

Using FlexIO for parallel Camera Interface

Hiroshi Hiraga, Katsuhiro Atsumi, Rastislav Pavlanin

1. Introduction

1.1. Abstract

FlexIO interface is a highly configurable module which provides a wide range of communication protocols. FlexIO provides features as an emulation of serial communication protocols to increase functionality and scalability. The microcontrollers supporting FlexIO include:

- Kinetis L series
 - KL33
 - KL43
 - KL17
 - KL27
 - KL28
- K series
 - K8x

KL28 and K8x additionally support parallel communication, programmable digital logic functions, and a state machine mode. This application note uses an example code on K80 to explain how to configure FlexIO to emulate a parallel camera interface.

Contents

1. Introduction.....	1
1.1. Abstract.....	1
1.2. Objective.....	2
2. Functional Description.....	2
2.1. Hardware requirements.....	2
2.2. TWR-K80F150M board configuration.....	2
2.3. Software requirements.....	4
2.4. Software configurations.....	4
2.5. System workflow.....	5
2.6. FlexIO configurations.....	6
3. Considering Low Power Operation.....	9
3.1. Energy saving smart peripheral.....	9
3.2. Configuration for low power mode.....	9
4. Conclusion.....	10
5. References.....	10
6. Glossary.....	10
7. Revision History.....	11



NOTE

The parallel mode is not available on FlexIO module V1.0 (e.g., KL43Z256 MCU). It is only available on FlexIO modules V1.1 or higher (e.g., K80FN256 MCU). The FlexIO module version is defined in the FLEXIOx_VERID register using the MAJOR and MINOR fields.

1.2. Objective

This application note describes:

- FlexIO camera interface;
- how FlexIO emulates a parallel camera interface to capture 30 fps QVGA image from camera module;
- how to configure the FlexIO with Kinetis SDK V1.3;
- how to configure the lower power operation, as FlexIO is designed to keep alive in low power mode as energy-saving peripheral.

2. Functional Description

2.1. Hardware requirements

FlexIO camera demo requires the following hardware environments:

- TWR-K80F150M
- TWR-LCD
- TWR-PROTO built with OV7670

2.2. TWR-K80F150M board configuration

Figure 1 uses TWR-K80F150M as a demo example and *Table 1* describes the pin configuration. TWR-K80F150M jumper option is set by default.

