

Hands-On Workshop: Implementing Custom Logic and Serial Interfaces Using the **Kinetis MCU FlexIO Feature**

FTF-DES-F1298

Rastislav Pavlanin | Application Engineer

J U N E . 2 0 1 5



External Use

Freescale, the Freescale logo, AIRMac, C-S, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, MagniC, mobileGFX, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Converge, Connive, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, VybeR and XtremeR are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airtast, BeepIt, BeeStack, CoreNet, FlexIO, Layerscape, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink and UMEMS are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2015 Freescale Semiconductor, Inc.



Agenda

- Basic Features
- Differentiation Between Available FlexIO Modules
- FlexIO
 - Module vs. Selected Serial Interfaces
 - Components
 - Modes of Operation
 - UART Emulation Use Case Based on KSDK 1.2
- CMOS VGA Camera OV7670 FlexIO Use Case Demonstration

Basic Features – In General

- Flexible functionality supported by highly configurable module
- Emulation of Serial / Parallel communication interfaces
 - UART, I2C master, SPI, I2S
 - Camera interface, Motorola 68K bus, Intel 8080 bus
- Complex timing functions generation using timers with flexible configuration
- Programmable digital logic
 - On-chip digital logic function
 - Interaction between internal and external modules
- Low software / CPU overhead compare to software based implementation
- Higher area / power efficiency in module integration

Basic Features – Functional

- Ability to work in low power mode (doze enable)
 - Full functionality available down to PSTOP2 mode
 - Asynchronous operation in PSTOP1, normal STOP, VLPS
 - Static in LLSx
 - Disabled in VLLSx
- Ability to continue operation in debug mode
- Ability for Fast access to module registers
 - NOTE: requires FlexIO clock to be twice the peripheral bus clock
- Ability to process reset to all module registers (except control register)
- Polled / interrupt / DMA operation support
- Selectable clock source
 - System clock
 - IRC48M or MCGPLLCLK or MCGFLLCLK with applicable fractional divider (NOTE: only on K80/K81)
 - OSCERCLK
 - MCGIRCLK
- Major / minor version + features / parameters identification

Differentiation Between Available FlexIO Modules

FlexIO module		KL17 / KL27 / KL33 / KL43 (48MHz)	KL27Z512 / KL28Z512 (72MHz)	K80FN256 / K81FN256
Versions	Major	1	1	1
	Minor	0	1	1
Features	Standard	x	x	x
	State mode	-	x	x
	Logic mode	-	x	x
	Parallel mode	-	x	x
Registers	FLEXIOx_PIN	-	x	x
	FLEXIOx_SHIFTSTATE	-	x	x
	FLEXIOx_SHIFTBUFNBSn	-	x	x
	FLEXIOx_SHIFTBUFHWSn			
Triggers		16	4 mux inputs with up to 64 sources	16
Pins		8	32	32
Timers		4	8	8
Shifters		4	8	8

NOTE 1: FLEXIOx_PIN register is tied to FlexIO version 1.1

NOTE 2: state / logic / parallel modes are not available on FlexIO 8-pin implementation (even it is version 1.1)

NOTE 3: nible / halfword swap registers are tied to parallel mode



FlexIO vs. Selected Peripheral Modules

- **FlexIO versus LPUART**

Peripheral module	Features						
	Flow control (CTS, RTS)	Oversampling	Parity	Stop bits	Receive data match	Break / Idle characters	Availability per module
FlexIO ver. 0	yes	no	no	1	fully or with low SW intervention	with low SW intervention	2 ¹
FlexIO ver. 1	yes	no	no	1	fully or with low SW intervention	with low SW intervention	4 ¹
LPUART	yes	yes	yes	up to 2	yes	yes	1

1 – no flow control

- **FlexIO versus SPI**

Peripheral module	Features						
	Master	Slave	Operational in STOP, VLPS	CPOL CPHA	MSB / LSB first	Max. baud rate	Availability per module
FlexIO ver. 0	yes	yes	yes	yes	yes	FlexIO CLK / 6	2 ²
FlexIO ver. 1	yes	yes	yes	yes	yes	FlexIO CLK / 6	4 ²
SPI	yes	yes	no	yes	yes	SPI CLK / 2 (4) ³	1

2 – in slave mode operation 2 timers left

3 – in slave mode operation



FlexIO vs. Selected Peripheral Modules

- **FlexIO versus I2C**

Peripheral module	Features						
	Master mode	Slave mode	Operational in STOP, VLPS	Clock stretching	Arbitration / Multimaster	Max. baud rate	Availability per module
FlexIO ver. 0	yes	no	yes	limited	no	FlexIO CLK / 6	2^4
FlexIO ver. 1	yes	no	yes	limited	no	FlexIO CLK / 6	4^4
I2C	yes	yes	no	yes	yes	400k	1

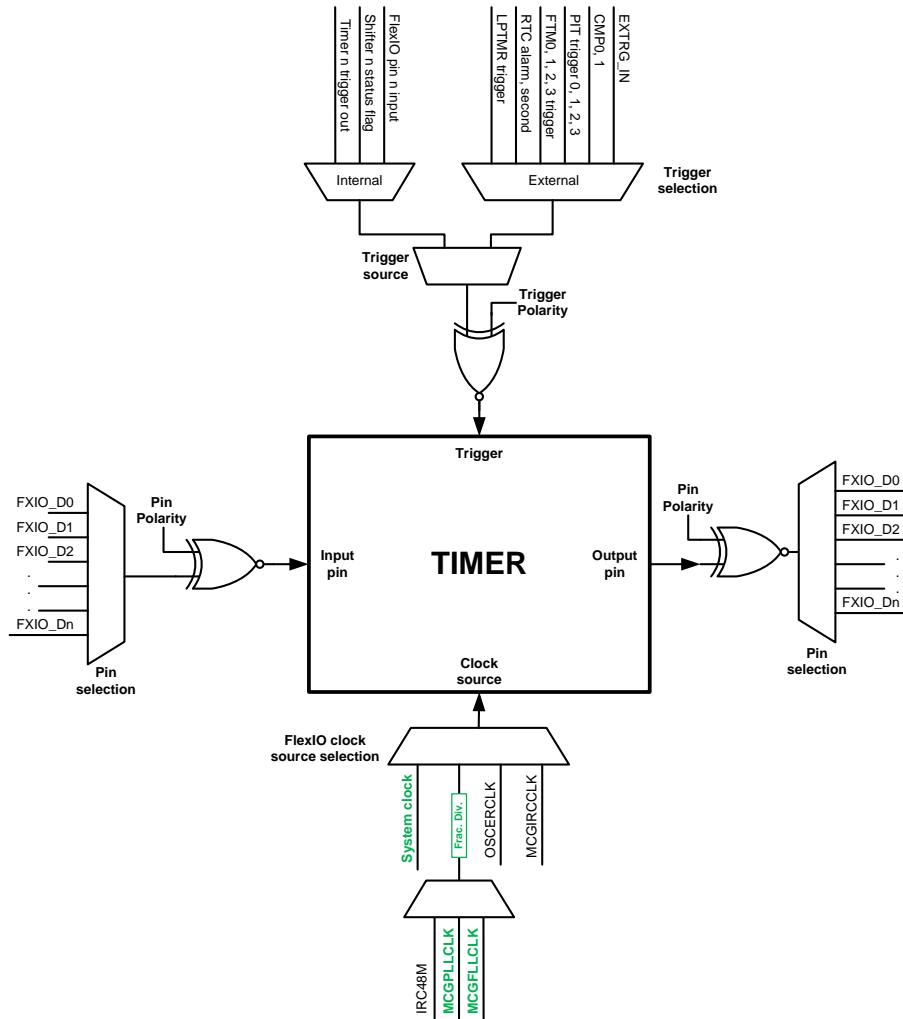
4 – only master mode

- **FlexIO versus I2S (SAI)**

Peripheral module	Features						
	Master mode	Slave mode	Operational in STOP, VLPS	MSB / LSB first	Bit clock polarity configuration	Max. baud rate	Availability per module
FlexIO ver. 0	yes	yes	yes	yes	yes	FlexIO CLK / 4 (6) ⁵	2
FlexIO ver. 1	yes	yes	yes	yes	yes	FlexIO CLK / 4 (6) ⁵	4
I2S (SAI)	yes	yes	yes	yes	yes	12.5M	1

5 – in slave mode operation

FlexIO Components – Timers



FlexIO Components – Timer Configuration

- Source of Timer decrement

Timer decrement on	FlexIO clock	Timer output	Shift clock is represented by
	Trigger input	Timer output	
		Trigger input	
	PIN input	PIN input	

- Timer reset

Timer is reset on	Never		
	PIN	equal to output	
		rising edge	
	Trigger	equal to output	
		rising edge	
		both edges	

- Initial value of timer output

Initial state of timer output	log. 1	when enabled	-
		on reset	
log. 0	when enabled	-	
		on reset	

