AN14588

Connect USB camera and show picture on LCD panel using i.MX RT1170

Rev. 1.0 — 12 August 2025

Application note

Document information

Information	Content
Keywords	AN14588, USB camera, RT1170, LCD
Abstract	This document describes how to connect a USB camera to get the picture and show it on the LCD panel in real time using i.MX RT1170 EVK.

1 Introduction

This document provides information on how to connect a USB camera to get the picture and show it on the LCD panel in real time using i.MX RT1170 EVK as shown in Figure 1.

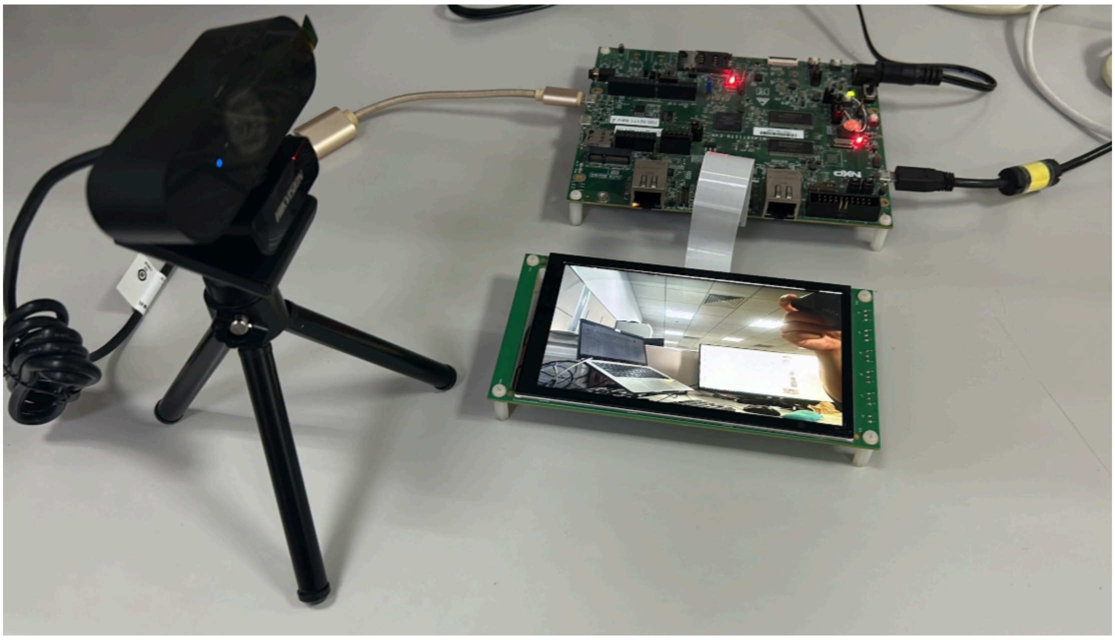


Figure 1. Connect USB camera and show picture on LCD panel using i.MX RT1170

2 How system runs

The way how the system runs is shown on Figure 2

At first, the 640*480 resolution picture in YUYV format is sent from the camera. Then it is scaled to be 1280*720 and rotated 270 degrees clockwise by PXP. The output picture size is 720*1280, which is displayed on an LCD panel.

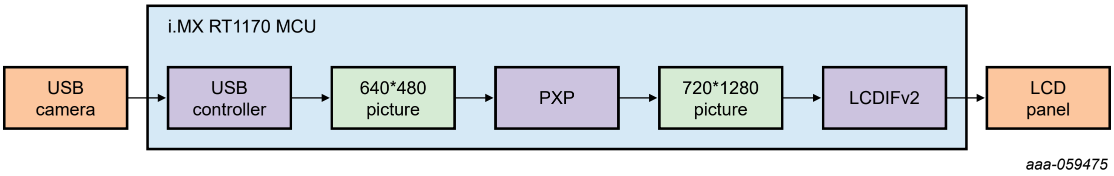


Figure 2. How system runs

3 Background knowledge

This section provides details on what background knowledge for USB Camera and PXP is required.

3.1 Background knowledge for USB camera

This chapter provides background information for the USB camera.

3.1.1 Basic parameters of a USB camera

Picture format

Typically, the picture format from a USB camera supports YUV and MJPEG. For real time application, use the YUV format to avoid decoding by CPU, which would cause delay and increase the CPU loading.