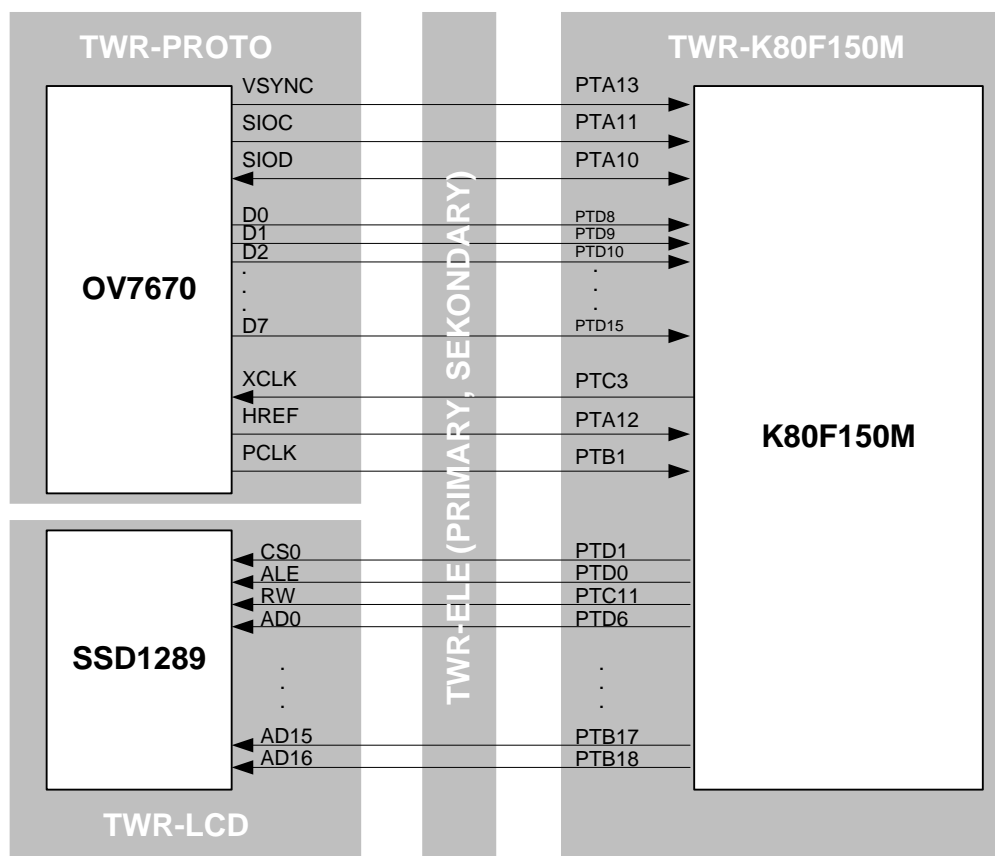


Figure 1. TWR-K80F150M connection diagram

Table 1. TWR-K80F150M pin configuration

		MCU pin	MCU function	Elevator	CAMERA OV7670	Comments
FlexIO HW configuration	FlexIO data pins	PTD8	FXIO0_D24	A7	D0	PULL-UP -> R189
		PTD9	FXIO0_D25	A8	D1	PULL-UP -> R190
		PTD10	FXIO0_D26	C38	D2	R295 populated, R114 removed
		PTD11	FXIO0_D27	C37	D3	R299 populated, R49 removed
		PTD12	FXIO0_D28	D40	D4	R113 populated, R52 removed
		PTD13	FXIO0_D29	D39	D5	R112 populated, R56 removed
		PTD14	FXIO0_D30	D38	D6	R111 populated, R81 removed
		PTD15	FXIO0_D31	D37	D7	R110 populated, R101 removed
	FlexIO timer 0 trigger	PTA12	FXIO_D18	B62	HREF	
	FlexIO timer 0 input pin	PTB1	FXIO0_D1	A37	PCLK	
I2C2 HW configuration	PTA10	SDA	C8	SIOD		
	PTA11	SCL	C7	SIOC		
GPIO HW configuration	PTA13	FXIO_D19	B61	VSYNC		
CLKOUT HW configuration	PTC3	CLKOUT	A64	XCLK	R187 removed, R143 populated	

Table 1. TWR-K80F150M pin configuration

		MCU pin	MCU function	Elevator	CAMERA OV7670	Comments
FlexBUS HW configuration	FlexBUS data/address pins	PTB17	FB_AD16	B67		
		PTB18	FB_AD15	B66		
		PTC0	FB_AD14	A66		
		PTC1	FB_AD13	A67		
		PTC2	FB_AD12	A68		
		PTC4	FB_AD11	A69		
		PTC5	FB_AD10	A70		
		PTC6	FB_AD9	A71		
		PTC7	FB_AD8	A72		R30 removed
		PTC8	FB_AD7	A73		
		PTC9	FB_AD6	A74		
		PTC10	FB_AD5	A75		R203 removed
		PTD2	FB_AD4	A76		
		PTD3	FB_AD3	A77		
		PTD4	FB_AD2	A78		
		PTD5	FB_AD1	A79		
			FlexBUS R/W pin	PTD6	FB_AD0	A80
	FlexBUS ALE pin	PTC11	FB_RW	B71		R195 removed
	FlexBUS chip select pin	PTD0	FB_ALE	B63		
	FlexBUS R/W pin	PTD1	FB_CS0	B64		

2.3. Software requirements

- Kinetis SDK v1.3.0

Project folder provided in zip: *app_an_k80_flexio_camera*

This demo's project folder, *app_an_k80_flexio_camera*, requires to be copied and placed in C:\Freescale\KSDK_1.3.0\user_projects. (*)

*: The folder is defined and generated by KSDK Project Generator tool.

(To download the KSDK Project Generator tool, please visit www.nxp.com/ksdk.)

2.4. Software configurations

Table 2 shows the software configurations of the FlexIO camera demo.

