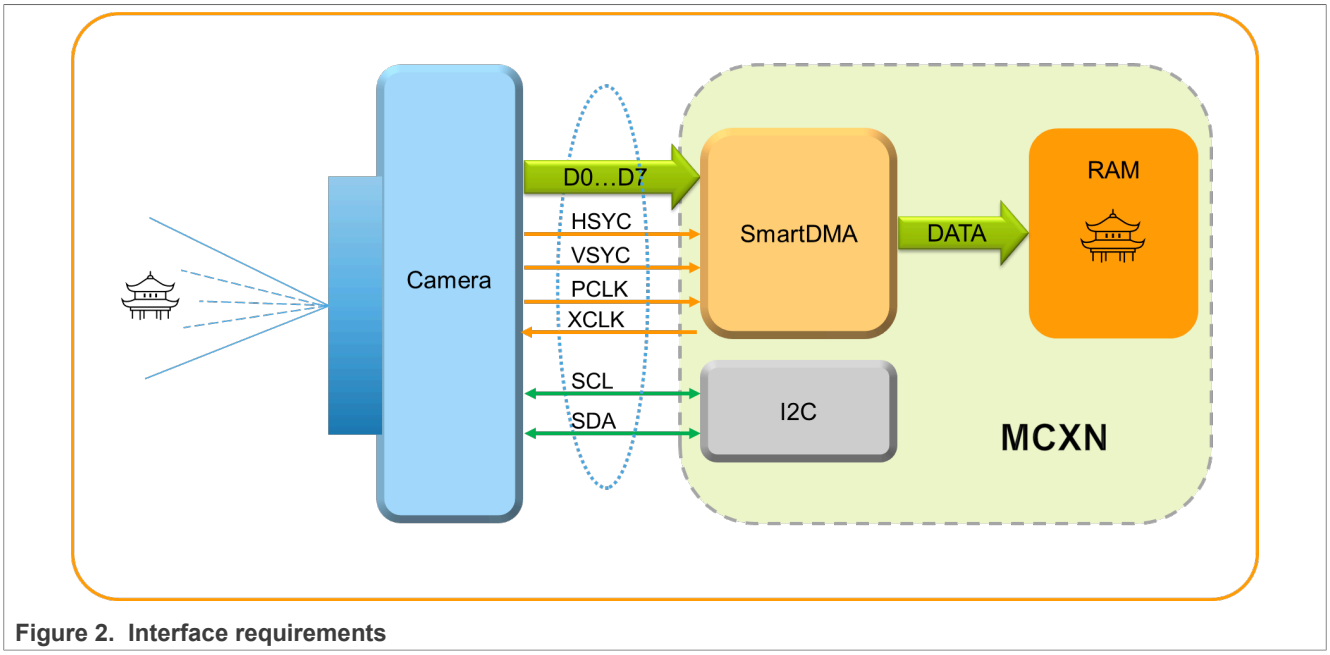

Figure 2. Interface requirements

8 Software

This chapter describes the software.

8.1 Demo code

This application note comes with the software code named "frdm_mcxn947_SmartDMA_camera". It is generated using the MCUXpresso IDE.

8.2 SmartDMA code array

As a reduced-instruction-set core, the SmartDMA must execute assembly instructions to perform tasks. To reduce the difficulty of user study, the SmartDMA code is presented in a data-array form in this application. Simply set the SmartDMA address of the array and the task can be executed. Since there are absolute address jumps instructions, the code array must be placed at a specified RAM address. The code array can be easily integrated in Keil IDE, MCUXpresso IDE, and IAR IDE. In the software code of this application note, the name of this array is "s_smartdmaCameraFirmware".

8.3 API routine

The main purposes of the API routines include the following:

- Enable the SmartDMA clock.
- Configure the IO as a camera interface function.
- Initialize the I2C interface.
- Enable the interrupt of SmartDMA to tell that the Arm core data is ready.
- Initialize and start the SmartDMA.
- LCD initialization.
- LCD refresh.

8.4 API routine description

Table 1. API routine

Routine	Description
SMARTDMA_InitWithoutFirmware	Initialize the SmartDMA
SMARTDMA_InstallFirmware	Install the firmware
SMARTDMA_InstallCallback	Install the complete callback function
SMARTDMA_Boot	Boot the SmartDMA to run a program
SMARTDMA_Deinit	Deinitialize the SmartDMA
SMARTDMA_Reset	Reset the SmartDMA
SMARTDMA_HandleIRQ	SmartDMA IRQ
SmartDMA_camera_callback	SmartDMA interrupt callback
lcd_impl_init	LCD driver initialization
st7796_lcd_init	LCD initialization
st7796_lcd_load	LCD refresh
Ov7670_Init	Camera module initialization

8.5 Detailed code description

This section describes the code in detail.

