



| FTF 2016
TECHNOLOGY FORUM

DEVELOPING APPLICATIONS BASED ON FLEXIO

FTF-MHW-N1950

DONNIE GARCIA, MICR SYS & APPLICATIONS ENG

DEREK SNELL, SENIOR FAE

FTF-MHW-N1950

MAY 17, 2016

PUBLIC USE





AGENDA

- Smart Peripherals and IoT
- What is FlexIO
- Enablement for FlexIO
- Autonomous Monitoring Example
 - Hands On
- Advanced Interfaces
 - Demonstration
- Summary

Explosive Growth of Smart Connected Solutions



SMART
HOME



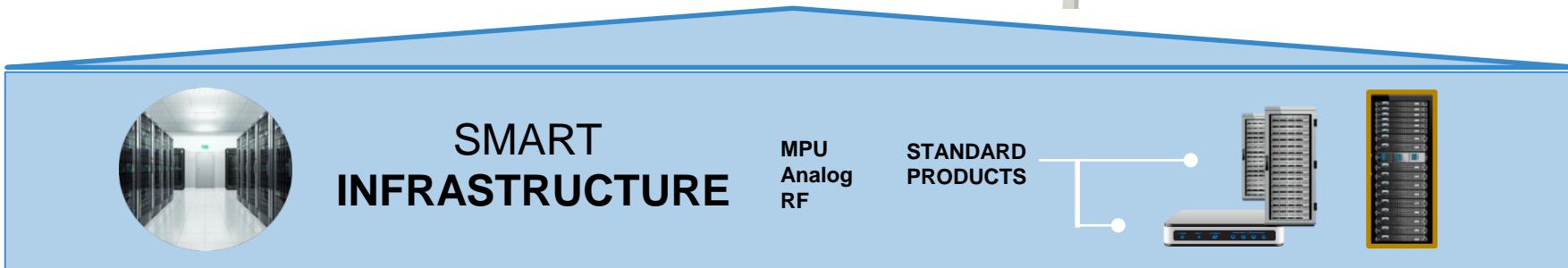
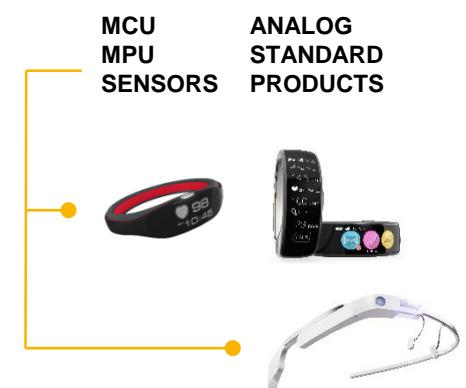
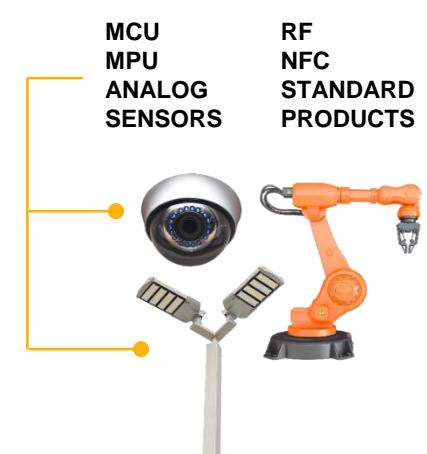
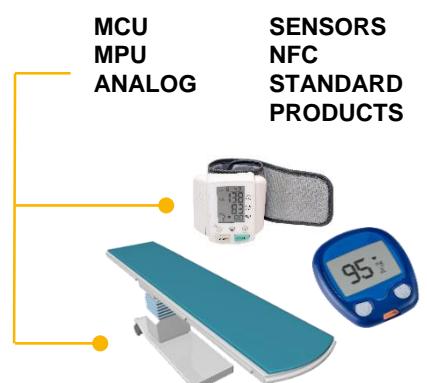
SMART
HEALTHCARE



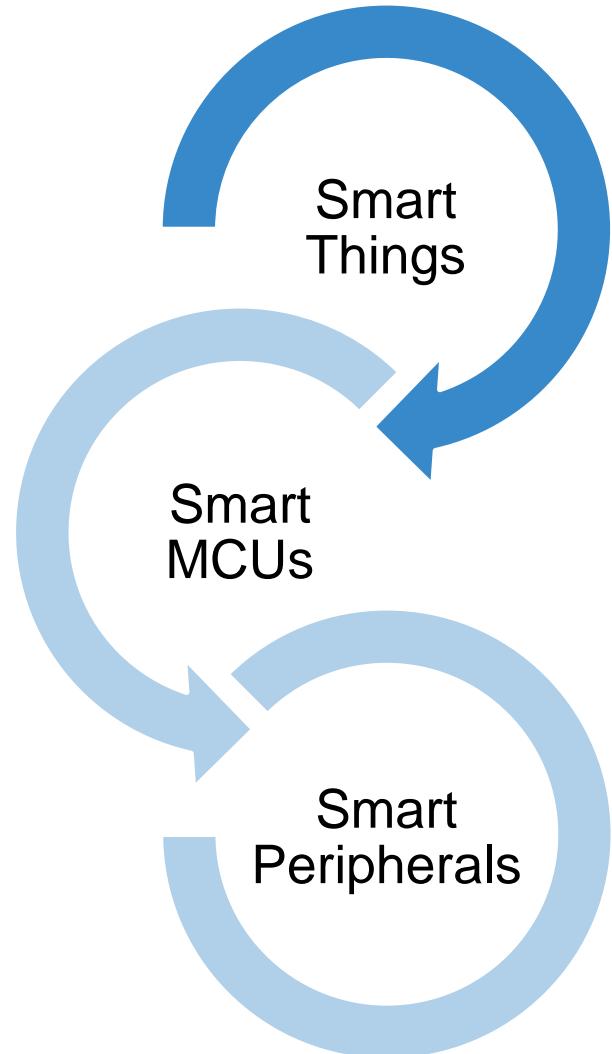
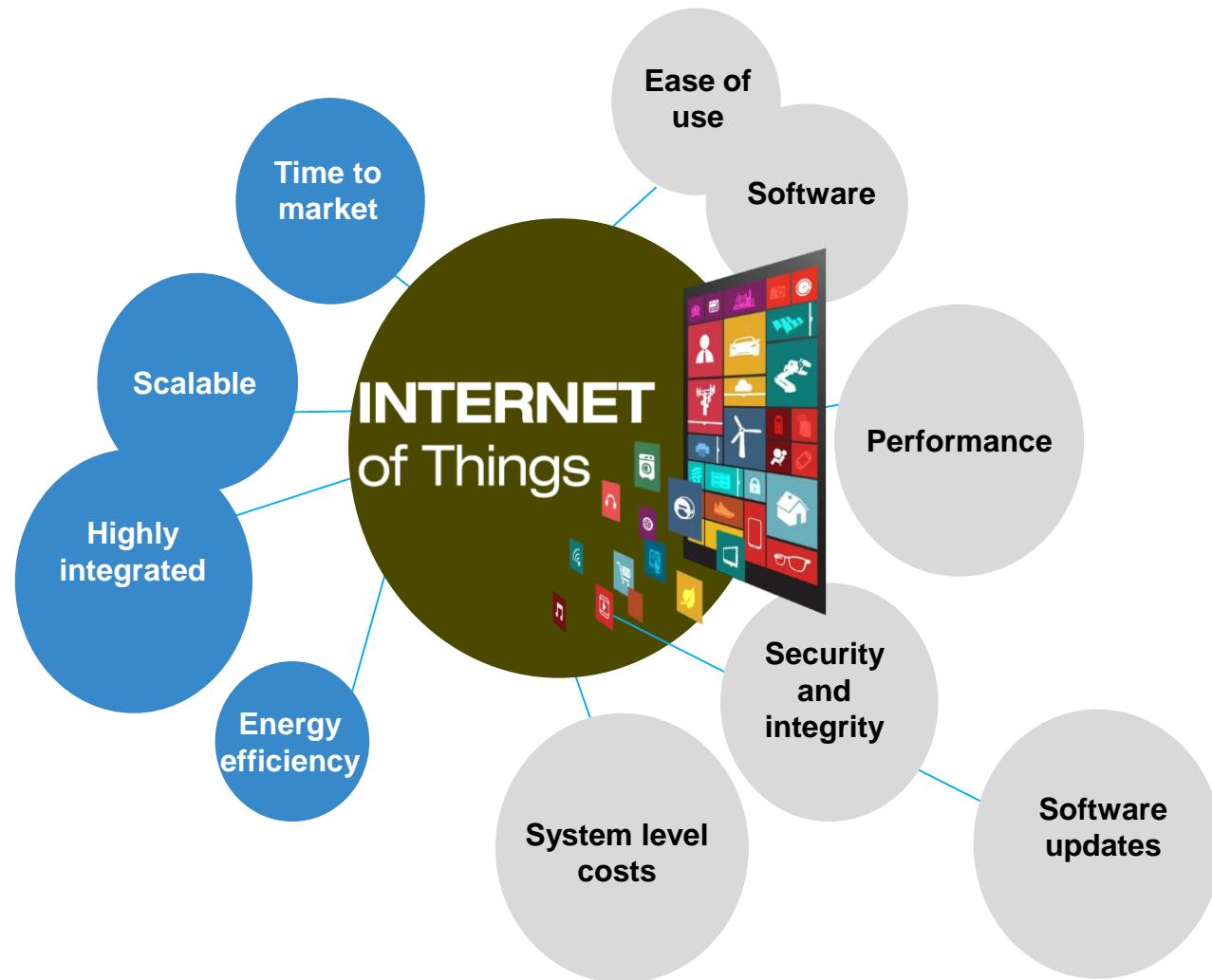
SMART
INDUSTRY