Table 2. System configuration

Clocks	MCGCLKOUT	150MHz in HS run mode (120MHz in normal run mode for ability to STOP mode entering)
	CORE/SYSTEM	150MHz (120MHz when STOP mode selected)
	Peripheral Bus	75MHz (60MHz when STOP mode selected)
	FlexBus	50MHz (40MHz when STOP mode selected)
	FlexIO	150MHz (120MHz when STOP mode selected)
	CLKOUT (for CLKIN on Camera module)	12MHz
Camera	Camera module	OV7670
	Interface	8bit CMOS parallel I/F
	Frame size	320 x 240 QVGA
	Color mode	16bits RGB565
	Control I/F	I2C 100kbps

Table 2. System configuration

	VSYNC	33 ms (30 fps)
FlexIO	Mode	Parallel receive mode
	Timer	1 timer
	Shifter	8 shifters concatenated
	Number of used pins	10 pins (8 data + 1 HREF + 1 PCLK)
eDMA	Ch 16 (not used in stop mode)	Transfer from SRAM camera buffer to FlexBus(LCD 16bit width). 2 Byte transfer Minor loop = 16 Major loop = 4800 2byte x 16 x 4800 = 150kB
	Ch 17	Transfer from FlexIO shift buffers to SRAM camera buffer. Minor loop = 32Byte burst transfer Major loop = 4800 32Byte x 4800 = 150kB
	Ch 2	Restore initial address of LCD at VSYNC edge.
	Ch 3	To restore eDMA ch17 iteration count (CITER) to synchronize with frame at VSYNC edge.