FlexIO Components – Timer Configuration

- Timer Enabling

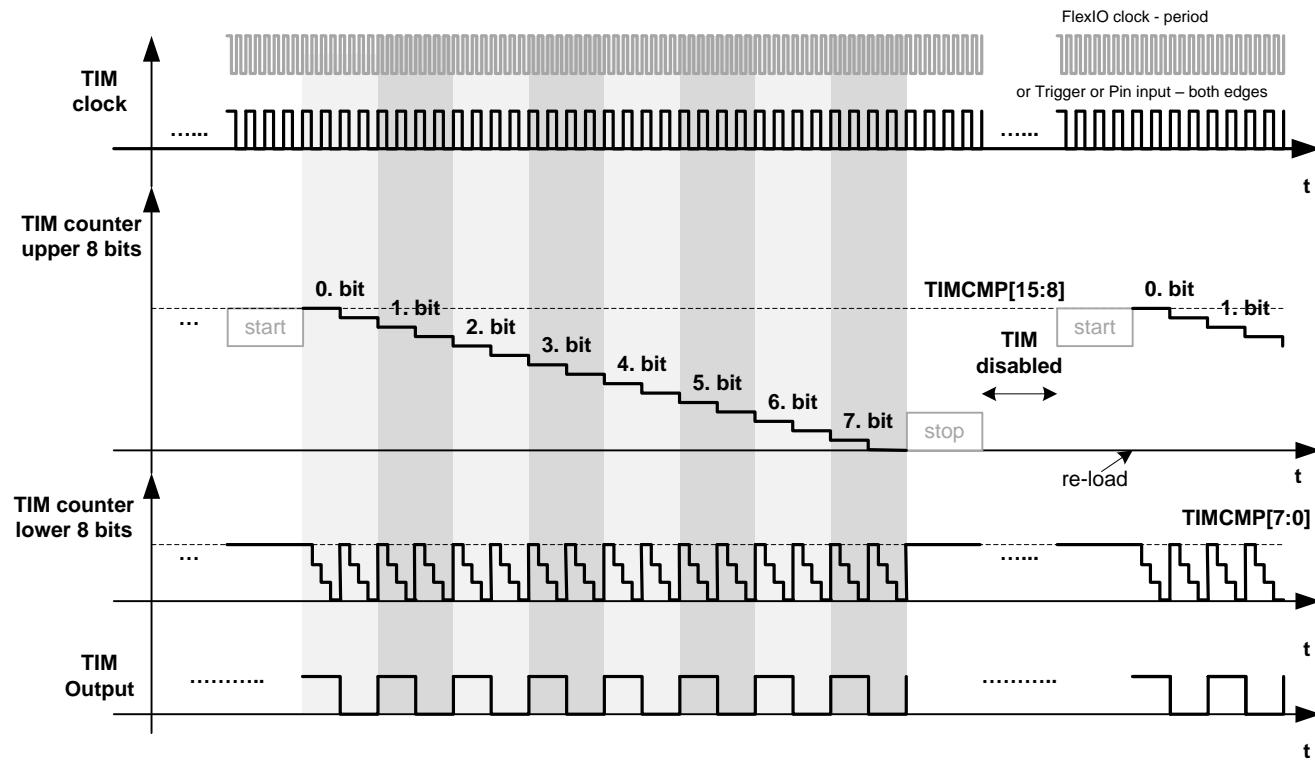
Timer is enabled on	Always		
	Timer N-1 enable		
	Trigger	high	-
		and PIN high	
		rising edge	
	both edges		
PIN	rising edge	-	
		and trigger high	

- Timer Disabling

Timer is disabled on	Never		
	Timer N-1 disable		
	Compare		-
	and trigger low		
	PIN	both edges	-
		both edges	trigger is high
	Trigger	falling edge	

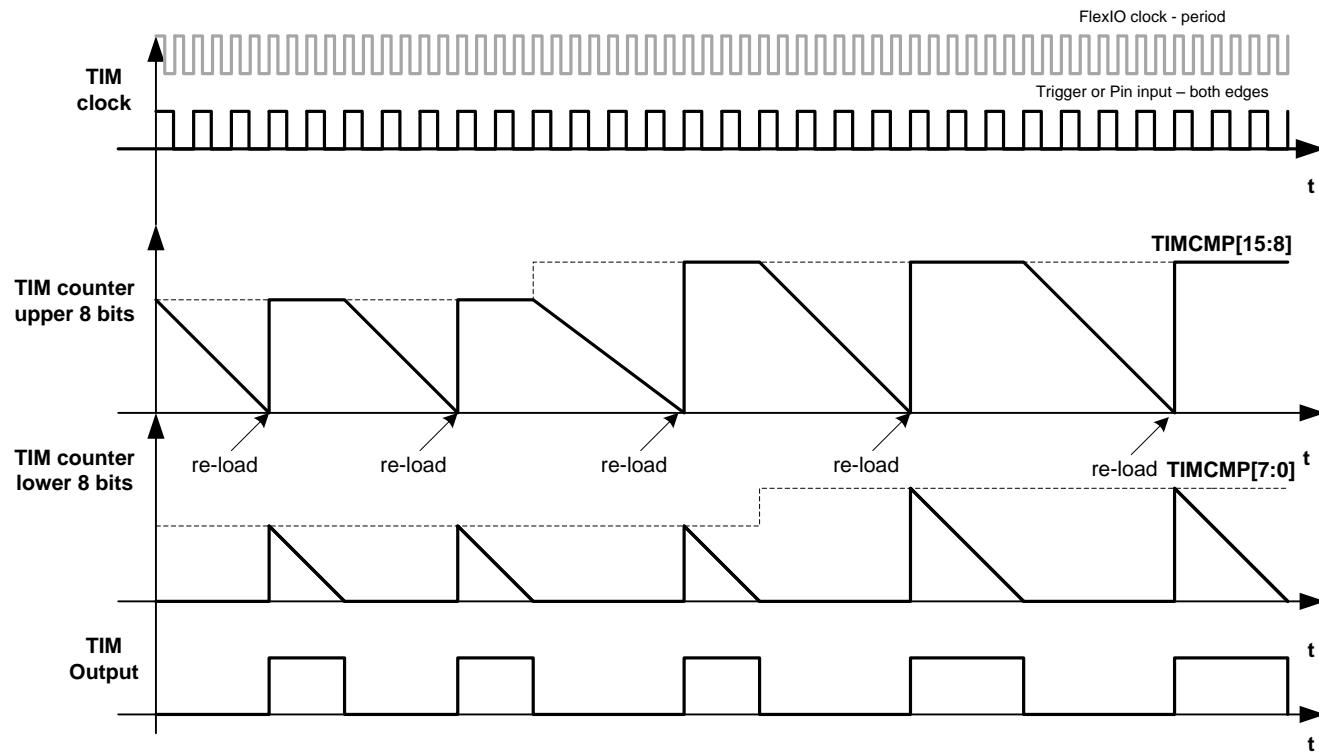
FlexIO Components – Timer Operation

- Dual 8-bit BAUD rate / bit count mode
 - Example of 8-bit data width and 8 baud rate divider