Resolution

The USB camera typically supports numerous resolutions, there are several examples below:

- w = 1920, h = 1080,
- w = 640, h = 480,
- w = 640, h = 360,
- w = 1280, h = 720,
- w = 1280, h = 960,
- w = 352, h = 288,
- w = 320, h = 240,
- w = 160, h = 120,

Frame rate

A USB camera reports the frame rate that it supports, there are several examples below:

- 30 fps
- 25 fps

3.1.2 Get information from class specific (CS) interface descriptor

The USB camera descriptor is organized as shown in Figure 3. There are format descriptors and frame descriptors. A USB camera has one or multi format descriptors, and one format descriptor includes one or multi frame descriptors.

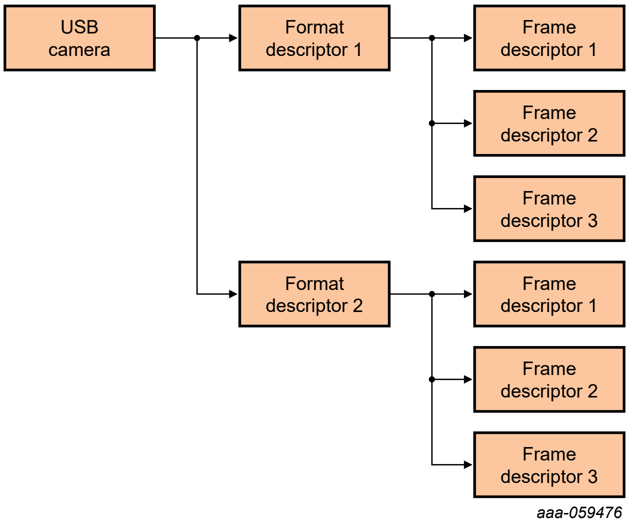


Figure 3. Descriptor structure

An example for a format descriptor is shown below.

Table 1. CS_INTERFACE Descriptor (27 bytes)

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bLength	8	1120	0x1B	0x1B	The descriptor size is 27 bytes.
bDescriptorType	8	1128	0x24	0x24	`CS_INTERFACE` descriptor type
bDescriptorSub Type1	8	1136	0x04	0x04	`VS_FORMAT_UNCOMPRESSED` descriptor subtype
bFormatIndex	8	1144	0x01	0x01	Index of this format descriptor
bNumFrame Descriptors	8	1152	0x03	0x03	Various frame descriptors following what corresponds to this format
guidFormat	32	1160	59555932-1000-800000AA389871	0x32595559	The globally unique identifier used to identify stream-encoding format 595559321000-800000AA-389B71
bBitsPerPixel	8	1288	0x10	0x10	Number of bits per pixel used to specify color in the decoded video frame
bDefaultFrame Index	8	1296	0x02	0x02	Optimum Index (used to select resolution) for this stream
bAspectRatioX	8	1304	0x00	0x00	The X dimension of the picture aspect ratio
bAspectRatioY	8	1312	0x00	0x00	The Y dimension of the picture aspect ratio
bmInterlaceFlags	8	1320	0x00	0x00	Specifies interlace information DO: Interlaced stream or variable 0 DI: Fields per frame 0 D2: Field 1 first O D3: Reserved O D5..4: Field pattern O

Table 1. CS_INTERFACE Descriptor (27 bytes)*...continued*

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
					D7..6: Display Mode O
bCopyProtect	8	1328	0x00	0x00	Specifies whether duplication of the video stream is restricted D0: video stream is restricted 0

An example for a frame descriptor is as below.

Table 2. CS_INTERFACE Descriptor (30 bytes)

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bLength	8	1576	0x1E	0x1E	The descriptor size is 30 bytes.
bDescriptorType	8	1584	0x24	0x24	`CS_INTERFACE` Descriptor Type
bDescriptorSub Type1	8	1592	0x05	0x05	`VS_FRAME_UNCOMPRESSED` descriptor subtype
bFrameIndex	8	1600	0x02	0x02	Index of this frame descriptor
bmCapabilities	8	1608	0x00	0x00	D0: Still image supported 0 D7..1: Reserved 0
wWidth	16	1616	0x0280	0x0280	Width of decoded bitmap frame in pixels
wHeight	16	1632	0x01E0	0x01E0	Height of decoded bitmap frame in pixels
dwMinBitRate	32	1648	01194000	0x01194000	Specifies the minimum bit rate at the longest frame interval in units of bit/s at which the data can be transmitted
dwMaxBitRate	32	1680	034BC000	0x034BC000	Specifies the maximum bit rate at the shortest frame interval in units of bit/s at which the data can be transmitted