2.5. System workflow

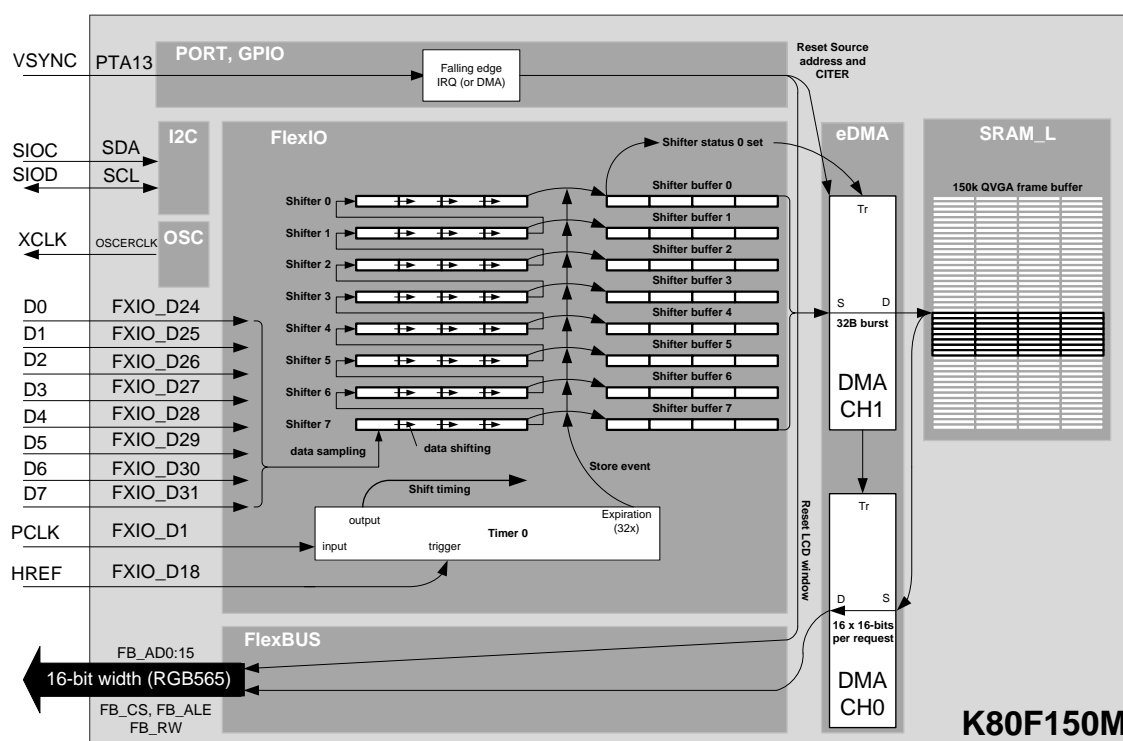


Figure 2. System workflow

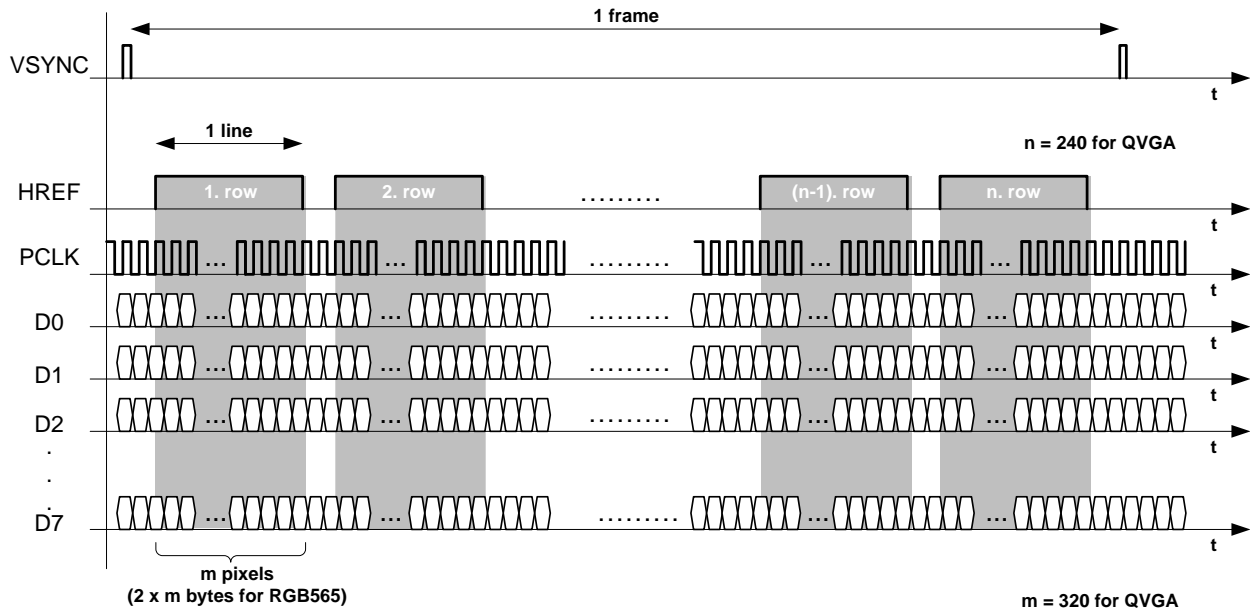


Figure 3. CMOS camera signal waveform

Figure 3 shows the system workflow and signal wave form of the FlexIO camera demo. The window size and position for LCD display is configured with FlexBus at VSYNC edge. To make sure that a frame is synchronized, eDMA Ch2 and Ch3 channels restore a loop counter of iteration count (CITER) and DMA source and destination address for camera buffer to initial address at VSYNC edge. This restoring source and destination address for buffer is necessary just in case that asynchronous interrupt occurs.

FlexIO is configured as a receiving mode and parallel shift mode for 8-bit CMOS sensor camera I/F.

Arrays of 32-bit shifters in FlexIO are concatenated one after the other for buffering samples of captured image. It can configure up to 32-byte buffer for receiving. Figure 4 shows the details of shift register configuration. During HREF signal high, FlexIO shifters shift data in on pixel clock edge. DMA request generates when 32-byte pixel data buffered in shifters has been loaded into shift buffers (SHIFTBUF register is full). eDMA transfers the pixel data from SHIFTBUF into a camera frame buffer in SRAM. The other channel of eDMA transfers the data in the camera buffer to LCD controller using FlexBus interface to display on TFT LCD.

2.6. FlexIO configurations

2.6.1. FlexIO pin configuration

Table 1 shows the FlexIO data pins used in this demo. GPIO pins are multifunctional and multiplexed. The pins need to be configured where external pins are used as FlexIO.