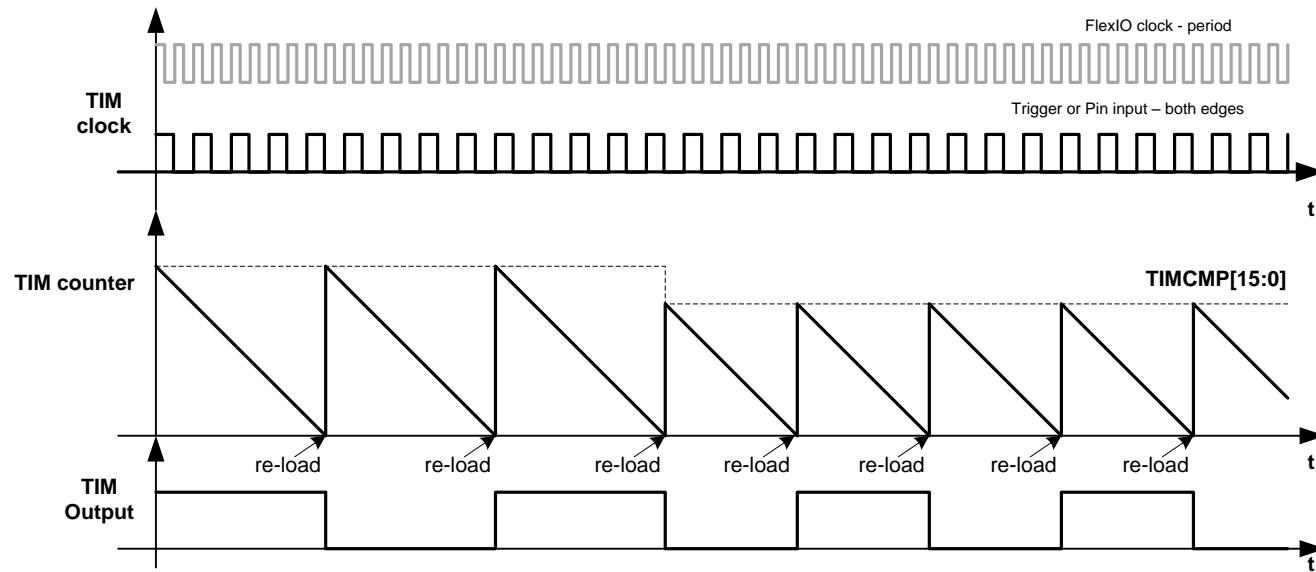
FlexIO Components – Timer Operation

- Dual 8-bit PWM mode

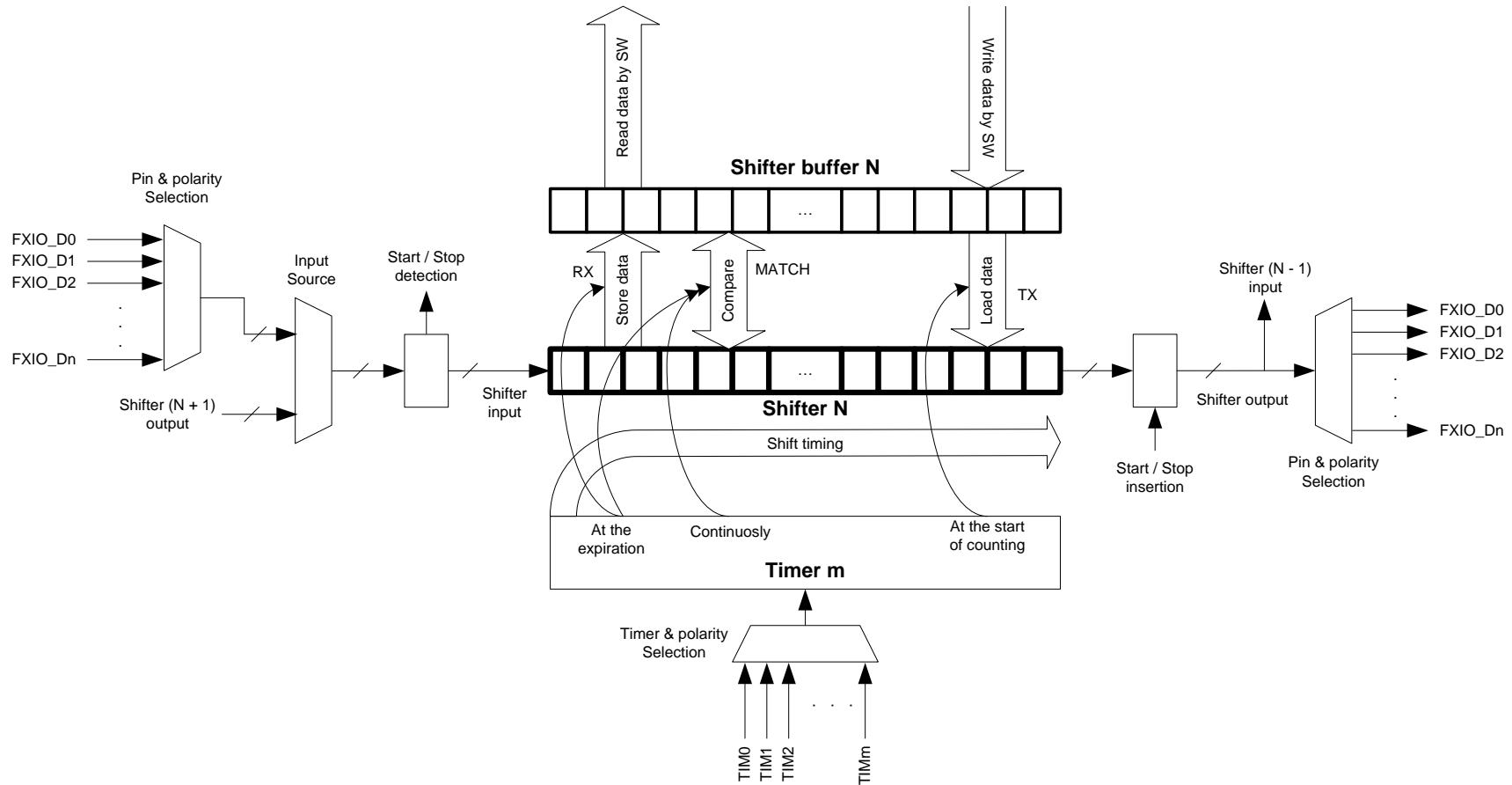


FlexIO Components – Timer Operation

- Single 16-bit counter mode

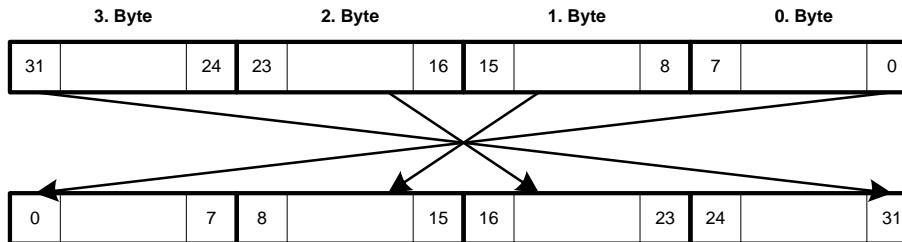


FlexIO Components – Shifters

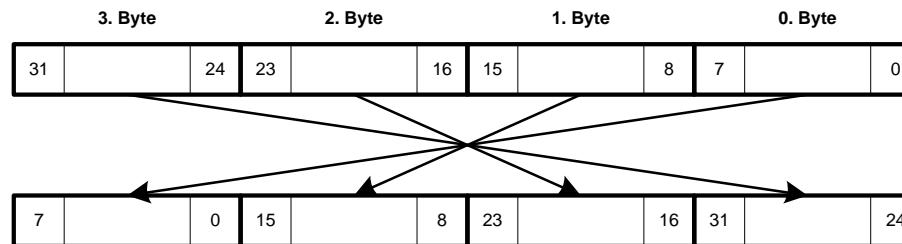


FlexIO Components – Shifters Buffer Aliases

- **SHIFTBUF + BIS → Bit swapped**

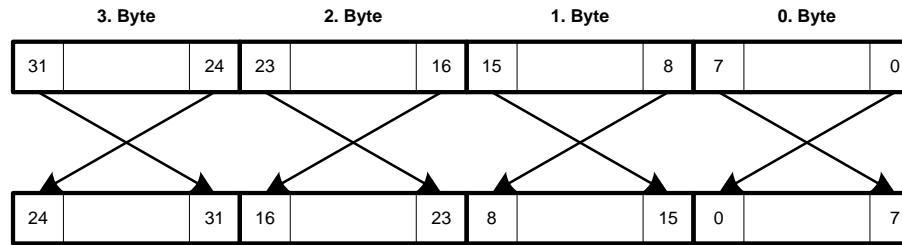


- **SHIFTBUF + BYS → Byte swapped**

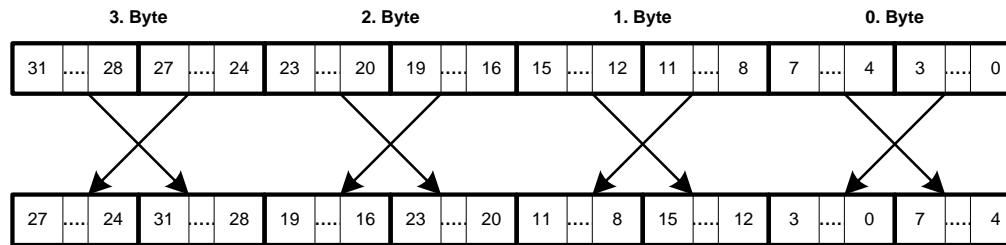


FlexIO Components – Shifters Buffer Aliases

- SHIFTBUF + BBS → Bit - byte swapped

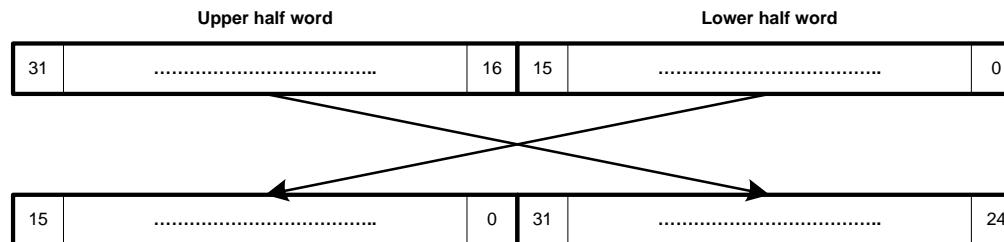


- SHIFTBUF + NBS → Nibble byte swapped

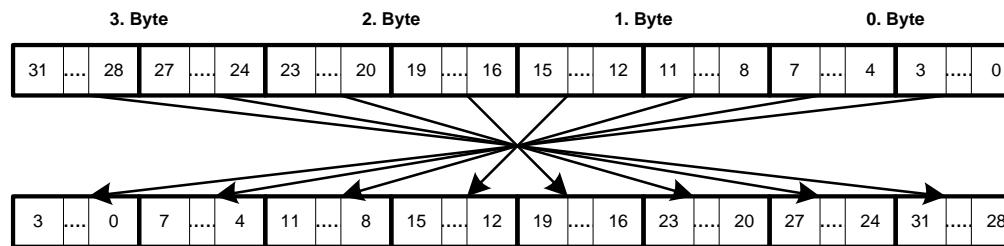


FlexIO Components – Shifters Buffer Aliases

- SHIFTBUF + HWS → Half word swapped

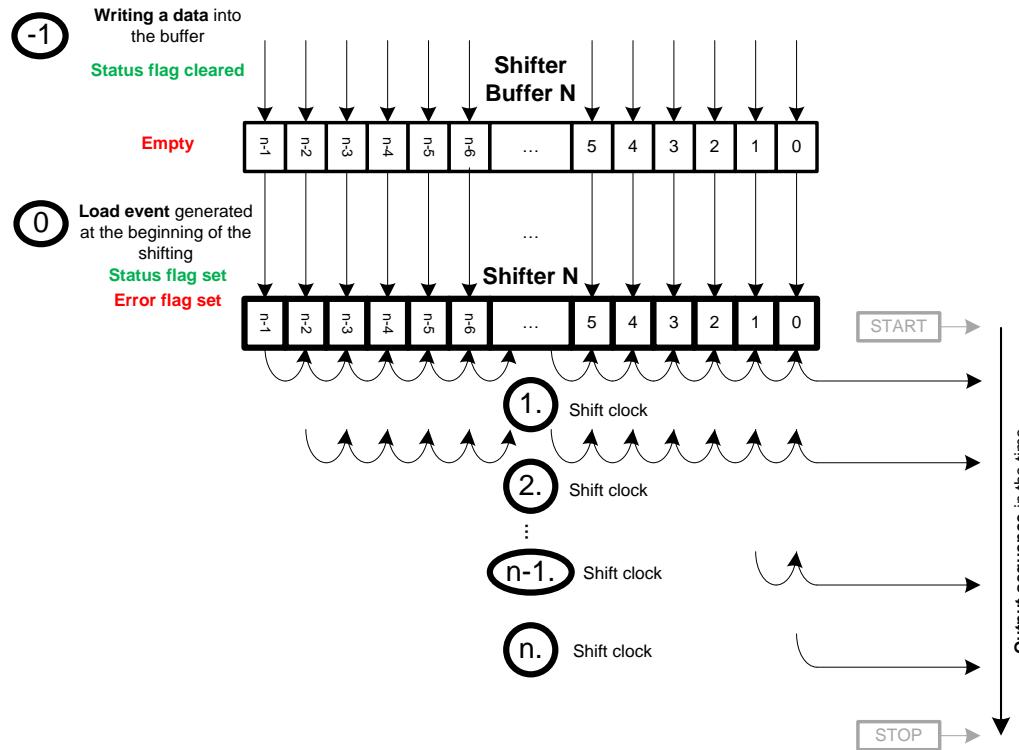


- SHIFTBUF + NIS → Nibble swapped



FlexIO Components – Shifters Modes of Operation

- Transmit mode - single



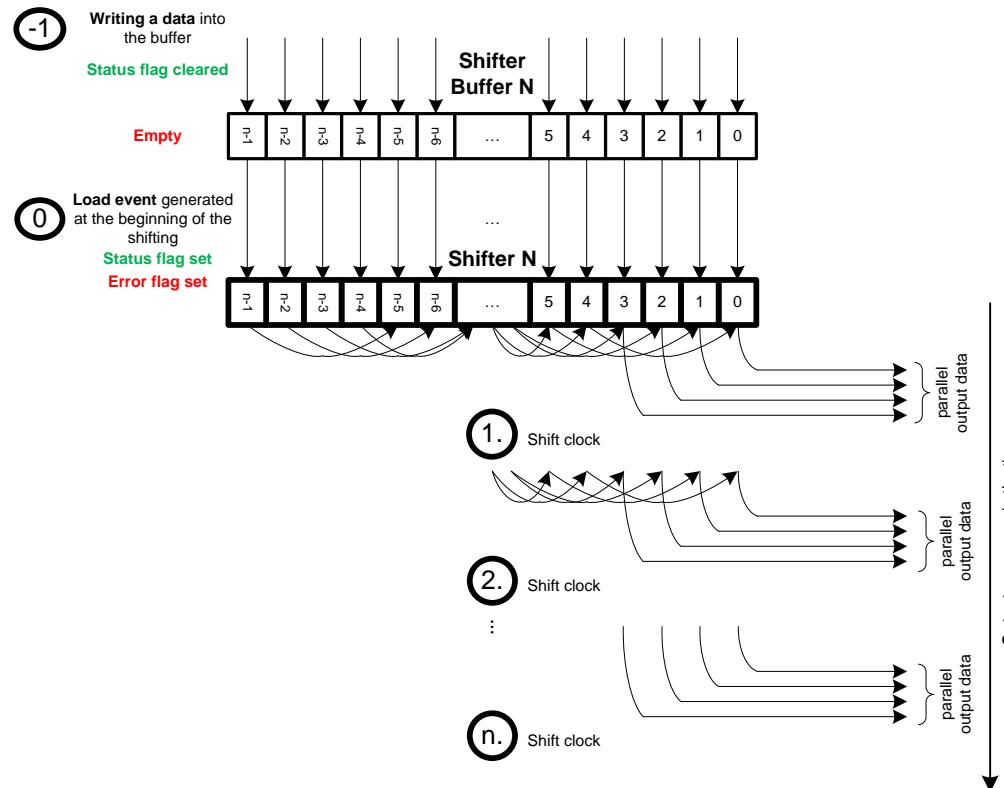
NOTE 1: Shifter0, Shifter4 only support parallel transmit to the output pins