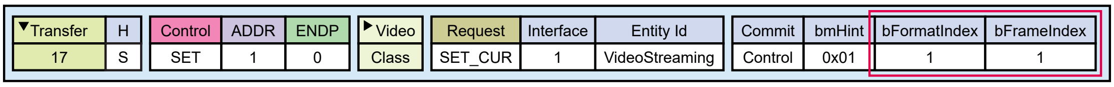
Table 2. CS_INTERFACE Descriptor (30 bytes)*...continued*

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
dwMaxVideo FrameBufferSize	32	1712	00096000	0x00096000	Specifies the maximum number of bytes that the compressor produces for a video frame or still image
dwDefaultFrame Interval	32	1744	00061A80	0x00061A80	Specifies the frame interval that the device would like to indicate for use by default
bFrameInterval Type	8	1776	0x01	0x01	Indicates how the frame interval can be programmed 0: Continuous frame interval 1..255: The number of discrete frames
dwFrame Interval(1)	32	1784	00061A80	0x00061A80	Longest frame interval supported (at lowest frame rate), in 100 ns units

From the CS interface descriptors, the important parameters are below:

- Bits per pixel: 16
- Frame format: `VS_FRAME_UNCOMPRESSED`
- Resolution: 640(0x280)*480(0x1E0)
- Interval: 40,000,000 (0x61A80) ns = 40 ms (Frame rate is 1000/40 = 25 fps)
- Buffer size: 614,400(0x96, 000) B = 640*480*2

The information would be decoded in the project code. Specify the working format CS descriptor and frame CS descriptor by the `SET_CUR` request according to the application requirement.



Transfer	H	Control	ADDR	ENDP	Video	Request	Interface	Entity Id	Commit	bmHint	bFormatIndex	bFrameIndex
17	S	SET	1	0	Class	SET_CUR	1	VideoStreaming	Control	0x01	1	1

Figure 4. SET_CUR request

3.1.3 Interface with multialternate setting

An interface has one or multiple alternate settings. Each alternate setting operates at a different bandwidth.

An example for an interface with an alternate setting 1 to support 3 ISO transaction per frame is shown below:

Table 3. INTERFACE descriptor (9 bytes)

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bLength	8	6968	0x09	0x09	The descriptor size is 9 bytes.
bDescriptorType	8	6976	0x04	0x04	INTERFACE descriptor type
bInterfaceNumber	8	6984	0x01	0x01	The number of this interface is 1.
bAlternateSetting	**8**	**6992**	**0x01**	**0x01**	**The value used to select the alternate setting for this interface is 1.**
bNumEndpoints	8	7000	0x01	0x01	The number of endpoints used by this interface is 1 (excluding endpoint zero)
bInterfaceClass	8	7008	0x0E	0x0E	The interface implements the Video class
bInterfaceSub Class	8	7016	0x02	0x02	The interface implements the SC_ VIDEOSTREAMING Subclass
bInterfaceProtocol	8	7024	0x00	0x00	The interface uses the `PC PROTOCOL UNDEFINED` Protocol
iInterface	8	7032	0x04	0x04	The interface string descriptor index is 4.

Table 4. ENDPOINT descriptor (7 bytes)

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bLength	8	7040	0x07	0x07	The descriptor size is 7 bytes.
bDescriptorType	8	7048	0x05	0x05	ENDPOINT Descriptor Type
bEndpointAddress	8	7056	0x82	0x82	This is an IN endpoint with endpoint number 2
bmAtributes	8	7064	0x05	0x05	Types - Transfer: ISOCHRONOUS Sync: Async Usage: Data EP
wMaxPacketSize	**16**	**7072**	**0x1400**	**0x1400**	**The maximum packet size for**

Table 4. ENDPOINT descriptor (7 bytes)*...continued*

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
					this endpoint is 1024. If Hi-Speed, 2 additional transactions per frame.
bInterval	8	7088	0x01	0x01	The polling interval value is every 1 frame. If Hi-Speed, every 1 uFrames

An example for an interface with an alternate setting 2 to support 2 ISO transaction per frame is shown below:

Table 5. INTERFACE descriptor (9 bytes)