Kinetis SDK provides API programming interface to configure signal multiplexing. Also, clock gate to those PORT pins and FlexIO module need to be enabled and called in hardware_init() function.

```
//hardware_init.c
/* enable clock for PORTs */
CLOCK_SYS_EnablePortClock(PORTA_IDX); //used as FlexIO
```

```

CLOCK_SYS_EnablePortClock(PORTB_IDX); //used as FlexIO
CLOCK_SYS_EnablePortClock(PORTC_IDX);
CLOCK_SYS_EnablePortClock(PORTD_IDX); //used as FlexIO
CLOCK_SYS_EnablePortClock(PORTE_IDX);

/* Enable clock gate to FlexIO module*/
CLOCK_HAL_SetFlexioSrc(SIM, 0, kClockFlexioSrcPllFllSelDiv);
CLOCK_HAL_SetPllfllSel(SIM, kClockPllFllSelPll);
CLOCK_HAL_SetPllFllDiv(SIM, 0, 0);
SIM_HAL_EnableClock(SIM, kSimClockGateFlexio0);

//hardware_init.c
void configure_app_flexio_pins(void)
{
    // Configure FlexIO pins
    PORT_HAL_SetMuxMode(PORTB,1u,kPortMuxAlt7); ///< FXIO_D1 -> OV7670_PCLK
    PORT_HAL_SetMuxMode(PORTA,12u,kPortMuxAlt5); ///< FXIO_D18 -> OV7670_HREF

    PORT_HAL_SetMuxMode(PORTD,8u,kPortMuxAlt7); ///< FXIO_D24 -> OV7670_D0
    PORT_HAL_SetMuxMode(PORTD,9u,kPortMuxAlt7); ///< FXIO_D25 -> OV7670_D1
    PORT_HAL_SetMuxMode(PORTD,10u,kPortMuxAlt7); ///< FXIO_D26 -> OV7670_D2
    PORT_HAL_SetMuxMode(PORTD,11u,kPortMuxAlt7); ///< FXIO_D27 -> OV7670_D3
    PORT_HAL_SetMuxMode(PORTD,12u,kPortMuxAlt7); ///< FXIO_D28 -> OV7670_D4
    PORT_HAL_SetMuxMode(PORTD,13u,kPortMuxAlt7); ///< FXIO_D29 -> OV7670_D5
    PORT_HAL_SetMuxMode(PORTD,14u,kPortMuxAlt7); ///< FXIO_D30 -> OV7670_D6
    PORT_HAL_SetMuxMode(PORTD,15u,kPortMuxAlt7); ///< FXIO_D31 -> OV7670_D7
}