NOTE 2: Shifters can be chained

NOTE 3: n – bit count (up to 32), N – component order

FlexIO Components – Shifters Modes of Operation

- Transmit mode – Parallel (available widths 4-bits, 8-bits, 16-bits, 32-bits)



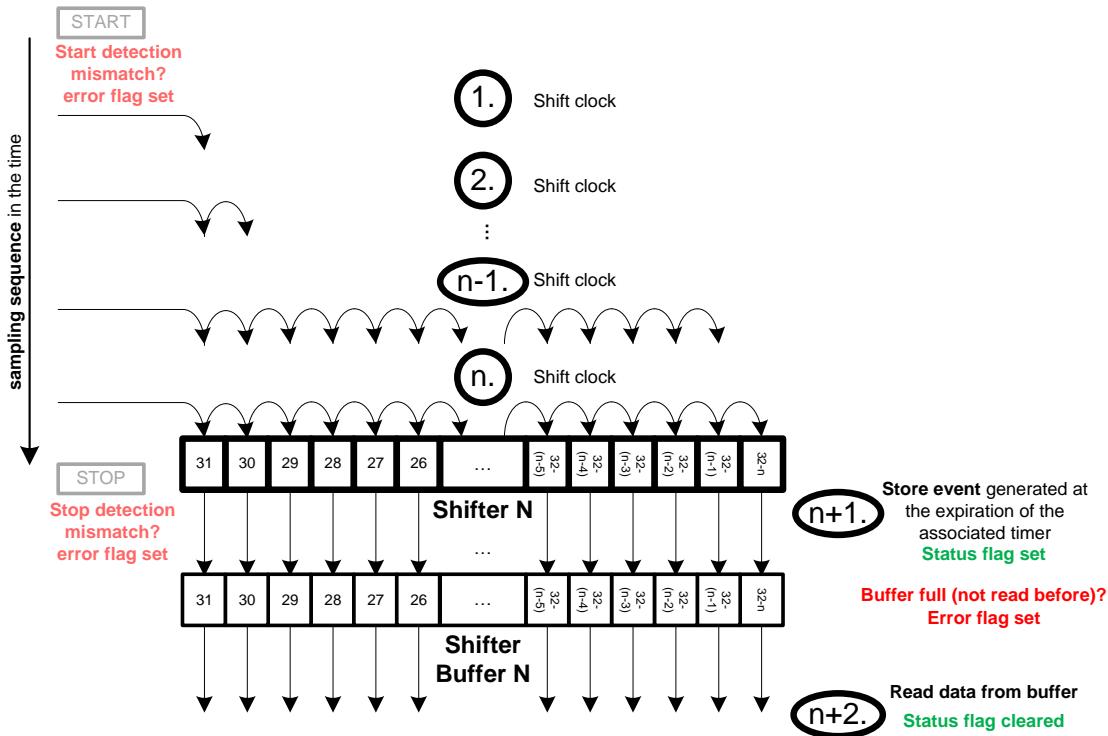
NOTE 1: Shifter0, Shifter4 only support parallel transmit to the output pins

NOTE 2: Shifters can be chained

NOTE 3: n – bit count (up to 32), N – component order

FlexIO Components – Shifters Modes of Operation

- Receive mode - single



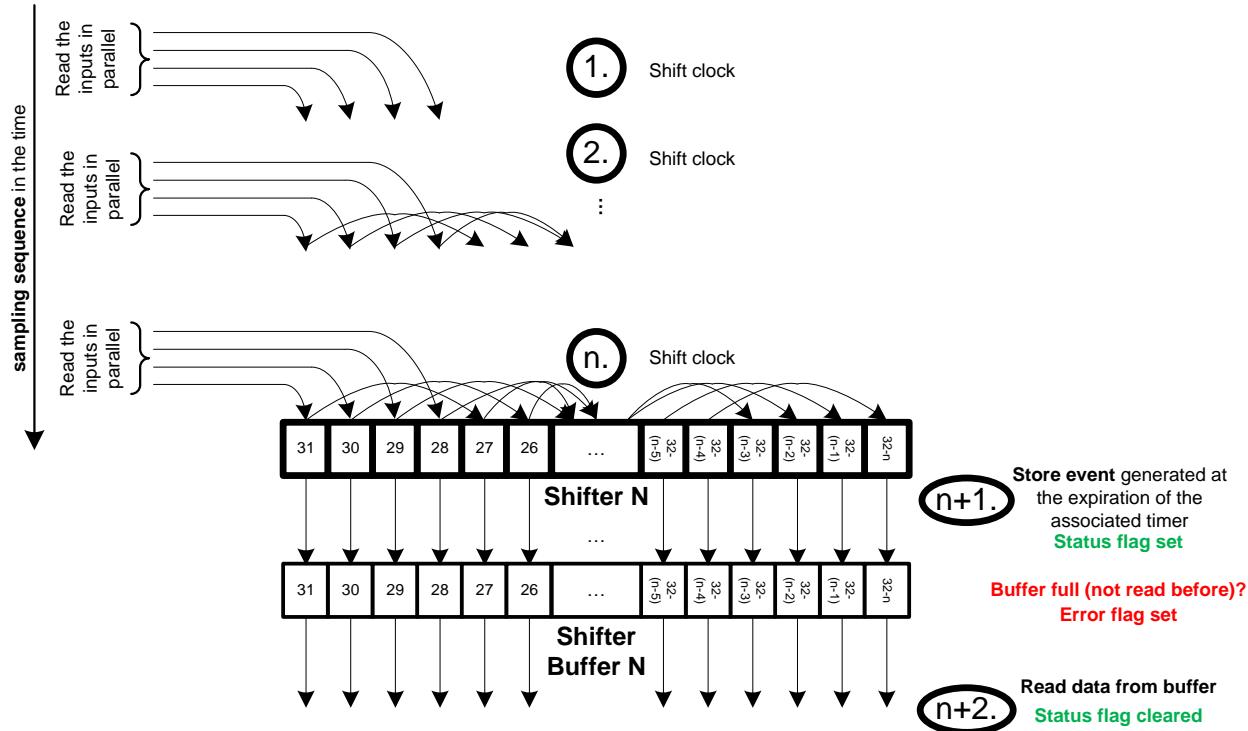
NOTE 1: Shifter3, Shifter7 only support parallel receive from the input pins

NOTE 2: Shifters can be chained

NOTE 3: n – bit count (up to 32), N – component order

FlexIO Components – Shifters Modes of Operation

- Receive mode - Parallel



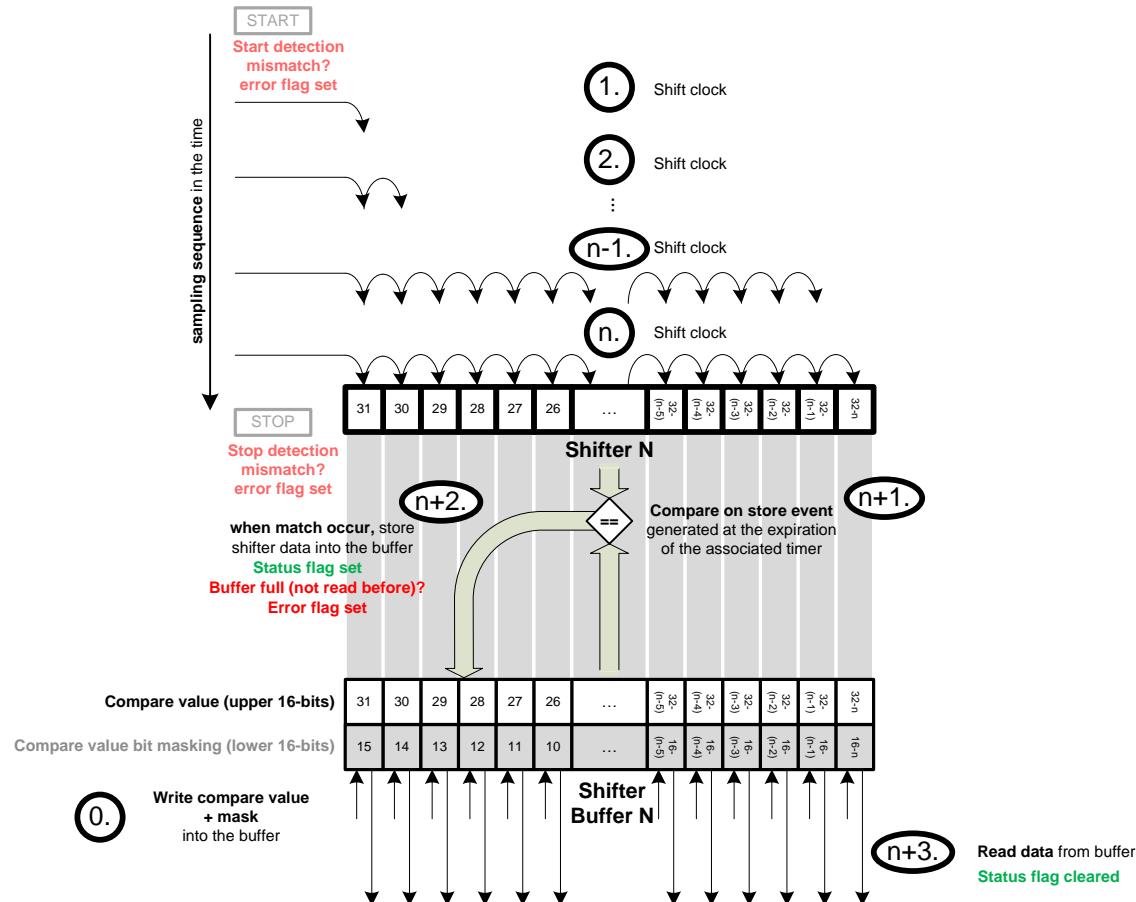
NOTE 1: Shifter3, Shifter7 only support parallel receive from the input pins