SMART
WEARABLES



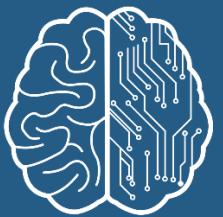
Challenges for Today's Embedded Designs



Attributes of a Smart Peripheral

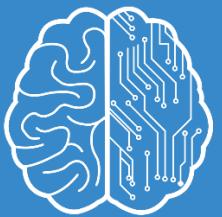
Asynchronous

*Independent clocking
and the ability to
perform operations
based on events*



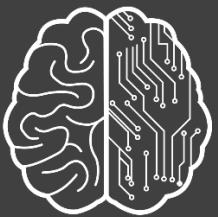
Autonomous

*Capable of
independent decisions
and actions*

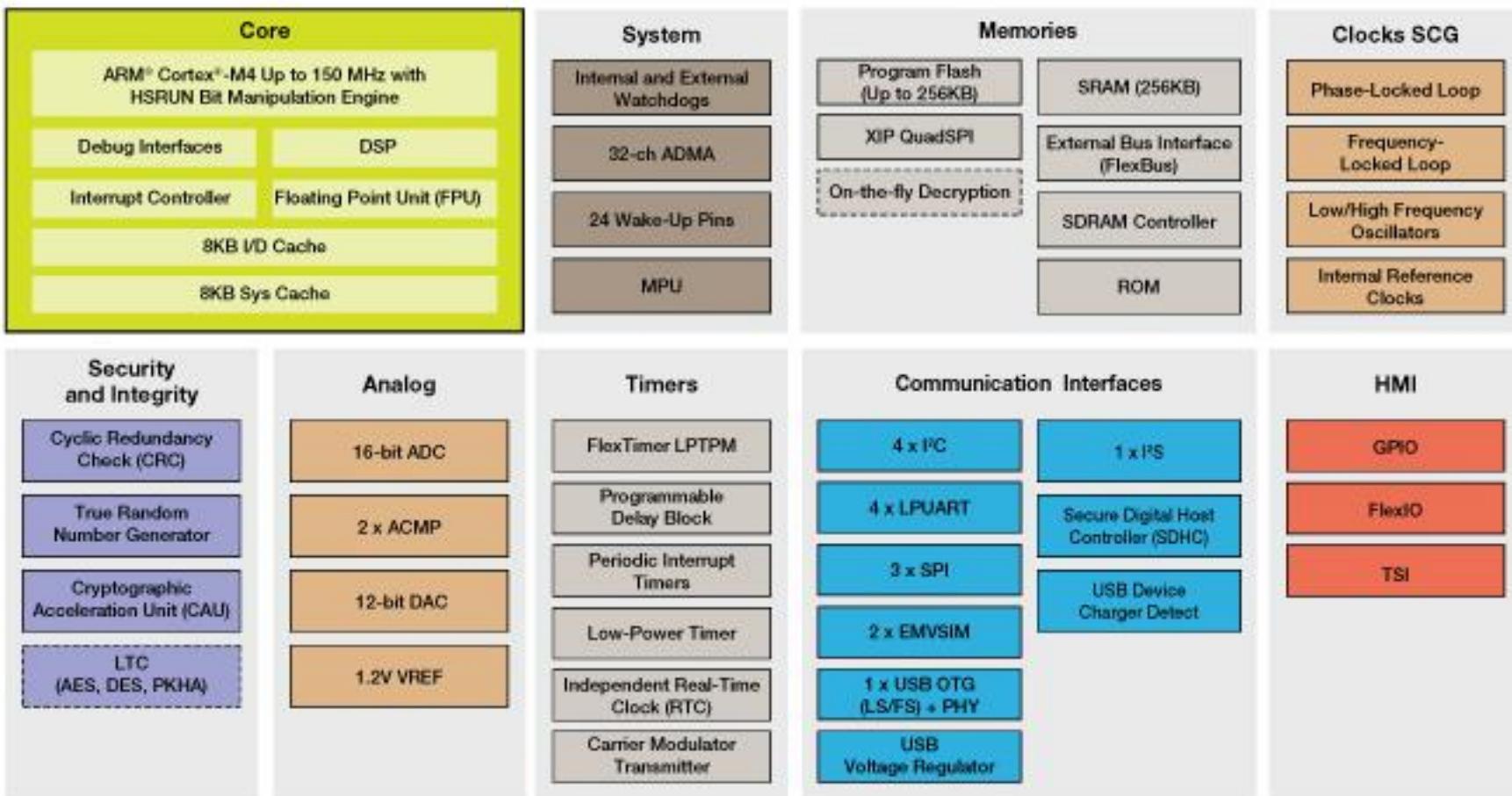


Adaptable

*Able to be modified to
for new use or
purpose*

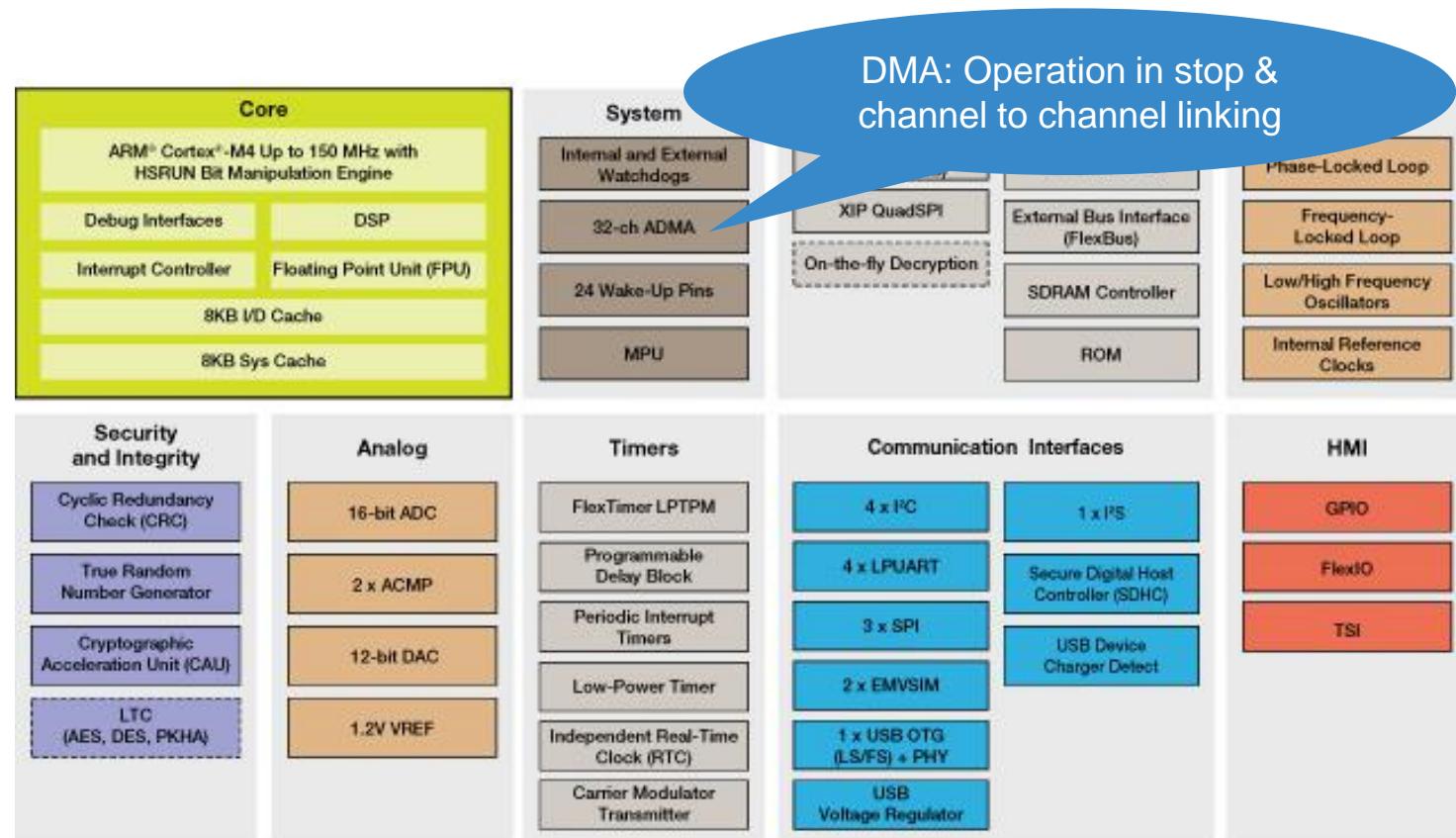


Example MCU Block Diagram

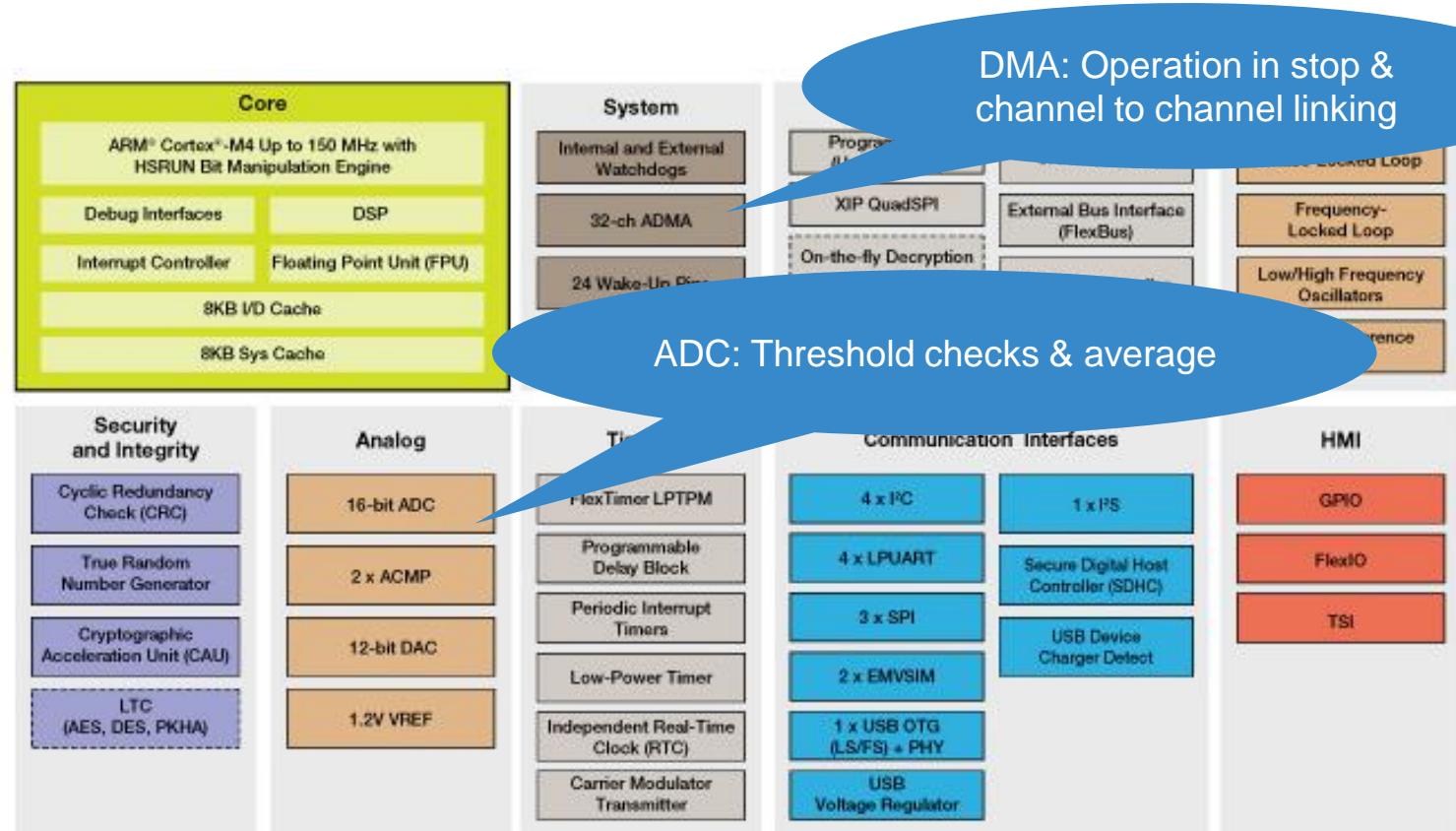


Cryptographic hardware support and execution from external encrypted memories is available on the Kinetis K82 MCU