8.5.1 System clock

SmartDMA needs limited time to store the data when every pixel edge comes. If the clock frequency of the engine is higher, the time cost is shorter. In this solution, the system clock must be set to 150 MHz when the engine is running. The code to configure the system clock is as follows:

```
BOARD_BootClockPLL150M();
```

8.5.2 I2C interface

For MCXN947, "flexcomm7" is used as the I2C function to initialize the camera.

8.5.3 Pin function

Table 2. Pin function

MCXN947	Function number	Input/output	Description
P1_4	7 (camera function)	Input	Camera engine function DATA0
P1_5	7 (camera function)	Input	Camera engine function DATA1
P1_6	7 (camera function)	Input	Camera engine function DATA2
P1_7	7 (camera function)	Input	Camera engine function DATA3

Table 2. Pin function*...continued*

MCXN947	Function number	Input/output	Description
P3_4	7 (camera function)	Input	Camera engine function DATA4
P3_5	7 (camera function)	Input	Camera engine function DATA5
P1_10	7 (camera function)	Input	Camera engine function DATA6
P1_11	7 (camera function)	Input	Camera engine function DATA7
P0_4	0 (GPIO function)	Input	GPIO as VSYNC input
P0_11	0 (GPIO function)	Input	GPIO as HSYNC input
P0_5	0 (GPIO function)	Input	GPIO as pixel clock input
P2_2	CLKOUT function	Output	Clock input to camera
P3_2(FLEXCOM7)	I2C function	Input/output	I2C_SDA
P3_3(FLEXCOM7)	I2C function	Output	I2C_SCL

Note: *The camera function mode is 7 in the PORT register.*

8.5.4 LCD function

The LCD is used to display the video of camera in real time. FlexIO and DMA are used to drive the LCD. When SmartDMA completes storing the camera data in the buffer, it gives an interrupt trigger to the Arm core. In the Arm core interrupt, the complete flag is set. Once the flag is set, the "st7796_lcd_load" route is called to refresh the LCD. For the LCD implementation, see the other application notes.

8.5.5 SmartDMA_camera_callback

As the other peripheral handler, the SmartDMA handler is implemented by the Arm core when SmartDMA finishes the storage operation.

In the initialization stage, the callback route of the handler is installed. In the callback routine, a flag is set to 1. In the "1" routine, the refreshing operation can be allowed when the flag turns to logic one.

8.5.6 Data buffer

There must be 300 kB of RAM space for one frame of video (480x320 resolution) and MCXN947 has about 512 kB of RAM. A double buffer is not possible for the whole frame mode. Only one buffer is used. Because the LCD refresh time (16 ms) is shorter than the data storage time (66 ms), Arm always reads the data for LCD refresh earlier than it is stored by SmartDMA. The LCD always displays the previous frame based on the data stored in RAM. Therefore, the media data is not lost.

8.5.7 Timing

The LCD always displays the previous frame data from the camera. Before displaying, the data stored must be optimized by the coprocessor for exchanging the high and low bytes of every pixel. Because the speed of the LCD module displaying is higher than the speed of the camera interface reading the data, a single data buffer is used in this application. While the current frame data is stored, the LCD displays the previous frame data.