NOTE 2: Shifters can be chained

NOTE 3: n – bit count (up to 32), N – component order

FlexIO Components – Shifters Modes of Operation

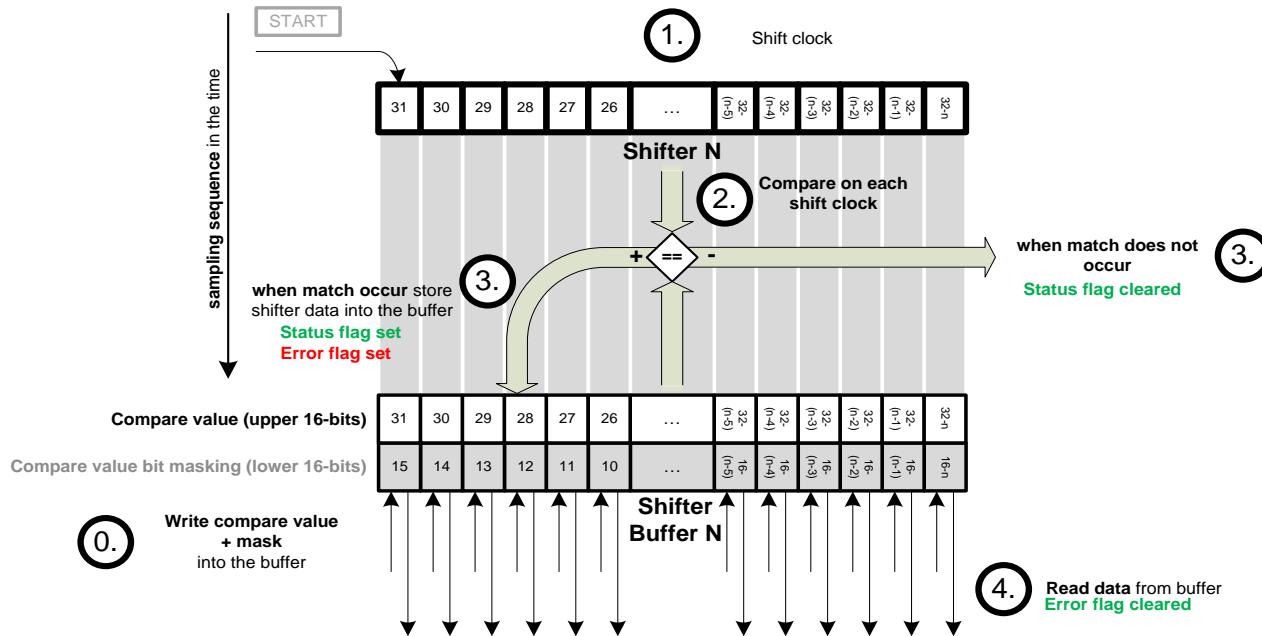
- Match store mode



NOTE 1: n – bit count (up to 16), N – component order

FlexIO Components – Shifters Modes of Operation

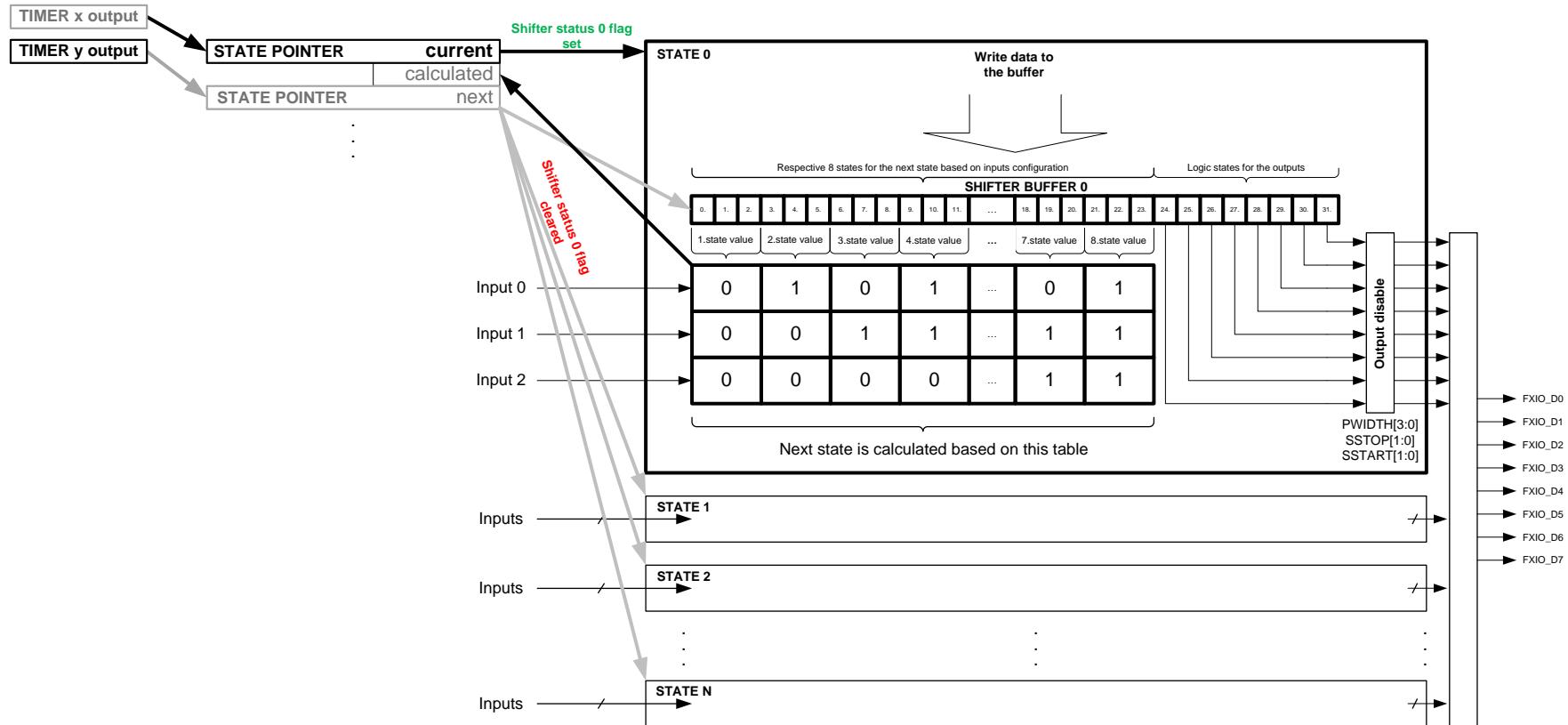
- Match continuous mode



NOTE 1: n – bit count (up to 16), N – component order

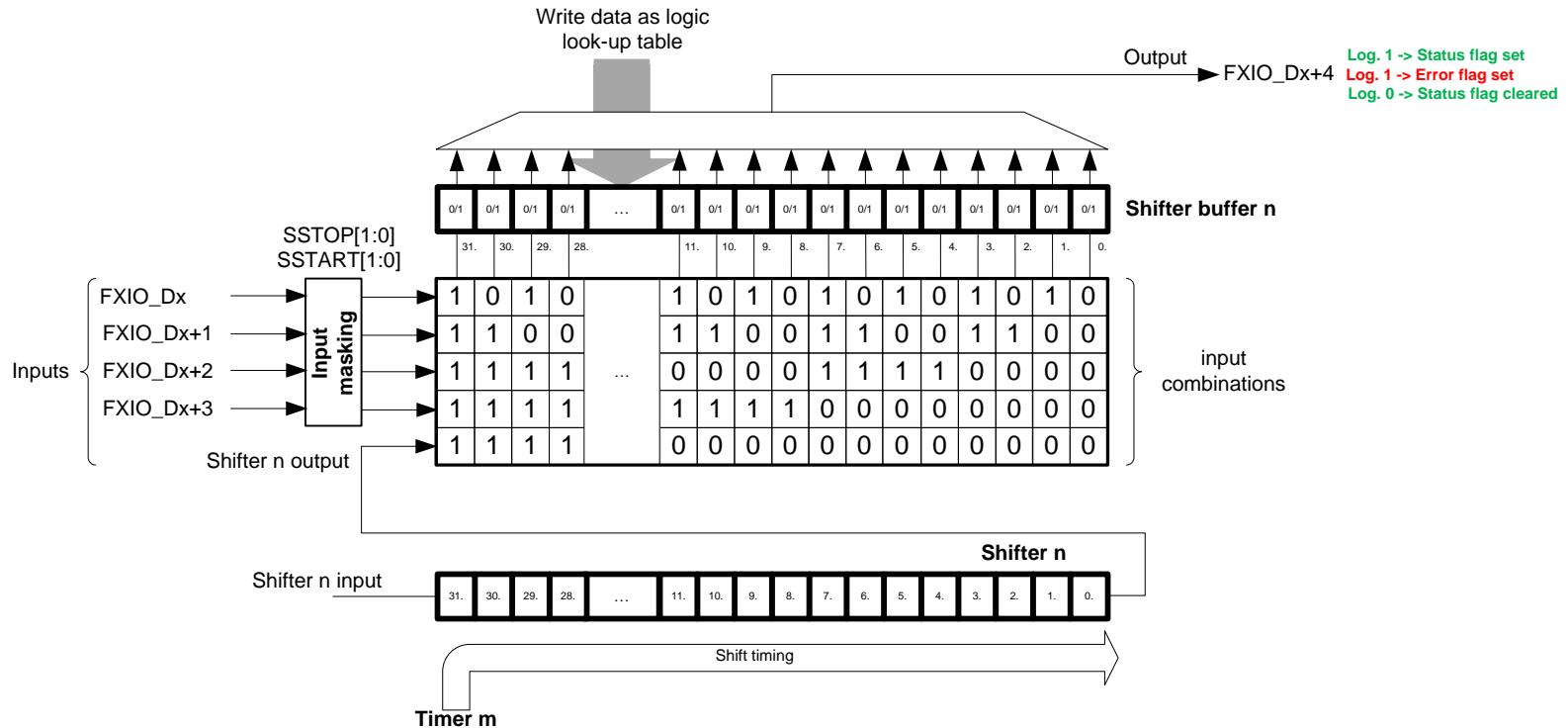
FlexIO Components – Shifters Modes of Operation