Field	Length (bits) 8	Offset (bits)	Decoded	Hex Value	Description
bLength	8	7096	0x09	0x09	The descriptor size is 9 bytes.
bDescriptorType	8	7104	0x04	0x04	INTERFACE descriptor type
bInterfaceNumber	8	7112	0x01	0x01	The number of this interface is 1.
bAlternateSetting	**8**	**7120**	**0x02**	**0x02**	**The value used to select the alternate setting for this interface is 2.**
bNumEndpoints	8	7128	0x01	0x01	The number of endpoints used by this interface is 1 (excluding endpoint zero)
bInterfaceClass	8	7136	0x0E	0x0E	The interface implements the Video class
bInterfaceSub Class	8	7144	0x02	0x02	The interface implements the SC VIDEOSTRE AMING Subclass
bInterfaceProtocol	8	7152	0x00	0x00	The interface uses the PC PRO TOCOL UND EFINED Protocol
iInterface	8	7160	0x04	0x04	The interface string descriptor index is 4

Table 6. ENDPOINT descriptor (7 bytes)

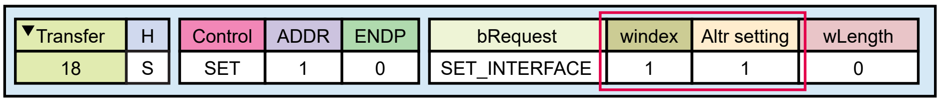
Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bLength	8	7168	0x07	0x07	The descriptor size is 7 bytes.

Table 6. ENDPOINT descriptor (7 bytes)*...continued*

Field	Length (bits)	Offset (bits)	Decoded	Hex Value	Description
bDescriptorType	8	7176	0x05	0x05	ENDPOINT Descriptor Type
bEndpointAddress	8	7184	0x82	0x82	This is an IN endpoint with endpoint number 2.
bmAttributes	8	7192	0x05	0x05	Types - Transfer ISOCHRONOUS Sync: Async Usage: Data EP
wMaxPacketSize	**16**	**7200**	**0x0C00**	**0x0C00**	**The maximum packet size for this endpoint is 1024 Bytes. If Hi-Speed, 1 additional transaction per frame.**
bInterval	8	7216	0x01	0x01	The polling interval value is every 1 Frame. If Hi-Speed, every 1 u Frames.

Note: *Verify that the bandwidth is sufficient to support the intended functionality. For some USB camera, multi-ISO in one micro frame is a must; otherwise, only part of the picture is transferred from the USB camera.*

To select a high-bandwidth alternate setting, use the `SET_INTERFACE` request as shown below.

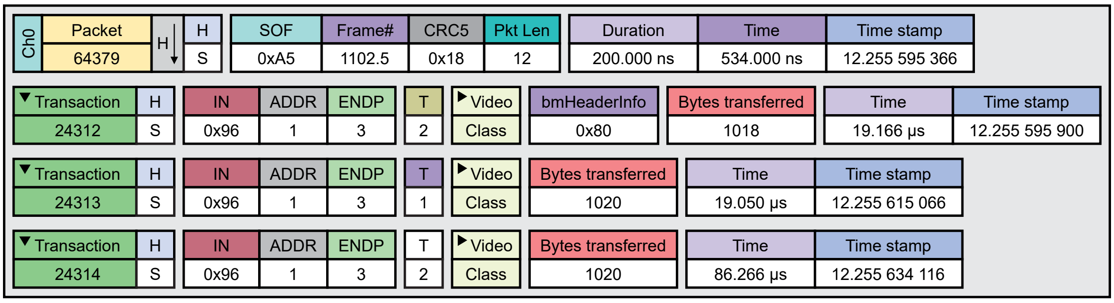


Transfer	H	Control	ADDR	ENDP	bRequest	windex	Altr setting	wLength
18	S	SET	1	0	SET_INTERFACE	1	1	0

aaa-059478

Figure 5. SET_INTERFACE request

Below is an example for image data transfer. There are 3 ISO transactions in 1 uFrame.