Example MCU Block Diagram

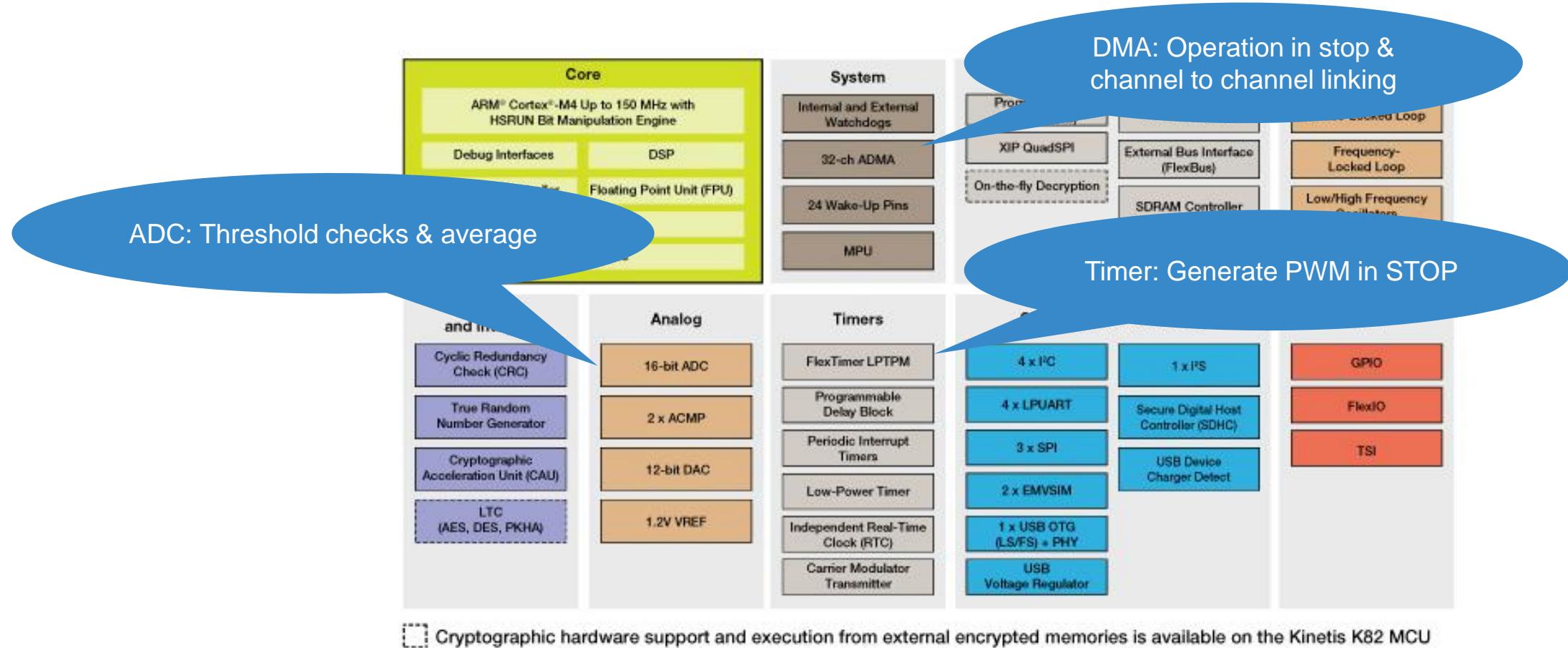


Example MCU Block Diagram

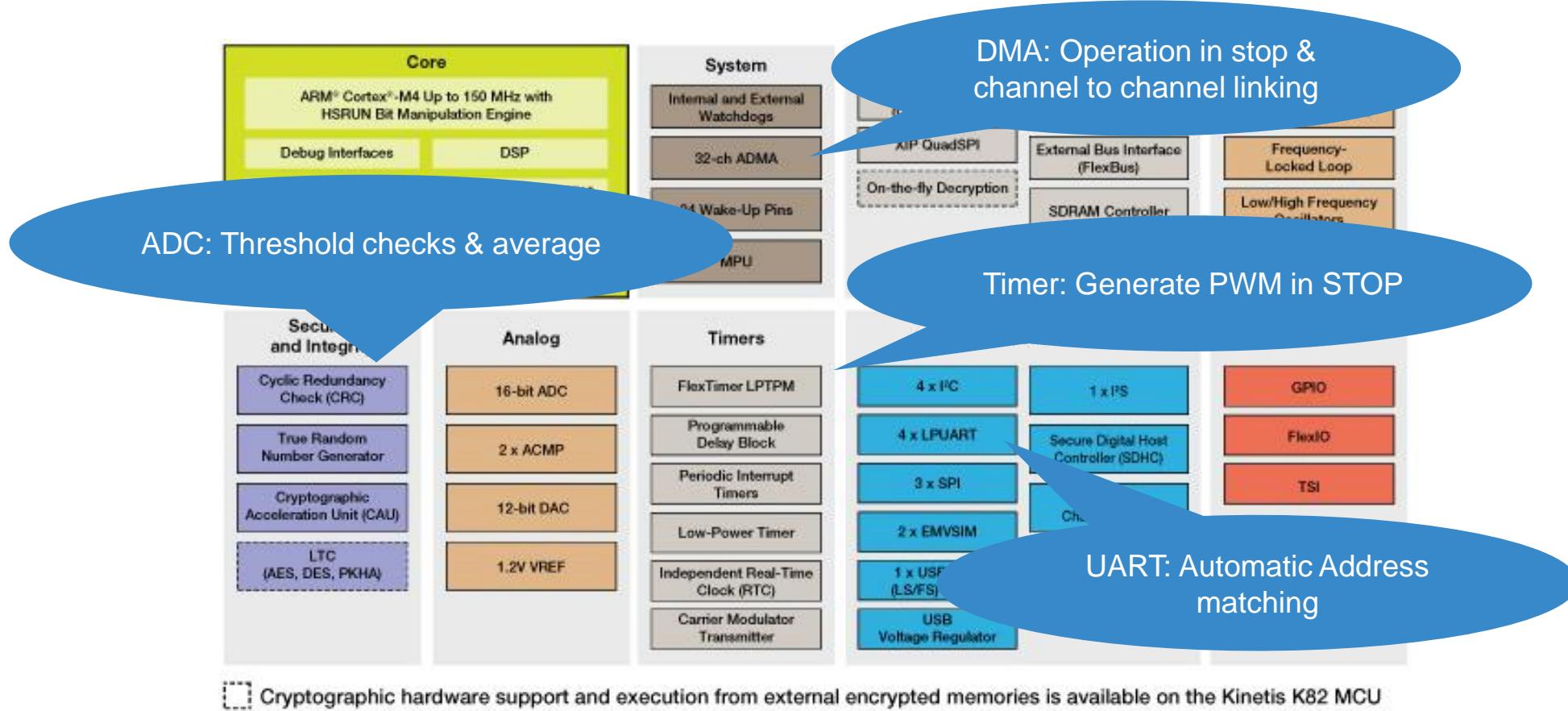


[] Cryptographic hardware support and execution from external encrypted memories is available on the Kinetis K82 MCU