- State machine mode



FlexIO Components – Shifters Modes of Operation

- Logic mode



NOTE 1: Pin inputs and output are not selectable, are fixed to the appropriate shifter (for shifters n = 0-3 x = n, for shifters n = 4-7 x = n+4)

FlexIO Components – Shifters Modes of Operation

- Logic mode
 - Logic look-up table example:

Logic function:

$$Y = A \cdot \bar{B} + \bar{C} \cdot D + E$$

Inputs		Combinations																							
FXIO_Dx	A	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
FXIO_Dx+1	B	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
FXIO_Dx+2	C	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0	0	1	1
FXIO_Dx+3	D	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
Shifter n output	E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Output		Shifter Buffer n value (Look-up table values)																							
FXIO_Dx+4	BIN	Y	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	0
		HEX	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	2	E	2	2	2	2	2	0

UART FlexIO Use Case – Requirements

- Software Requirements
 - IAR or KEIL (pre-installed)
 - Freescale KSDK 1.2 (pre-installed)
- Hardware Requirements
 - TWR-KL43Z (or FRDM-KL43Z48M)
 - 1 x Micro USB cable (or 1 x mini USB cable when FRDM used)
 - 2 x jumper cable wires female (or male when FRDM used)
- How to Start
 - Download _KL43_template_ folder here: https://freescale-my.sharepoint.com/personal/b34185_freescale_com/Documents/MyProjects/Trainings/FTF%20Americas/FlexIO%20UART%20example/Empty%20project%20generation.zip (or from shared USB stick)

UART FlexIO Use Case – Requirements (Cont'd)

- Copy whole folder _KL43_template_ in to the
...\\Freescale\\KSDK_1.2.0\\apps\\
- Run the KSDK_NewProjGen.exe located in
...\\Freescale\\KSDK_1.2.0\\apps_KL43_template_\\
 - Enter project name: e.g. FlexIO_UART_example
 - Select 2 (twrkl43z48m) as the target board for the SDK based project (or select 3 if FRDM board used)
 - This will generate project files with the same name located under
...\\Freescale\\KSDK_1.2.0\\apps\\twrkl43z48m\\examples\\
 - Run the FlexIO_UART_example.eww
- Compile the ksdङ_platform_lib - debug (if not already compiled)
- Try build project (e.g. FlexIO_UART_example)

UART FlexIO Use Case – HW Configuration

- Use 2 x jumper cable wires to interconnect pins 1

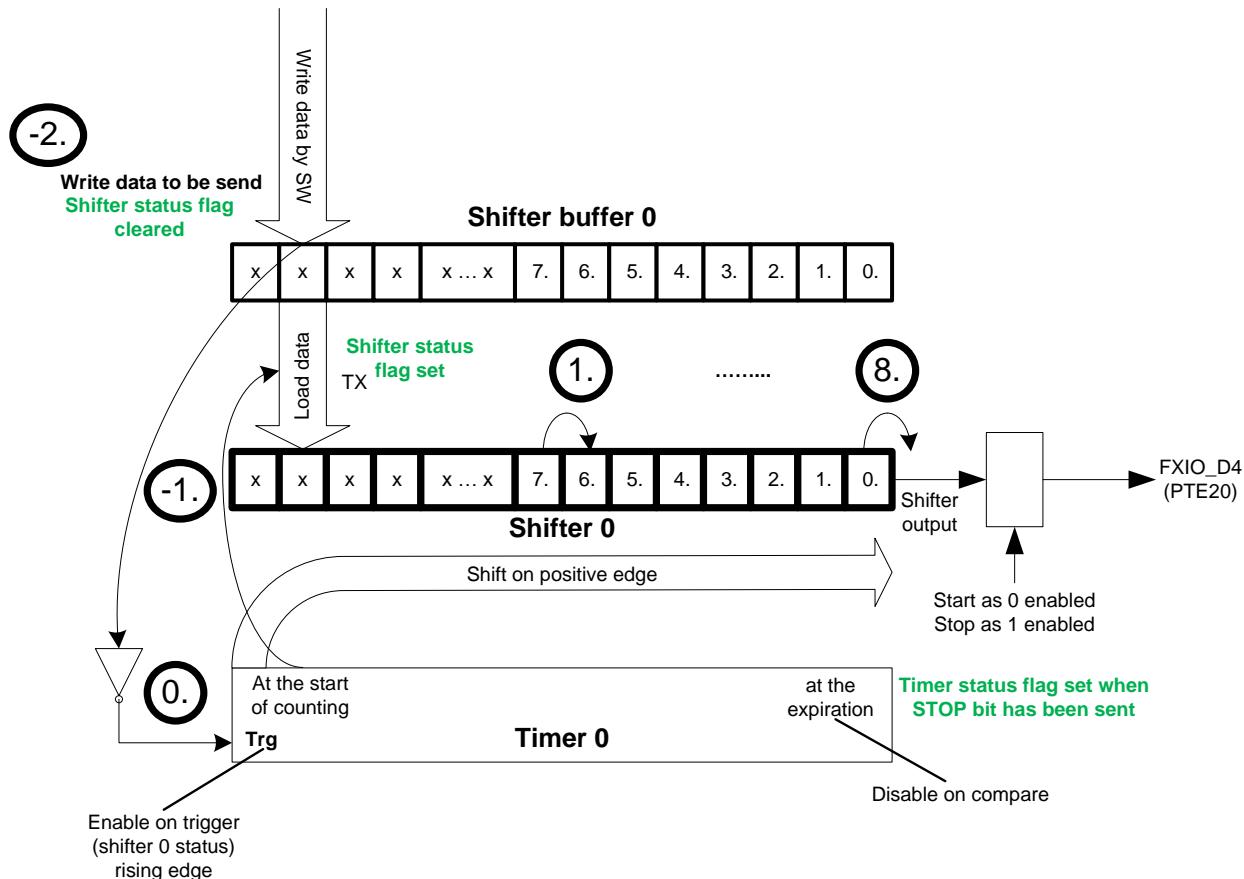
TWR-KL43Z48M	
K20 - OpenSDA serial interface pins	KL43 FlexIO pins
J3 - 3 (RX)	J15 - 1 (FXIO_D4 -> TX)
J5 - 3 (TX)	J15 - 2 (FXIO_D5 -> RX)

- Or for FRDM-KL43Z

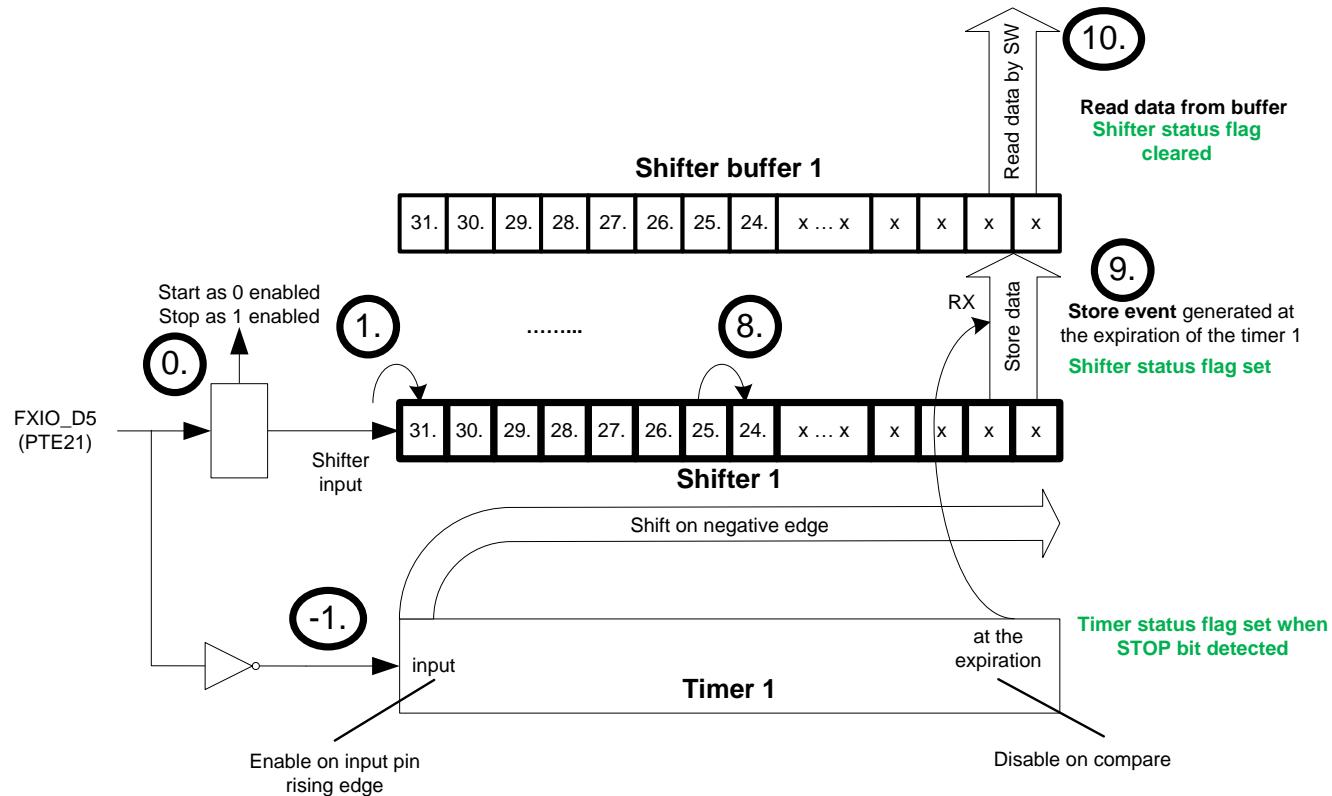
FRDM-KL43Z		
K20 - OpenSDA serial interface pins	MCU bridge	KL43 FlexIO pins
UART1_RX_TGTMCU	J1 - 2 (PTA1 -> RX)	J4 - 1 (FXIO_D4 -> TX)
UART1_TX_TGTMCU	J1 - 4 (PTA2 -> TX)	J4 - 3 (FXIO_D5 -> RX)