Ch0	Packet	H	H	SOF	Frame#	CRC5	Pkt Len	Duration	Time	Time stamp
	64379		S	0xA5	1102.5	0x18	12	200.000 ns	534.000 ns	12.255 595 366

Transaction	H	IN	ADDR	ENDP	T	Video	bmHeaderInfo	Bytes transferred	Time	Time stamp
24312	S	0x96	1	3	2	Class	0x80	1018	19.166 µs	12.255 595 900

Transaction	H	IN	ADDR	ENDP	T	Video	Bytes transferred	Time	Time stamp
24313	S	0x96	1	3	1	Class	1020	19.050 µs	12.255 615 066

Transaction	H	IN	ADDR	ENDP	T	Video	Bytes transferred	Time	Time stamp
24314	S	0x96	1	3	2	Class	1020	86.266 µs	12.255 634 116

aaa-059479

Figure 6. Example for image data transfer

3.2 Background knowledge for PXP

PXP is a 2-D image accelerator provided by NXP. It supports color space conversion, scaling, rotation, and blending.

The block diagram is shown in Figure 7.

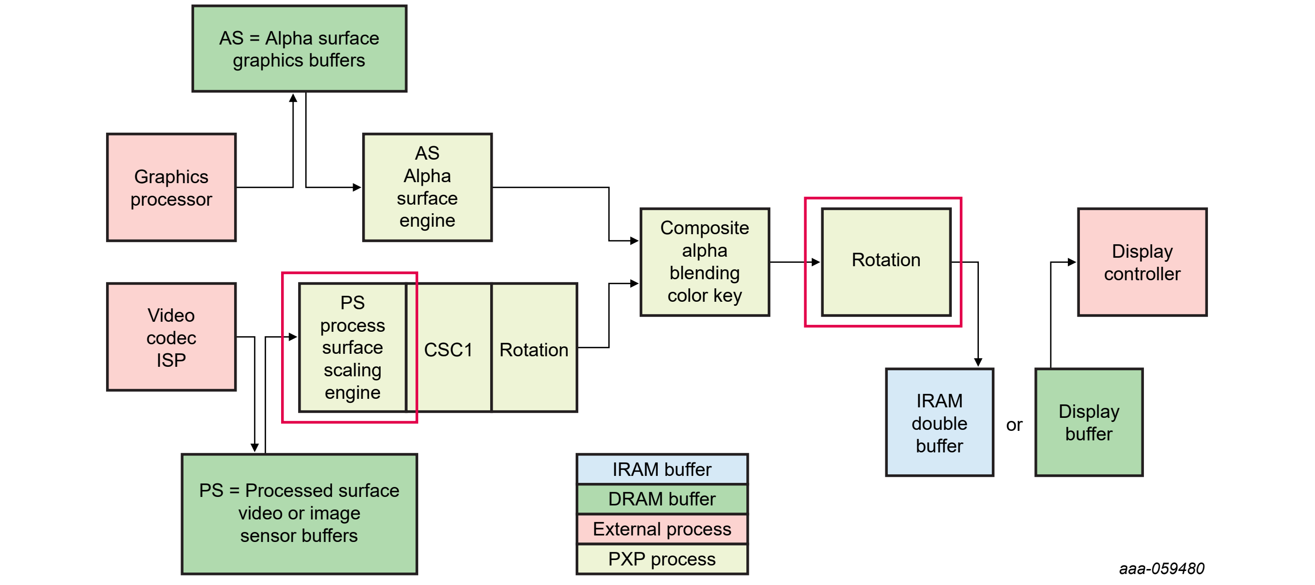


Figure 7. PXP block diagram

There are three domains in PXP: PS, AS, and OUTPUT.

If blending is not necessary, use the PS domain as input. From the PS domain input path, the image can be scaled and rotated.

In this document, the scaling engine and rotation unit marked in the red rectangle in the picture above are used.

The input image from the USB camera is 640 * 480. Then it is scaled to be 1280 * 720 and rotated to be 720*1280. Finally, the 720*1280 picture is sent to an LCD panel for display.

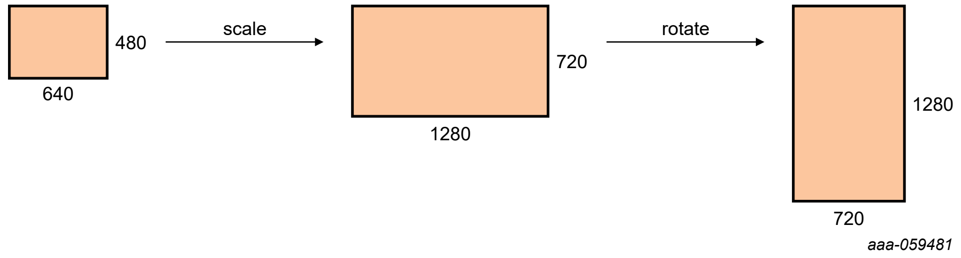


Figure 8. Scaling and rotation

To achieve this, configure the PS region correctly. For the case above, the PS region should be configured to be (0，0，1280-1，1280-1), which means, the upper left X, upper left Y, the lower right X, the lower right Y. After this configuration, the PS region should be as shown in Figure 9.