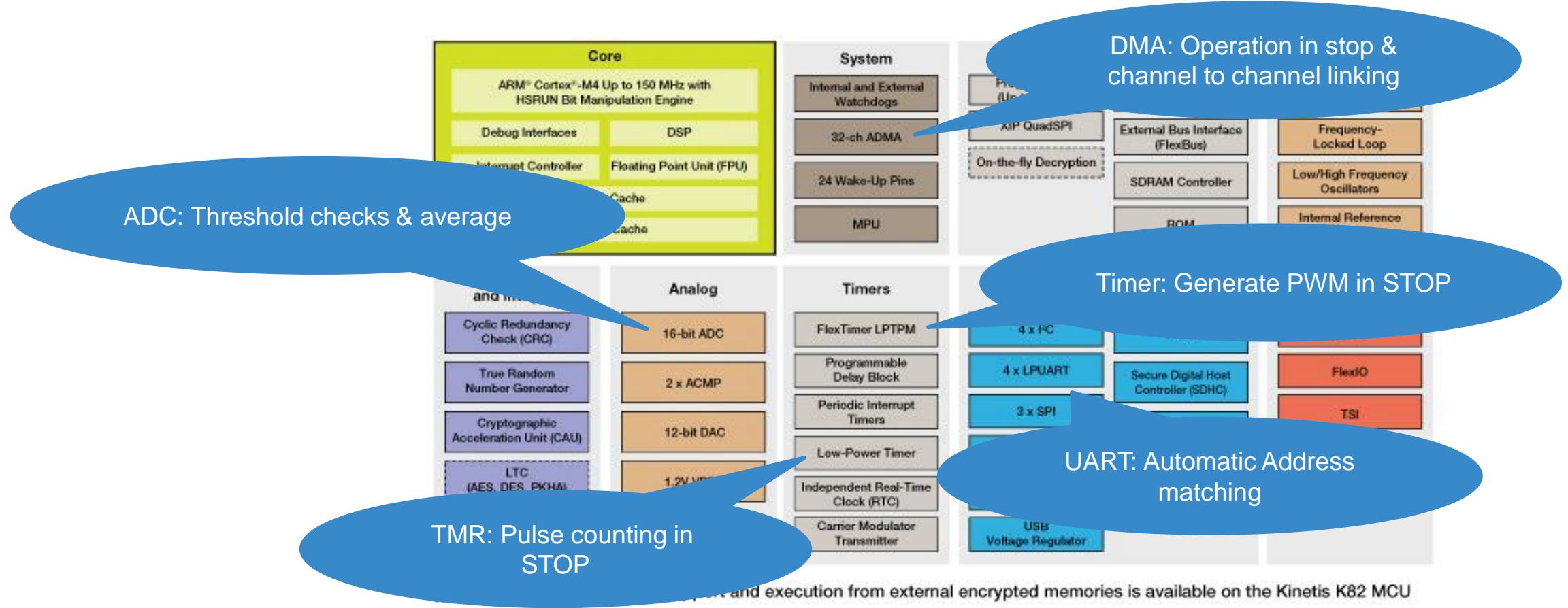
Example MCU Block Diagram



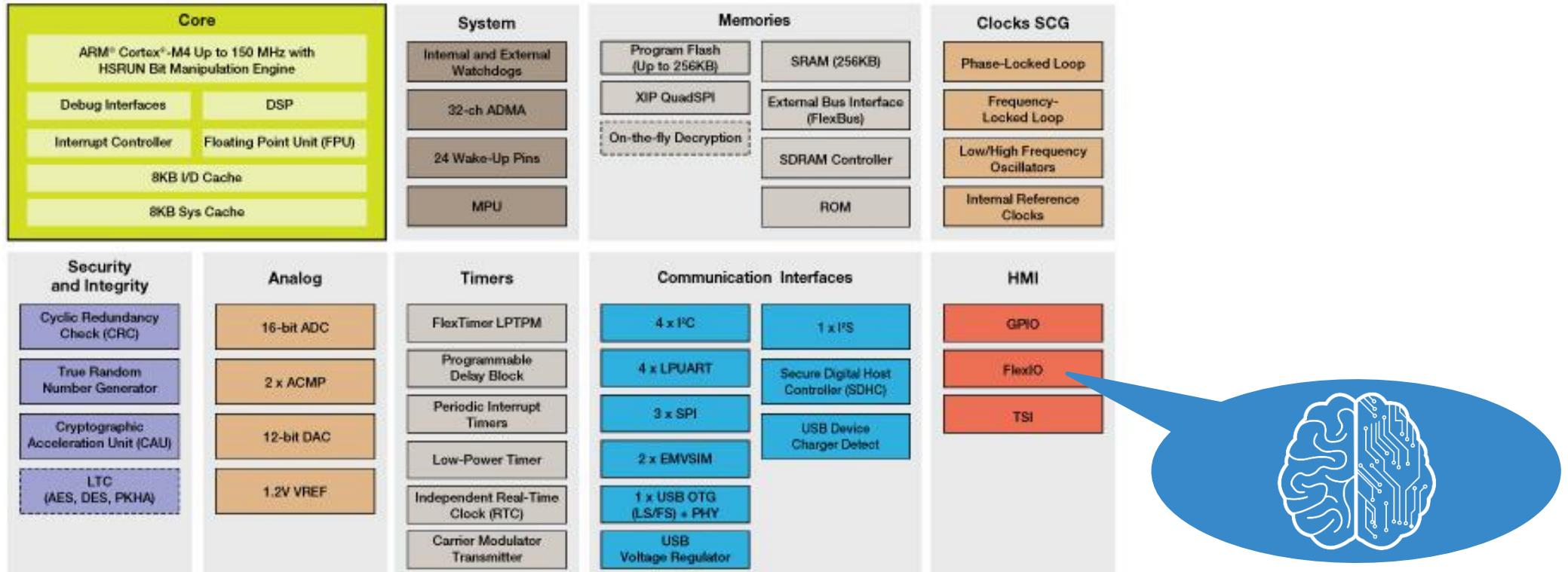
Example MCU Block Diagram



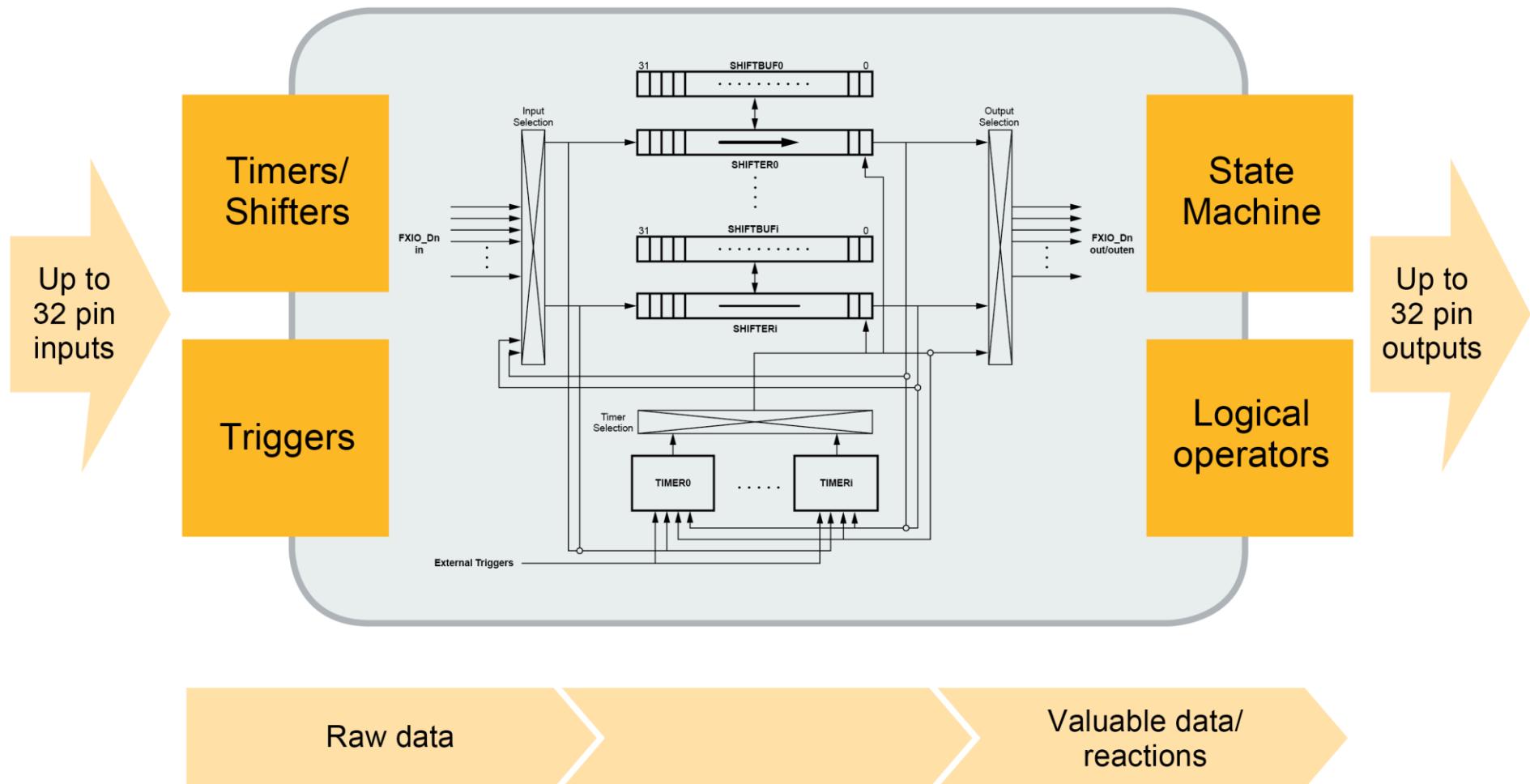
Example MCU Block Diagram



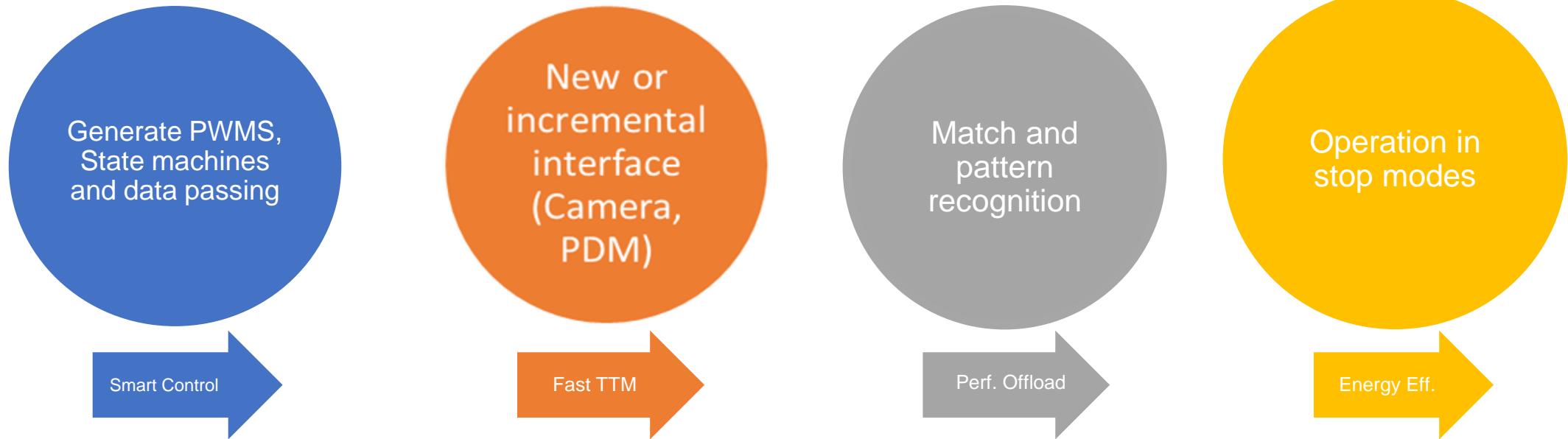
Example MCU Block Diagram