```

2.6.2. FlexIO shifter configuration

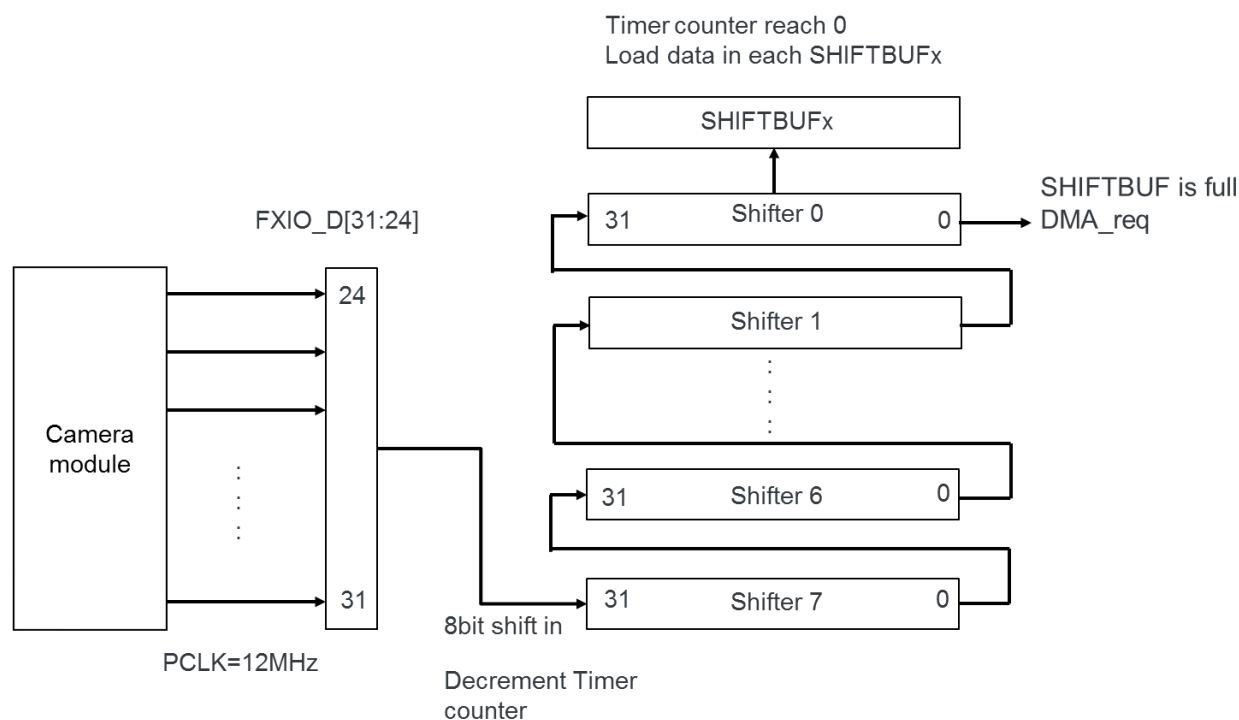


Figure 4. FlexIO shifter configuration

FlexIO has a capability to output and input data in serial or parallel. When the parallel receive mode is configured, only Shifter 3 and 7 can support parallel capturing from FlexIO pins. *Figure 3* shows the FlexIO shifter configuration. FlexIO has 8 shifters with 32-bit length. The Shifter can be concatenated each other to buffer a large size of data up to 32 byte. The shifter status flag (SHIFTSTAT[SSF]) is set when the shift buffer (SHIFTBUFn) has been loaded with data from Shifter according to Shifter Mode (SHIFTCTL[SMOD]). SSF can generate DMA request. In this demo, only the last shifter status flag (Shifter 0) initiates DMA transfer. FlexIO Timer is configured as 16-bit timer with FXIO_D1 as clock input, and it is enabled on rising edge on HREF (FXIO_D18) and disabled on falling edge on HREF. The timer decrements on both edges of the pixel clock.

This example uses 8 chained Shifters for buffering (32 byte = 32bits x 8) to capture 16 pixels of image data (RGB565: 2byte per one pixel), so that 32 shifts are required to shift in to the full of the shifters. The timer compare value (TIMCMP0[15:0]) is then set to *(the number of bits) x 2 - 1 = 32 x 2 - 1 = 63*. When the timer value reaches zero, the pixel data in the shifter is loaded in SHIFTBUF and the SSF flag is set.

2.6.2.1. Initializing FlexIO module

Kinetis SDK provides API programming interface to initialize FlexIO module and set user configuration as an application need. To initialize FlexIO module, FLEXIO_DRV_Init() function is called and the user configuration structure is shown as below.

```
//main.c FlexIO driver init function
FLEXIO_DRV_Init(0, &flexioModuleConfig);

//main.c FlexIO user config.
static flexio_user_config_t flexioModuleConfig = FLEXIO_MODULE_CONFIG;

//main.h
#define FLEXIO_MODULE_CONFIG
{
    .useInt = true,
    .onDozeEnable = false,
    .onDebugEnable = false,
    .fastAccessEnable = true
}
```

2.6.3. FlexIO shifter, timer and DMA configuration

Kinetis SDK provides FlexIO camera driver library that configures the FlexIO working as a camera parallel interface, the indicated eDMA channel moving data from FlexIO interface to user defined memory. With this function, an image of camera sensor can be mapped into user-defined memory. Then the user can read the mapped memory any time during the application.