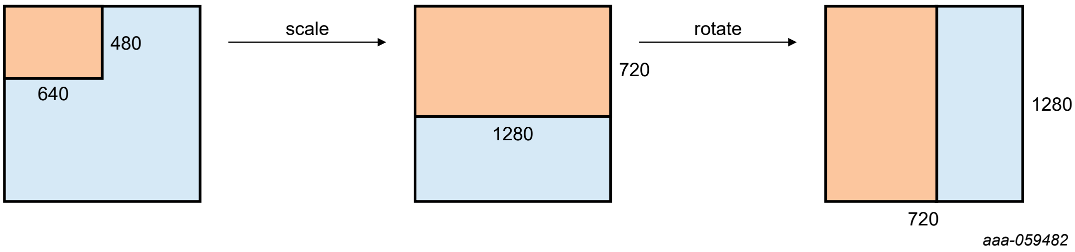


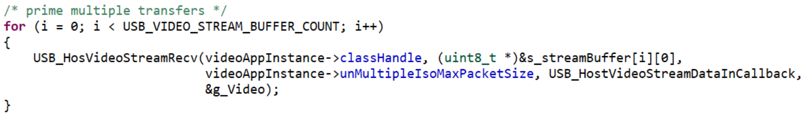
Figure 9. Configure the PS region

Here, since we configure the PS region to be a square of 1280*1280, then in the process of scaling and rotation, the whole picture is within the PS region range. For the range beyond PS region, PXP does not guarantee the data is correct. This is important in the PXP PS region configuration.

4 Keycode analysis

The process of getting picture data from the USB camera is as described below.

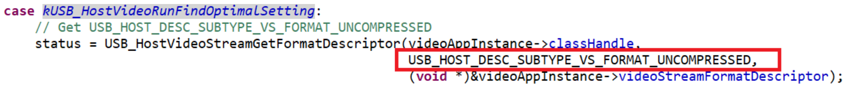
1. After enumeration, call `USB_HosVideoStreamRecv()` to prime USB RX transfers.



```
/* prime multiple transfers */
for (i = 0; i < USB_VIDEO_STREAM_BUFFER_COUNT; i++)
{
    USB_HosVideoStreamRecv(videoAppInstance->classHandle, (uint8_t *)&s_streamBuffer[i][0],
                           videoAppInstance->unMultipleIsoMaxPacketSize, USB_HostVideoStreamDataInCallback,
                           &g_Video);
}
```

Figure 10. Call USB_HosVideoStreamRecv() to prime USB RX transfers

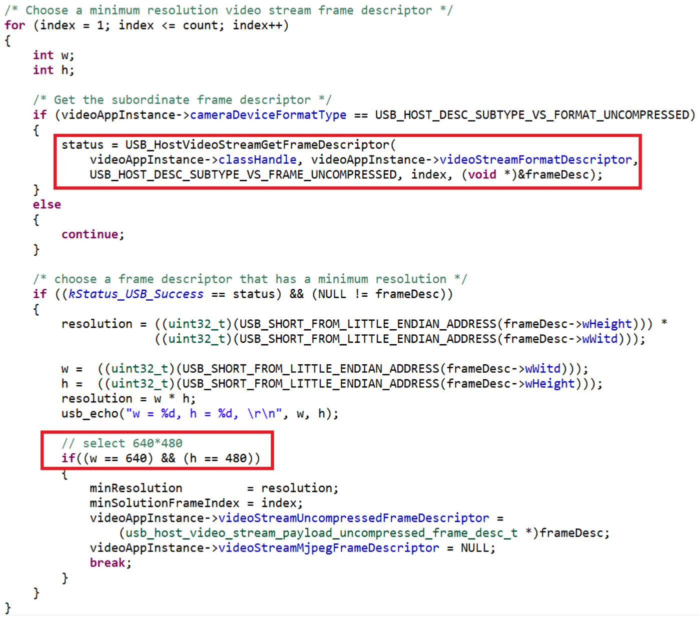
2. In the callback, call `USB_HosVideoStreamRecv()` again to start a new USB RX transfer as shown below.
3. Get an uncompressed format descriptor as shown in Figure 11



```
case kUSB_HostVideoRunFindOptimalSetting:
    // Get USB_HOST_DESC_SUBTYPE_VS_FORMAT_UNCOMPRESSED
    status = USB_HostVideoStreamGetFormatDescriptor(videoAppInstance->classHandle,
                                                    USB_HOST_DESC_SUBTYPE_VS_FORMAT_UNCOMPRESSED,
                                                    (void *)&videoAppInstance->videoStreamFormatDescriptor);
```