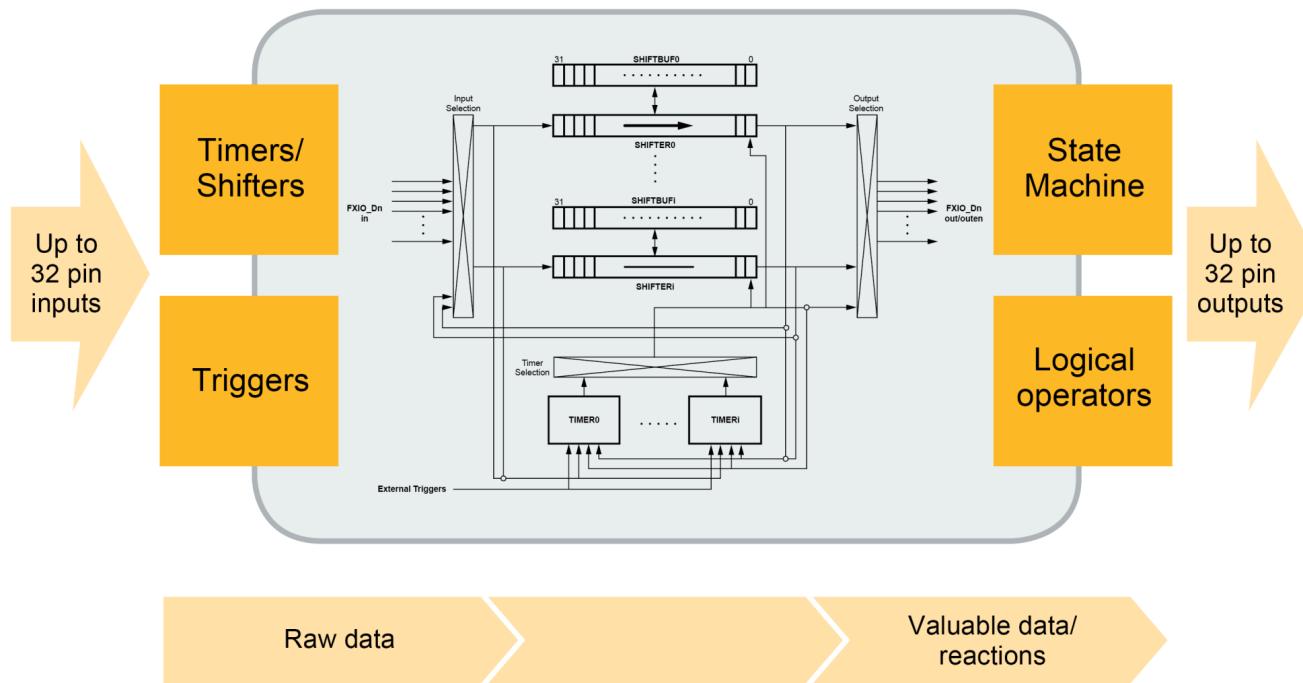
FlexIO: The Smartest Peripheral



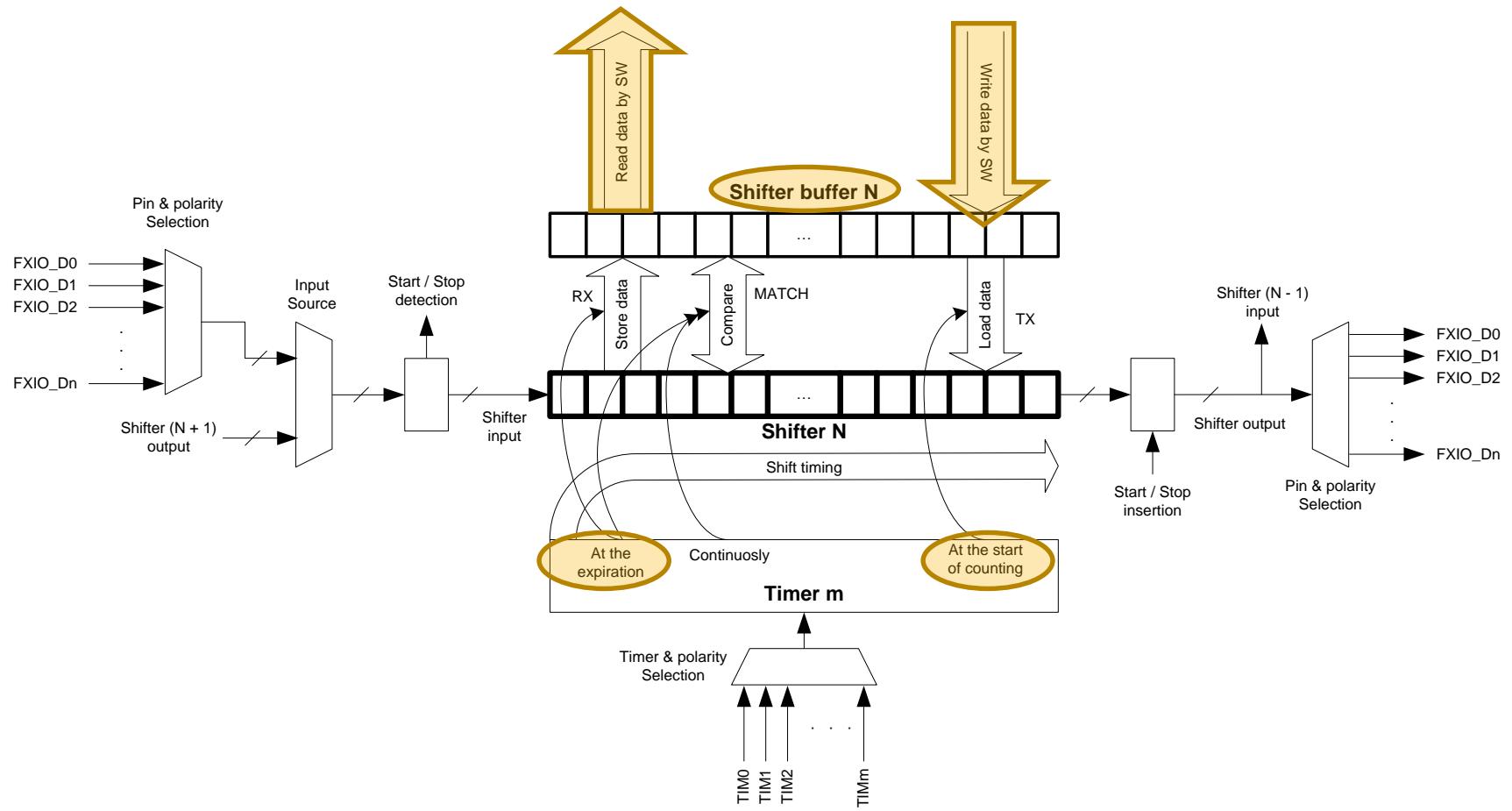
Use the Flex IO Smart Peripheral to Change Raw Data Into Valuable Data and Reactions



Use the Flex IO Smart Peripheral to Change Raw Data Into Valuable Data and Reactions