- TWR-KL43 (FRDM-KL43) configuration screen shots can be found here: https://freescale-my.sharepoint.com/personal/b34185_freescale_com/Documents/MyProjects/Trainings/FTF%20Americas/FlexIO%20UART%20example/Images (or find it on shared USB stick)
- plug micro (mini) USB cable into the OpenSDA connector on the board side and into the PC
- Try download and debug the project (e.g. FlexIO_UART_example)

UART FlexIO Use Case: Functional Description - Receive



UART FlexIO Use Case: Functional Description – Transmit



UART FlexIO Use Case: Functional Description – Setting the Baud Rate and Bit Count

Timer 0, 1 compare register

15.	Upper byte	8. 7.	Lower byte	0.
Bit count		Baud rate divider		

$2^{*(\text{Num of bits}) - 1}$

$2^{*(8)} - 1 = \text{0x0F}$

$((\text{Baud rate divider}) / 2) - 1$

Baud rate = 9600

FlexIO clock = 2MHz

$((2000000 / 9600) / 2) - 1 = \text{0x67}$

UART FlexIO Use Case – Writing a Code #Includes

- **Include required drivers (e.g. project_name.h which is also included in hardware_init.c)**

- **Include clock manager to handle MCG_Lite**

```
#include "fsl_clock_manager.h"
```

- **Include smc HAL to handle very low power mode**

```
#include "fsl_smc_hal.h"
```

- **Include FlexIO UART driver to handle UART emulated by FlexIO module**

```
#include "fsl_flexio_uart_driver.h"
```

UART FlexIO Use Case – Writing a Code: Ports, Clocks, Low Power Configuration

- **Hardware initialization (hardware_init.c)**

```
/* Define and initialize clock configuration structure */
clock_manager_user_config_t g_demoRunModeClockConfig = DEMO_RUN_MODE_CLOCK_CONFIG;

/* enable clock for PORTs */
CLOCK_SYS_EnablePortClock(PORTA_IDX);
CLOCK_SYS_EnablePortClock(PORTE_IDX);

/* Select FlexIO functionality for PTE20, PTE21 */
PORT_HAL_SetMuxMode(PORTE,20u,kPortMuxAlt6);
PORT_HAL_SetMuxMode(PORTE,21u,kPortMuxAlt6);
/* disable PTA1, PTA2 to put them into the High-Z (required for FRDM-KL43) */
PORT_HAL_SetMuxMode(PORTA,1u,kPortPinDisabled);
PORT_HAL_SetMuxMode(PORTA,2u,kPortPinDisabled);
```

UART FlexIO Use Case – Writing a Code: Ports, Clocks, Low Power Configuration

- **Hardware initialization (hardware_init.c)**

```
/* Select the clock source for FlexIO module -> MCGIRCLK (be able to work in VLPS mode) */
CLOCK_SYS_SetFlexioSrc(0, kClockFlexioSrcMcgIrClk);

/* Allow very low power mode */
SMC_HAL_SetProtection(SMC, kAllowPowerModeVlp);

/* Initialize clock MCG_Lite */
CLOCK_SYS_SetConfiguration(&g_demoRunModeClockConfig);
```

UART FlexIO Use Case – Writing a Code:Clocks Configuration

- **Hardware initialization (e.g. in project_name.h which is included in hardware_init.c)**

```
/**************************************************************************//*
 * @brief  Clock manager user configuration structure initialization
*************************************************************************/
#define DEMO_RUN_MODE_CLOCK_CONFIG
{
    .mcgliteConfig =
    {
        .mcglite_mode      = kMcgliteModeHirc48M,
        .irclkEnable       = true,
        .irclkEnableInStop = true,
        .ircs              = kMcgliteLircSel2M,
        .fcrdiv            = kMcgliteLircDivBy1,
        .lircDiv2          = kMcgliteLircDivBy1,
        .hircEnableInNotHircMode = true,
    },
    .simConfig =
    {
        .er32kSrc = kClockEr32kSrcRtc,
        .outdiv1  = 0U,
        .outdiv4  = 1U,
    },
    .oscerConfig =
    {
        .enable     = false,
        .enableInStop = false,
    }
}
```

NOTE: Try to download the project before continuing

UART FlexIO Use Case – Writing a Code: FlexIO Configuration

- Configuration Structure Definitions (`project_name.c`)

```
/* Define and initialize FlexIO module configuration structure */
static flexio_user_config_t flexioModuleConfig = FLEXIO_MODULE_CONFIG;

/* Define and initialize FlexIO UART driver user configuration structure */
const flexio_uart_userconfig_t flexioUartConfig = FLEXIO_UART_CONFIG;

/* Define FlexIO UART runtime state structure */
static flexio_uart_state_t flexioUartState;

/* Define and initialize FlexIO transmit buffer */
static uint8_t u8TxBuffer[] = {"\n\rThis is a demo - UART emulated by FlexIO \n\r"};

/* Define FlexIO receive buffer */
static uint8_t u8RxBuffer[18];
```

UART FlexIO Use Case – Writing a Code: FlexIO Configuration

- FlexIO general features configuration (project_name.h)

```
/**************************************************************************/*!  
 * @brief  FlexIO module instance definition  
*****/  
#define FLEXIO_0      0x0u  
  
/**************************************************************************/*!  
 * @brief  FlexIO module configuration structure initialization  
 * @details Flexio is enabled to work in debug and doze (stop, wait) mode.  
 *          Fast acces is not enabled.  
*****/  
#define FLEXIO_MODULE_CONFIG  
{  
    .useInt = true,  
    .onDozeEnable = false,  
    .onDebugEnable = false,  
    .fastAccessEnable = false  
}
```

UART FlexIO Use Case – Writing a Code: FlexIO Configuration

- FlexIO UART Driver features configuration (project_name.h)

```
/**************************************************************************//*
 * @brief FlexIO UART configuration structure initialization
 * @details The UART communication is configured to 8-bit mode, 9600baud/s with
 *         duplex communication.
*****
#define FLEXIO_UART_CONFIG
{
    .baudRate = 9600,
    .bitCounter = kFlexIOUart8BitsPerChar,
    .uartMode = flexioUART_TxRx,
    .txConfig = FLEXIO_UART_TX_CONFIG,
    .rxConfig = FLEXIO_UART_RX_CONFIG,
}

/**************************************************************************//*
 * @brief FlexIO UART transmitter configuration structure initialization
 * @details Flexio UART transmitter uses FXIO_D4 pin, shifter 0 and timer 0.
*****
#define FLEXIO_UART_TX_CONFIG
{
    .pinIdx = 4,
    .shifterIdx = 0,
    .timerIdx = 0,
}
```



UART FlexIO Use Case – Writing a Code: FlexIO Configuration

- FlexIO UART Driver features configuration (project_name.h)

```
/**************************************************************************//*
 * @brief  FlexIO UART receiver configuration structure initialization
 * @details Flexio UART receiver uses FXIO_D5 pin, shifter 1 and timer 1.
 ****/
#define FLEXIO_UART_RX_CONFIG
{
    .pinIdx = 5,
    .shifterIdx = 1,
    .timerIdx = 1,
}
```

UART FlexIO Use Case – Writing a Code: FlexIO Configuration

- **FlexIO Initialization (project_name.c, in main function after hardware_init ())**

```
/* FlexIO module initialization */
FLEXIO_DRV_Init(FLEXIO_0,&flexioModuleConfig);

/* FlexIO UART emulation driver initialization */
FLEXIO_UART_DRV_Init(FLEXIO_0, &flexioUartState, &flexioUartConfig);

/* Enable FlexIO module */
FLEXIO_DRV_Start(FLEXIO_0);
```

UART FlexIO Use Case – Writing a Code: FlexIO Configuration

- FlexIO transmit / receive (project_name.c, in main function)

```
/* Send initial message */
FLEXIO_UART_DRV_SendData(&flexioUartState, u8TxBuffer, sizeof(u8TxBuffer));
/* Wait until buffer send out */
while(flexioUartState.isTxBusy);

/* Initialize transmit status buffer pointer to receive buffer */
flexioUartState.txBuff = u8RxBuffer;

while(1)
{
    /* Fill the receive buffer by receiving data */
    FLEXIO_UART_DRV_ReceiveData(&flexioUartState, u8RxBuffer, sizeof(u8RxBuffer));

    /* Send received character back to console - ECHO */
    if (flexioUartState.txBuff != flexioUartState.rxBuff)
    {
        FLEXIO_UART_DRV_SendData(&flexioUartState, (flexioUartState.rxBuff-1), 1);
        flexioUartState.txBuff++;
    }
}
```

UART FlexIO Use Case – Writing a Code: FlexIO Configuration

- **FlexIO IRQ handler definition (fsl_project_name_irq.c)**

```
/**************************************************************************//*
 * @brief  Flexio interrupt handler
 * @details Call the FLEXIO driver interrupt handler (it is handling
 *          shifter status flags and shifter callback TX/RX functions).
 ****/
void UART2_FLEXIO_IRQHandler(void)
{
    FLEXIO_DRV_IRQHandler(FLEXIO_IDX);
}
```

UART FlexIO Use Case – Writing a Code: FlexIO Configuration – Data Match Wake-up Emulation

- Configure VLPS mode (project_name.c)

```
/* Define and initialize power mode control configuration strcuture */
static smc_power_mode_config_t power_mode_config = SMC_POWER_MODE_CONFIG;
```

- Configure VLPS mode (project_name.h)

```
/**************************************************************************!
 * @brief  SMC power mode configuration structure initialization
 * @details VLPS mode selected.
 *****/
#define SMC_POWER_MODE_CONFIG
{
    .powerModeName = kPowerModeVlps,
    .stopSubMode = kSmcStopSub0,
    .porOptionValue = kSmcPorDisabled,
    .pstopOptionValue = kSmcPstopStop,
}
```