Figure 11. Get an uncompressed format descriptor

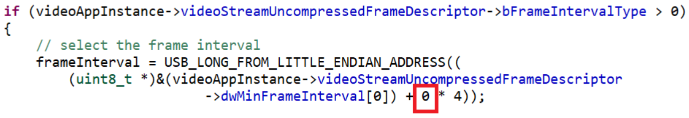
4. Get a frame descriptor and choose the one with the required resolution.



```
/* Choose a minimum resolution video stream frame descriptor */
for (index = 1; index <= count; index++)
{
    int w;
    int h;

    /* Get the subordinate frame descriptor */
    if (videoAppInstance->cameraDeviceFormatType == USB_HOST_DESC_SUBTYPE_VS_FORMAT_UNCOMPRESSED)
    {
        status = USB_HostVideoStreamGetFrameDescriptor(
            videoAppInstance->classHandle, videoAppInstance->videoStreamFormatDescriptor,
            USB_HOST_DESC_SUBTYPE_VS_FRAME_UNCOMPRESSED, index, (void *)&frameDesc);
    }
    else
    {
        continue;
    }

    /* choose a frame descriptor that has a minimum resolution */
    if ((kStatus_USB_Success == status) && (NULL != frameDesc))
    {
        resolution = ((uint32_t)(USB_SHORT_FROM_LITTLE_ENDIAN_ADDRESS(frameDesc->wHeight))) *
                     ((uint32_t)(USB_SHORT_FROM_LITTLE_ENDIAN_ADDRESS(frameDesc->wWitd)));

        w =  ((uint32_t)(USB_SHORT_FROM_LITTLE_ENDIAN_ADDRESS(frameDesc->wWitd)));
        h =  ((uint32_t)(USB_SHORT_FROM_LITTLE_ENDIAN_ADDRESS(frameDesc->wHeight)));
        resolution = w * h;
        usb_echo("w = %d, h = %d, \r\n", w, h);

        // select 640*480
        if((w == 640) && (h == 480))
        {
            minResolution         = resolution;
            minSolutionFrameIndex = index;
            videoAppInstance->videoStreamUncompressedFrameDescriptor =
                (usb_host_video_stream_payload_uncompressed_frame_desc_t *)frameDesc;
            videoAppInstance->videoStreamMjpegFrameDescriptor = NULL;
            break;
        }
    }
}
```

Figure 12. Get frame descriptor

5. Select a frame interval



```
if (videoAppInstance->videoStreamUncompressedFrameDescriptor->bFrameIntervalType > 0)
{
    // select the frame interval
    frameInterval = USB_LONG_FROM_LITTLE_ENDIAN_ADDRESS((
        (uint8_t *)&(videoAppInstance->videoStreamUncompressedFrameDescriptor
                     ->dwMinFrameInterval[0]) + 0 * 4));
```

Figure 13. Select a frame interval

6. Get a picture from a USB camera. When a picture from a USB camera is received, it is detected by the `USB_HostVideo_CheckOneFrameReady()` function, which processes the picture from the USB camera.
7. Process picture by the PXP. The function `APP_Scale_Rotate()` would be called to implement scaling and rotation by PXP.

5 Skills for optimization in MCUXpresso

In this case, the code must be optimized for performance; otherwise, visual artifacts (such as intermittent strips) may appear, as shown in Figure 14.