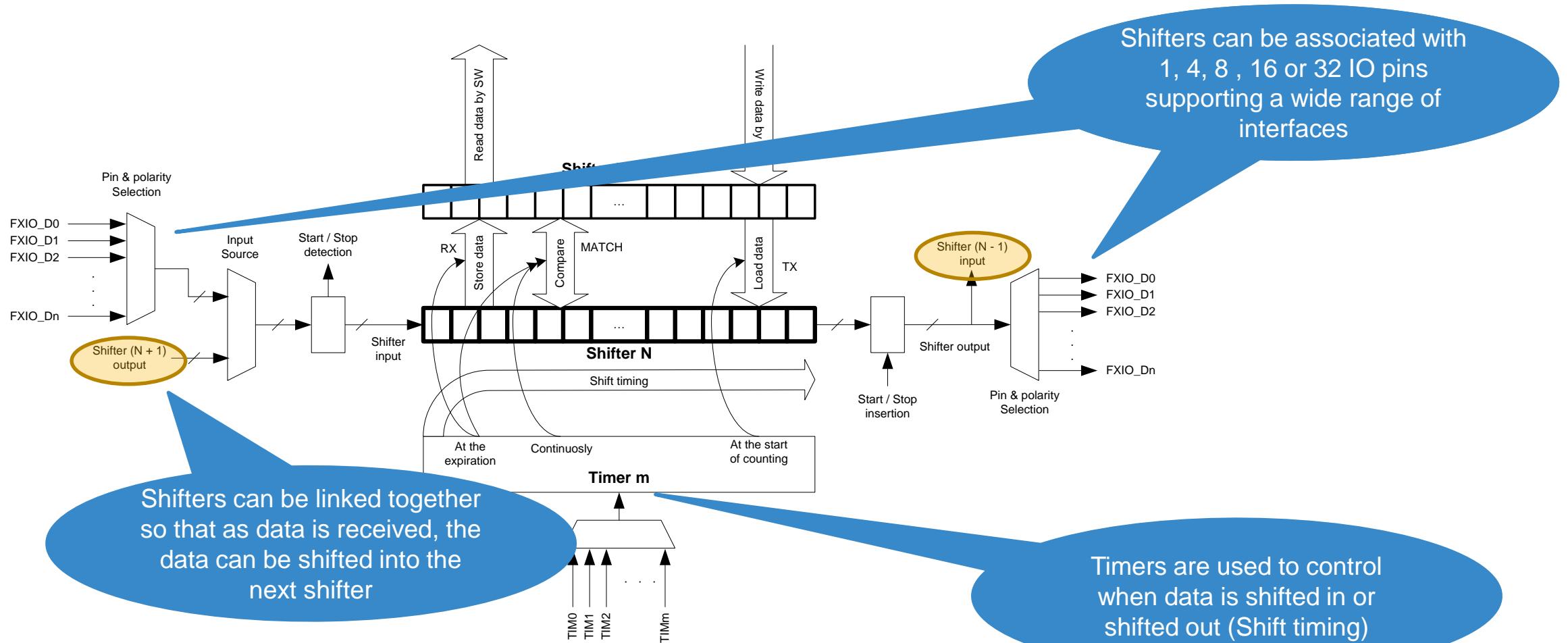


Shifters and Timers



The MCU on the FRDM-K82F has 8 shift registers & 8 Timers

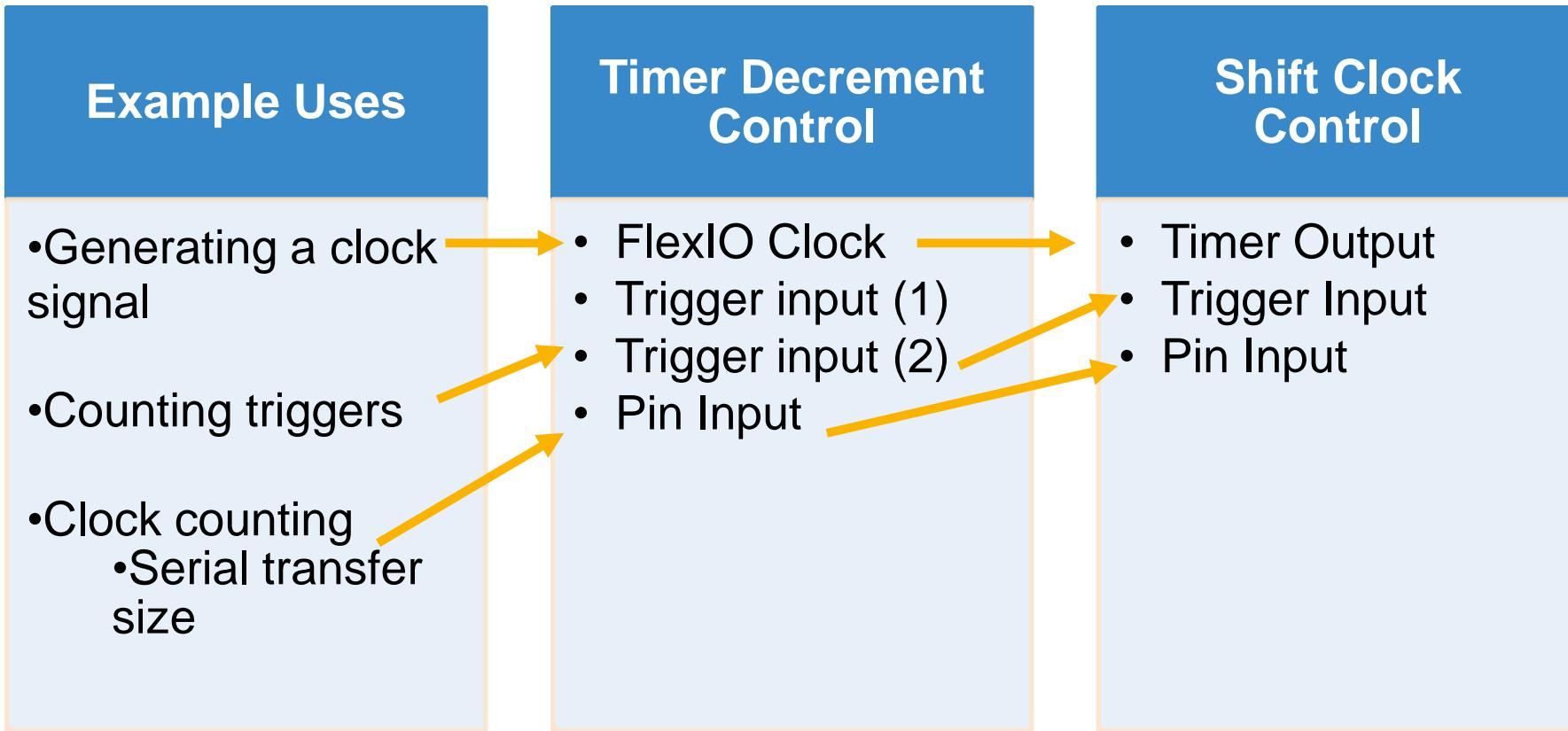
Shifters and Timers



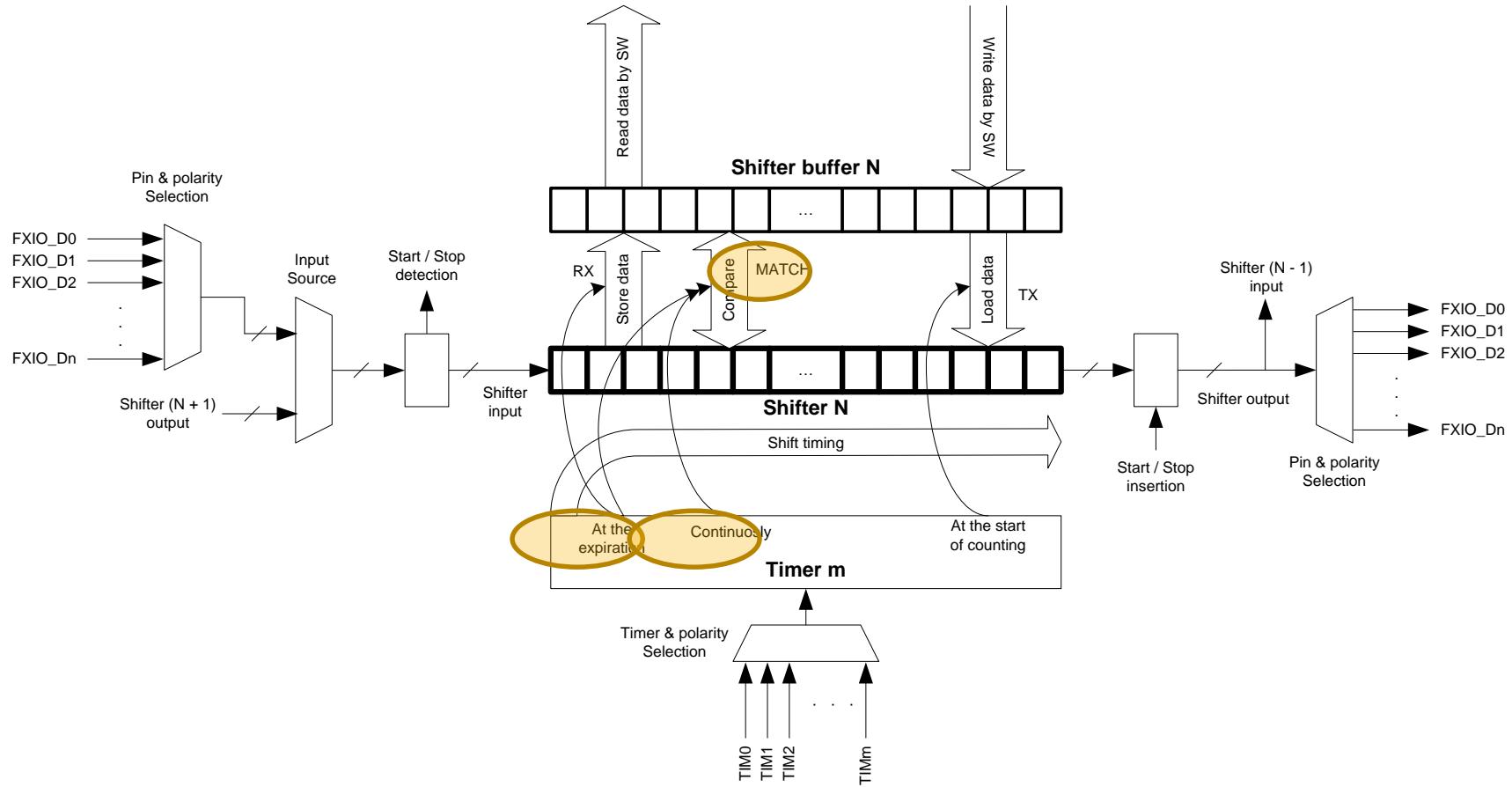
Timer Triggers

Internal <i>(to the FlexIO)</i> Triggers	Any FlexIO pin
	Any FlexIO Shifter status flag
	Any other timer output
External <i>(Chip Specific)</i> Triggers	Chip Trigger pin (EXTRIG_IN)