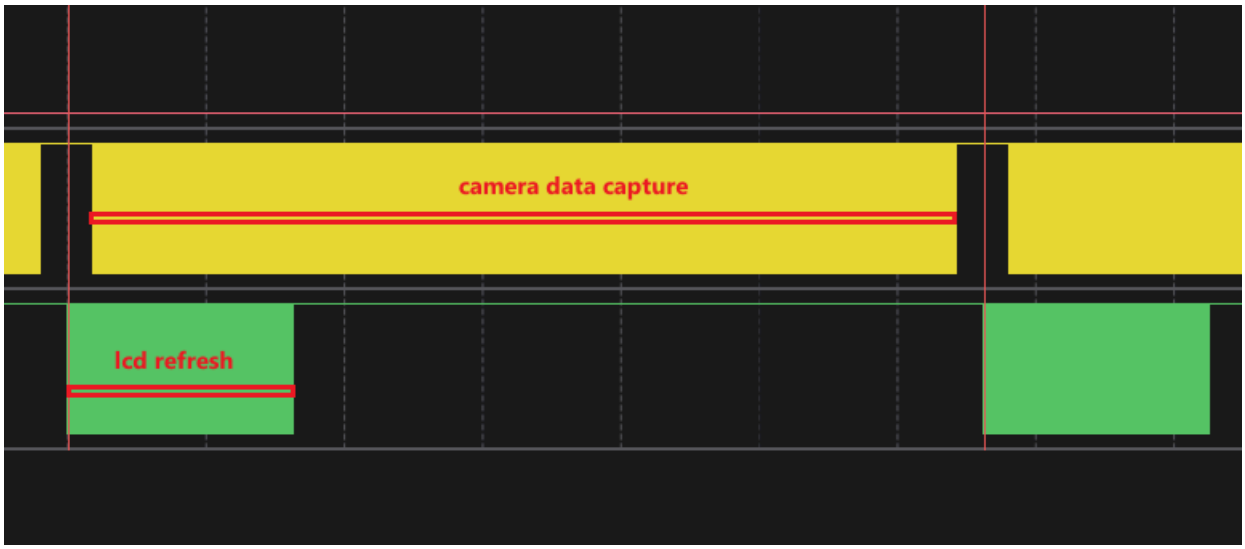

Figure 3. Timing

8.5.8 Demo

1. Build and compile the project.
2. Plug one side of the USB cable to the PC USB port and the other side to the debug link port in the FRDM-MCXN947 board and then download the firmware into the MCU. Disconnect the cable from the FRDM-MCXN947 board.
3. Connect the camera to the MCU by referring to the connection of interface in Figure 4.
4. Connect the LCD panel to the FlexIO LCD port on the FRDM-MCXN947 board.
5. The LCD displays the video frame from the camera, as shown in Figure 4.

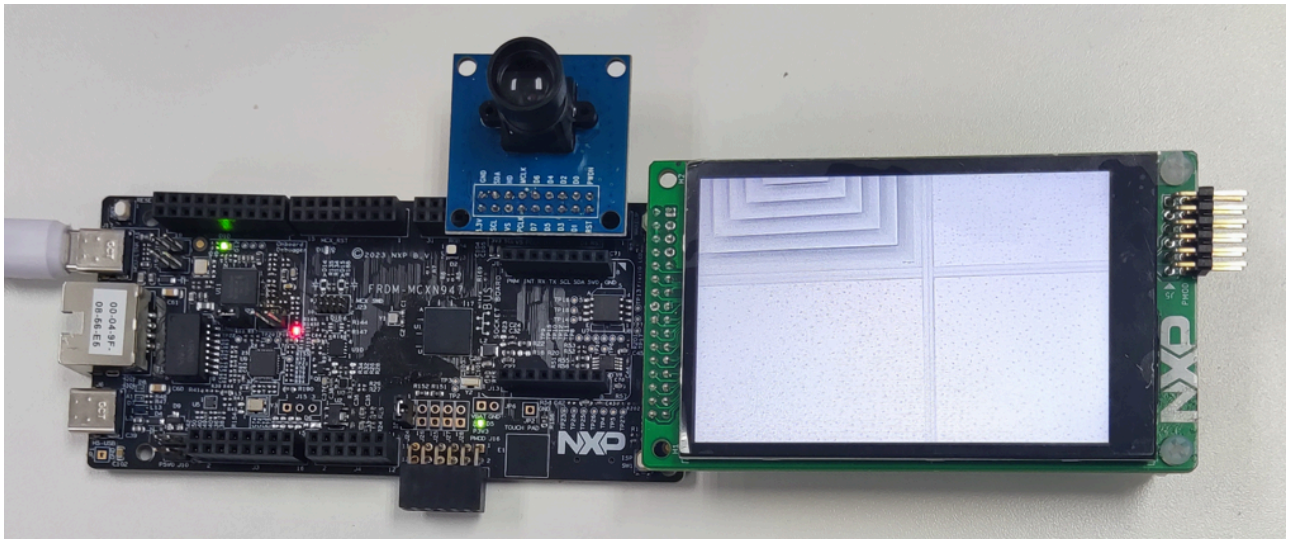

Figure 4. Demo

9 Note about the source code in the document

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

10 Revision history

Table 3 summarizes the changes done to this document.

Table 3. Revision history

Revision number	Release date	Description
1	18 January	Initial external release.

Contents

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

Date of release: 18 January 2024
Document identifier: AN14191