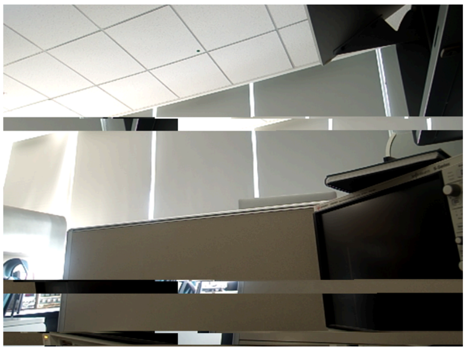


Figure 14. Strips

To avoid it, place some key code into TCM instead of running it in flash.

This is done by linkscripts in MCUXpresso.

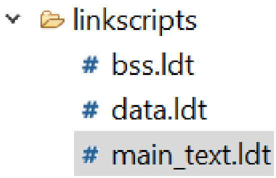


Figure 15. linkscripts in MCUXpresso

In `data.ldt`.

```
<#if memory.name=="SRAM_ITC_cm7">
*app.o(.text*)
*host_video.o(.text*)
*usb_host_hub_app.o(.text*)
*usb_host_hub.o(.text*)
*usb_host_video.o(.text*)
*usb_host_devices.o(.text*)
*usb_host_ehci.o(.text*)
*usb_host_framework.o(.text*)
*usb_host_hci.o(.text*)
*port.o(.text*)
```

```
*heap_4.o(.text*)
*croutine.o(.text*)
*event_groups.o(.text*)
*list.o(.text*)
*queue.o(.text*)
*stream_buffer.o(.text*)
*tasks.o(.text*)
*timers.o(.text*)
</#if>
```

In `main_text.ldt:`

```
*(EXCLUDE_FILE(
*app.o
*host_video.o
*usb_host_hub_app.o
*usb_host_hub.o
*usb_host_video.o
*usb_host_devices.o
*usb_host_ehci.o
*usb_host_framework.o
*usb_host_hci.o
*port.o
*heap_4.o
*croutine.o
*event_groups.o
*list.o
*queue.o
*stream_buffer.o
*tasks.o
*timers.o
) .text*)
```

In `bss.ldt:`

```
<#if memory.name=="SRAM_OC1">
*(.bss*)
</#if>
```

6 HW setup

For HW setup, refer to Figure 1.

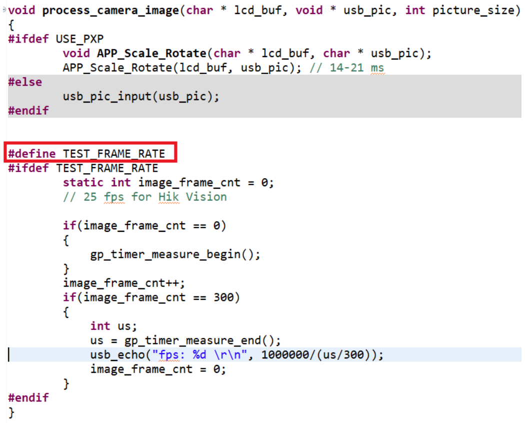
Power the board by a power adapter instead of a USB cable.

The below camera is tested in this AN:

1. HIK VISION E14
2. Logi C270
3. Newman NM-13-2K.

7 Performance test

To test the performance (frame rate), enable the `TEST_FRAME_RATE` in function `process_camera_image()` as shown below:



```
void process_camera_image(char * lcd_buf, void * usb_pic, int picture_size)
{
#ifdef USE_PXP
        void APP_Scale_Rotate(char * lcd_buf, char * usb_pic);
        APP_Scale_Rotate(lcd_buf, usb_pic); // 14-21 ms
#else
        usb_pic_input(usb_pic);
#endif


#define TEST_FRAME_RATE
#ifdef TEST_FRAME_RATE
        static int image_frame_cnt = 0;
        // 25 fps for Hik Vision

        if(image_frame_cnt == 0)
        {
            gp_timer_measure_begin();
        }
        image_frame_cnt++;
        if(image_frame_cnt == 300)
        {
            int us;
            us = gp_timer_measure_end();
            usb_echo("fps: %d \r\n", 1000000/(us/300));
            image_frame_cnt = 0;
        }
#endif
}
```

Figure 16. Performance test

From the console, the frame rate is 25 fps, it matches the 25 fps declared in the frame descriptor from the USB camera, see Section 3.1.1.

8 Revision history

Table 7. Revision history

Document ID	Release date	Description
AN14588 v.1.0	12 August 2025	Initial version

9 Note about the source code in the document

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

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.

Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

Contents

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