	Chip Comparators
	Chip Timers
	DMA Timers (PIT) PWM timers (FTM) Low Power Timers (LPTMR) RTC (Seconds and Alarm)

Timers: Uses and Generating Shift Timing



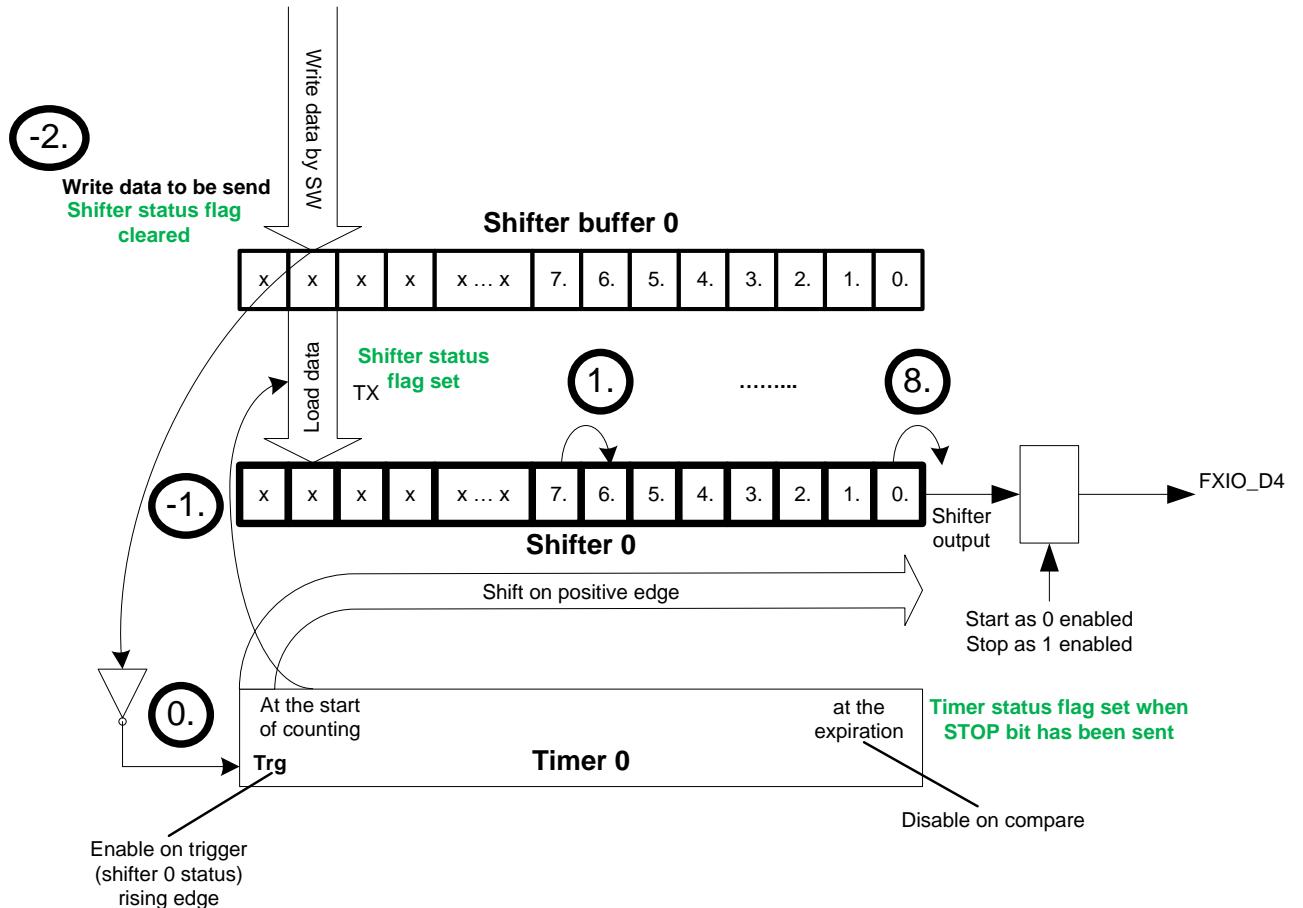
Logical Operators: Match Capabilities



- Each shift register supports 16 bits of match data and 16 bits of mask data to easily configure thresholds
- Match can be used to monitor data and create reactions