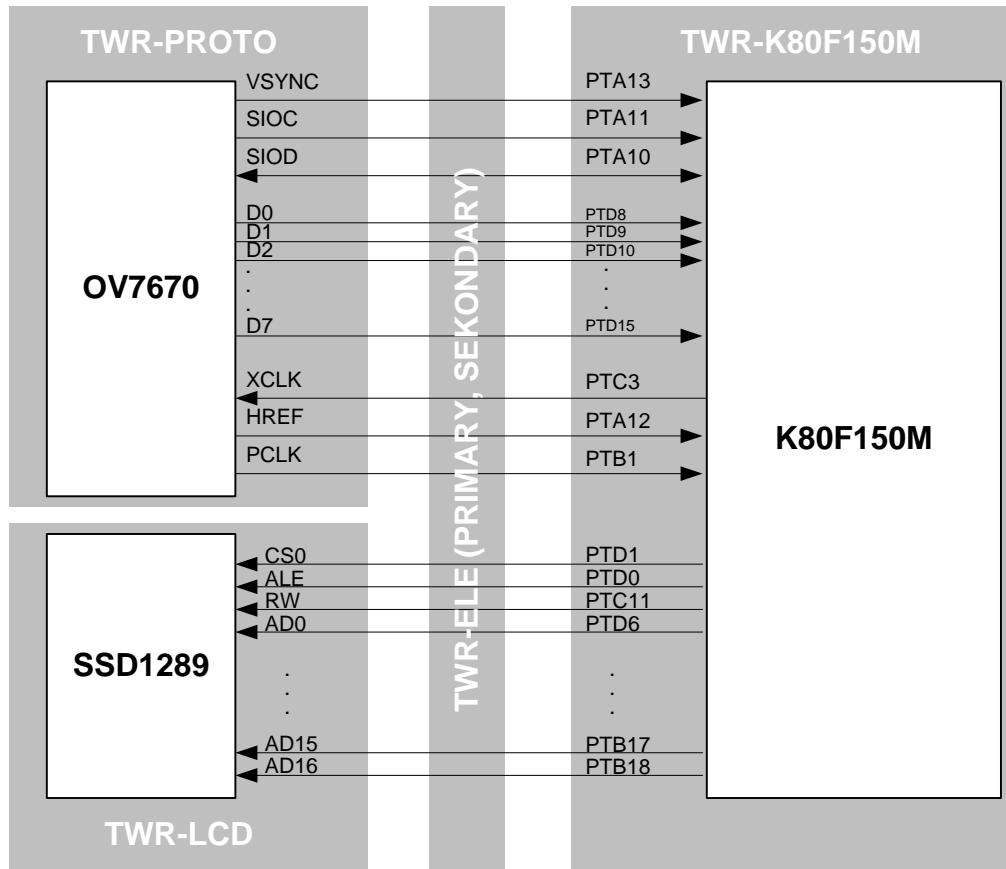
UART FlexIO Use Case – Writing a Code: FlexIO Configuration – Data Match Wake-up Emulation

- Enter VLPS mode (project_name.c, in while(1) loop)

```
/* ask the last received character is 'E' */  
if (*(flexioUartState.rxBuff-1) == 'E')  
{  
    /* Reconfigure FlexIO shifter 1 receive mode into the match store mode */  
    FLEXIO_WR_SHIFTCTL_SMOD(FLEXIO, 1, 0x4);  
  
    /* Write data match character 'w' into the appropriate shifter buffer register */  
    FLEXIO_HAL_SetShifterBuffer(FLEXIO, 1, ('w'<<24));  
  
    /* Wait until transfer done – to avoid abort STOP mode */  
    while(flexioUartState.isTxBusy);  
  
    /* Wait until transfer done – to avoid abort STOP mode */  
    SMC_HAL_SetMode(SMC, &power_mode_config);  
  
    /* Reconfigure FlexIO shifter 1 back to receive mode */  
    FLEXIO_WR_SHIFTCTL_SMOD(FLEXIO, 1, 0x1);  
}
```

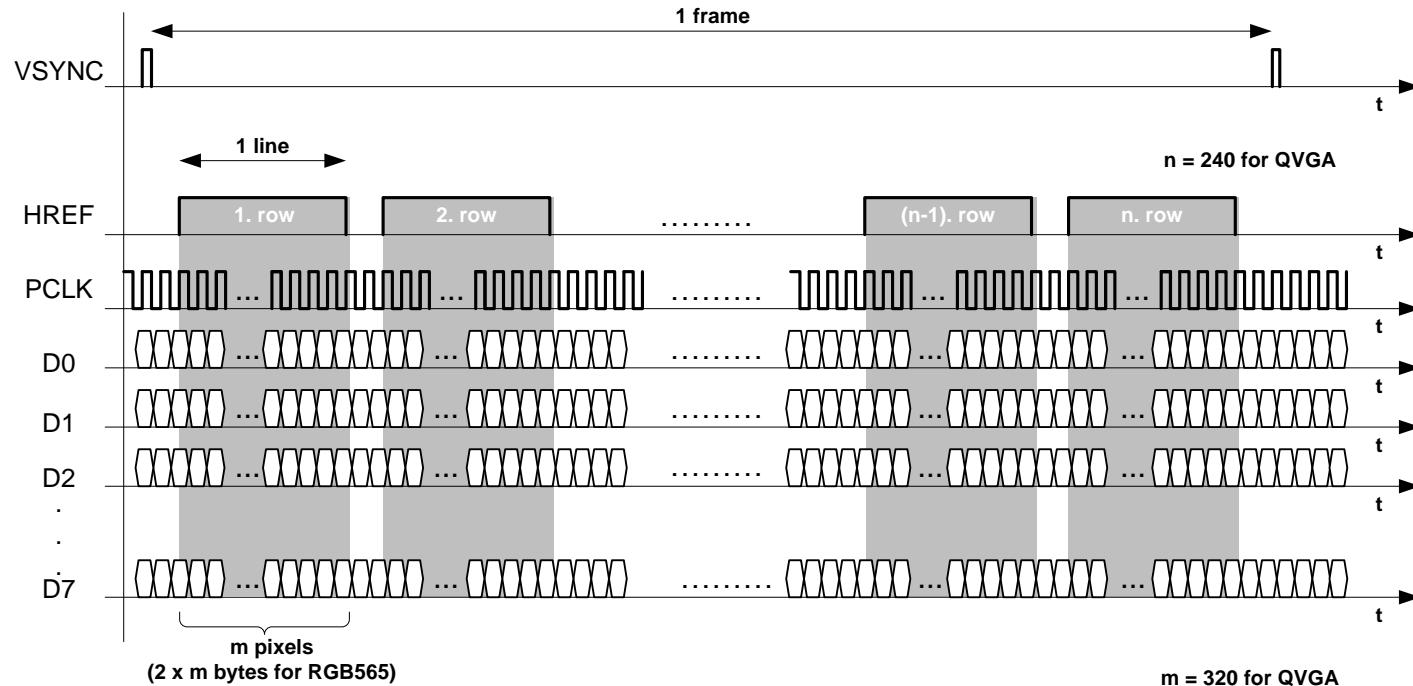
CMOS VGA Camera OV7670 FlexIO Use Case Demonstration

- General HW Configuration



CMOS VGA Camera OV7670 FlexIO Use Case Demonstration

- OV7670 Frame and horizontal timing



FlexIO Timer0 trigger is HREF

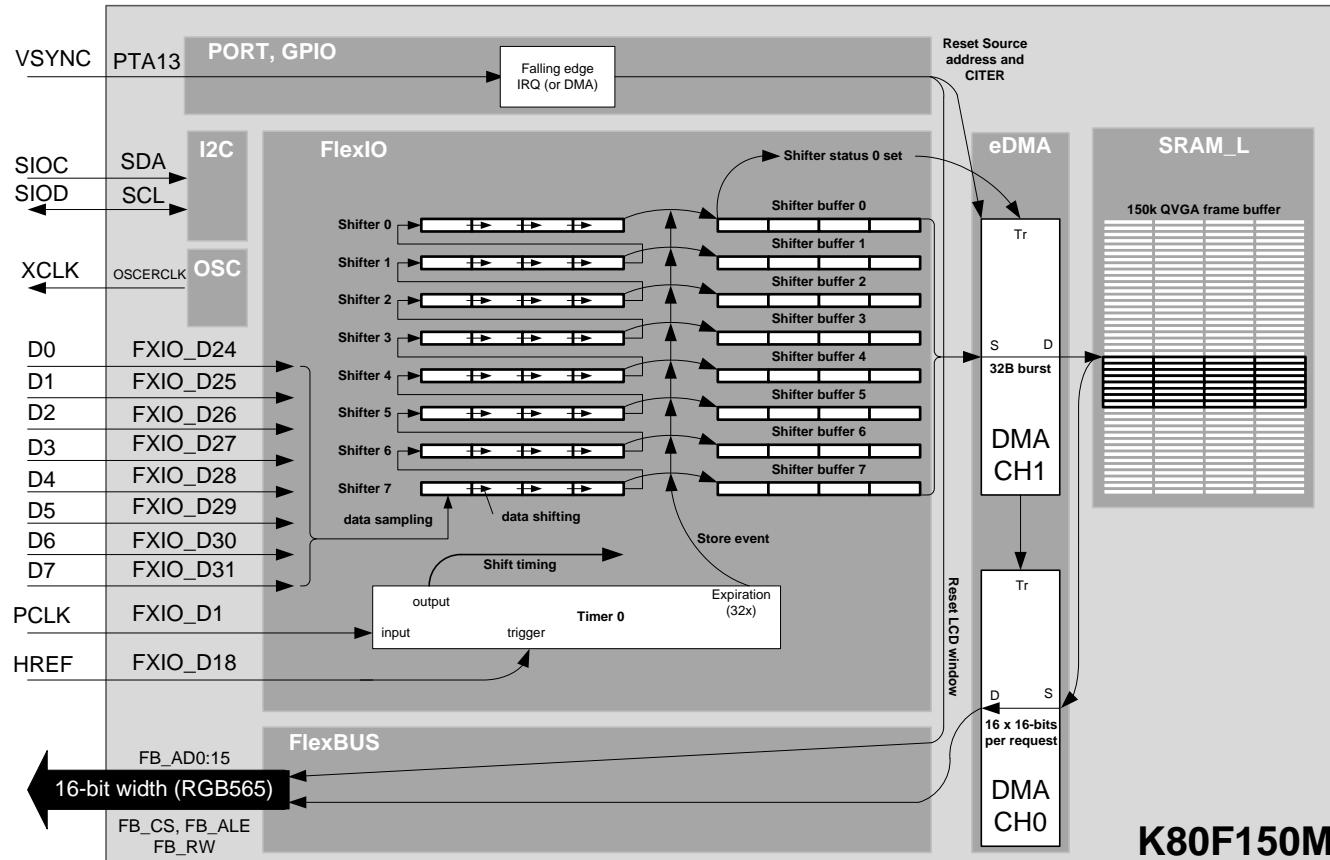
when HREF high timer enabled

FlexIO Timer0 input is PCLK

rising edge on PCLK is shift clock – data sampling

CMOS VGA Camera OV7670 FlexIO Use Case Demonstration

- Functional description – QVGA with RGB565 mode





www.Freescale.com