Functions and FlexIO camera handler structure are shown as below.

```
//main.c FlexIO camera driver functions
FLEXIO_Camera_DRV_InitEdmaRx(    &flexio_camera_handler,
                                &flexio_camera_user_config,
                                &camera_edma_user_config);
FLEXIO_Camera_DRV_SetBufferTriggerForExtEdma(&flexio_camera_handler, 0x0, true);
FLEXIO_Camera_DRV_StartEdmaRx(&flexio_camera_handler);

//main.c FlexIO camera handler structure for FlexIO camera driver
```



```

static flexio_camera_edma_handler_t flexio_camera_handler = FLEXIO_CAMERA_HANDLER;

//main.h
#define FLEXIO_CAMERA_HANDLER
{
    .flexioCameraHwConfig =
    {
        .flexioBase = FLEXIO0,
        .datPinStartIdx = 24,
        .pclkPinIdx = 1,
        .hrefPinIdx = 18,
        .shifterStartIdx = 0,
        .shifterCount = 8,
        .timerIdx = 0,
    },
    .rxEdmaChnState =
    {
        .channel = kEDMAChannel17,
        .callback = NULL,
        .parameter = NULL,
        .status = kEDMAChnNormal,
    },
    .userEdmaChn = kEDMAChannel16,
    .userBufAddr = (uint32_t)&u16CameraFrameBuffer,
    .userBufLenByte = sizeof(u16CameraFrameBuffer),
}

```

3. Considering Low Power Operation

3.1. Energy saving smart peripheral

For the low power application, FlexIO is designed to be able to be active as energy-saving peripheral. As eDMA supports asynchronous operation that can run in the low power mode as well, FlexIO can continue capturing image data from camera module during the low power mode. The clock source should remain active in such low power mode and is selected. When DMA request happens during low power mode, the system initiates the Normal Exit to transfer data but the core is in the Sleep mode. Once DMA transfer has completed, the power mode will automatically go back to the original low power state including non-CPU bus masters and slaves to enter the Stop mode. PSTOP1 mode can be used and achieved lower power consumption for capturing camera image. As STOP/VLPS takes recovery time to wake up, it may require lower pixel clock rate to capture image data in time.

Note that FlexBus interface is not available in low power mode for displaying image on LCD. Hence when the STOP mode is selected (pressing the SW1 button on TWR-K80 board the first time after power up in example), the last camera picture is pushed to be displayed on LCD by software in SW1 button interrupt service routine (no DMA CH16 used in this case).

3.2. Configuration for low power mode

[Table 3](#) shows how the low power mode is configured in this demo.

In this demo, the default mode is Normal Run.

For the low power operation mode, DEMO_OPERATION_MODE needs to define STOP_DMA_OPERATION_MODE.

```
//main.h
#define STOP_DMA_OPERATION_MODE      0x0
#define NORMAL_RUN_DMA_OPERATION_MODE 0x1
#define NORMAL_RUN_IRQ_OPERATION_MODE 0x2
```

Table 3. Lower power mode configuration

FlexIO clock source	MCGPLLCLK
MCG_C5[PLLSTEN]	1 MCGPLLCLK is enabled in Normal Stop mode
Low power mode	PSTOP1

4. Conclusion

This FlexIO camera demo is built based on Kinetis SDK v1.3.0 for NXP Kinetis K8x MCU families. This demo uses FlexIO emulate 8-bit CMOS camera interface.

5. References

- *K80 Sub-Family Reference Manual* (document K80P121M150SF5RM)
- *K80 Sub-Family Datasheet* (document K80P121M150SF5)
- *K80 MCU Tower System Module* (document TWR-K80F150M)
- *Graphical LCD Tower System Module* (document TWR-LCD)
- *OmniVision VGA CMOS Camera Module* (document OV7670)
- *Software Development Kit for Kinetis MCU* (document Kinetis SDK v1.3)

6. Glossary

- API Application Programming Interface
- SDK Software Development Kit
- DMA Direct Memory Access
- eDMA Enhanced Direct Memory Access
- GPIO General Purpose Input and Output
- HREF Horizontal Reference Signal
- PSTOP Partial-STOP mode
- RGB Read Green Blue
- SHIFTBUF Shift buffers
- Shifter Shift registers
- VSYNC Vertical Synchronized Signal
- VLPS Very-Low-Power-Stop

7. Revision History

Table 4. Revision history

Revision number	Date	Substantive changes
0	04/2016	Initial release



How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

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

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

NXP, the NXP logo, and the Energy Efficient Solutions logo, are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners

© 2016 NXP B.V.

Document Number: AN5275
Rev. 0
04/2016