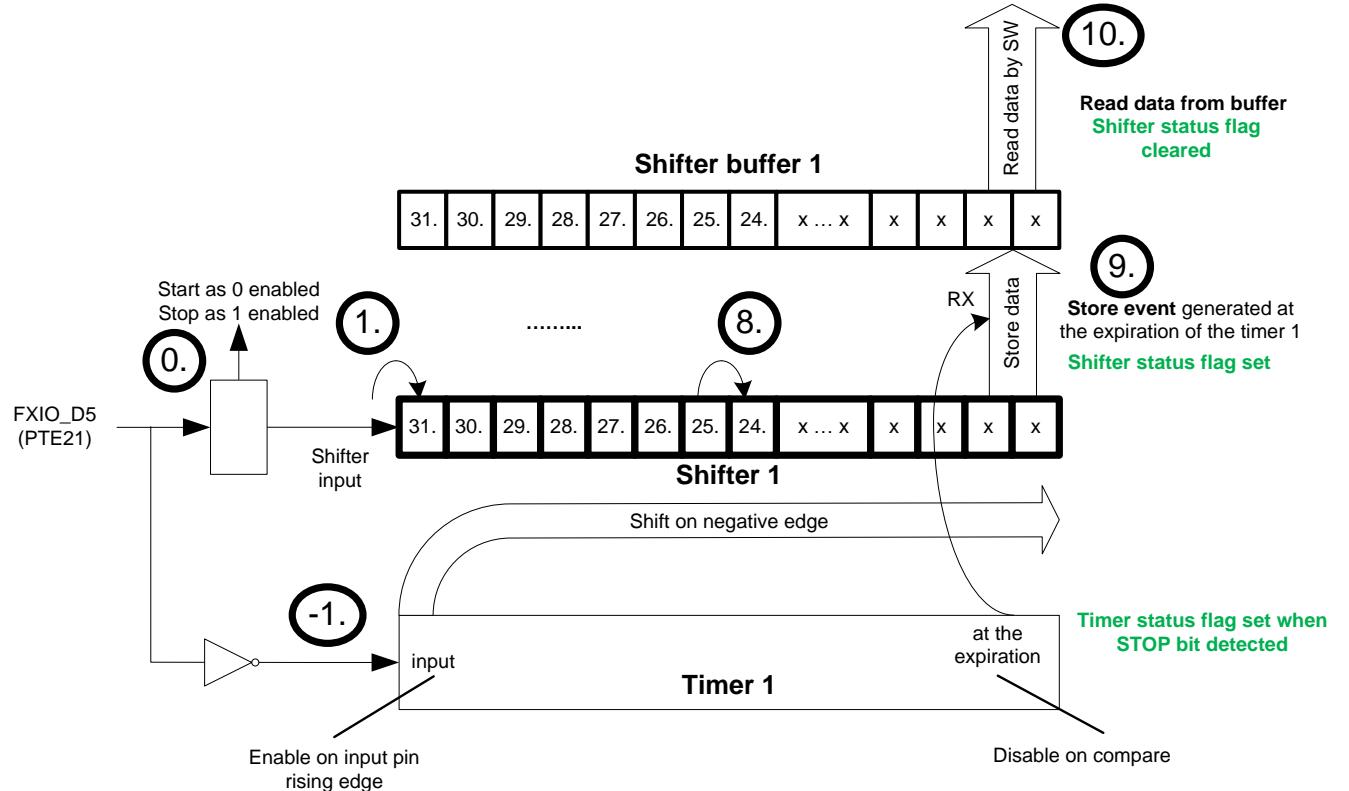
FlexIO Use Case: Functional Description – Transmit Data

- Starting at -2, the shifter is ready for data to be loaded, SW or DMA is used to write the data
- The timer triggers after the data is written as the shifter status flag is set
- The timer starts counting and data is shifted out (transmitted)

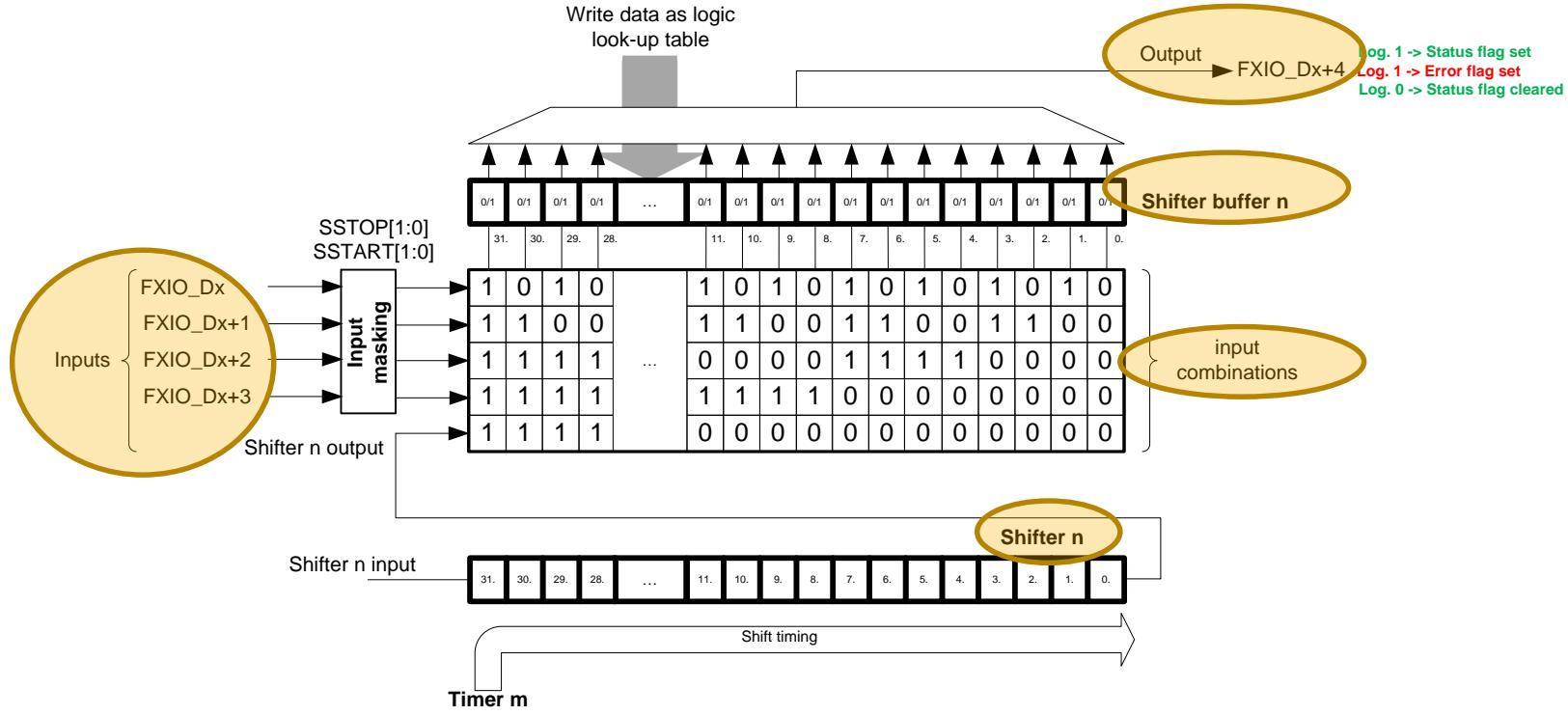


FlexIO Use Case: Functional Description – Receive Data

- Starting at -1, timer counting is triggered by the rising edge of a FlexIO pin (a clock)
- Data is shifted into the shift register until the bit count is reached at the expiration of the timer
- The data is transferred to the shift buffer and can be read



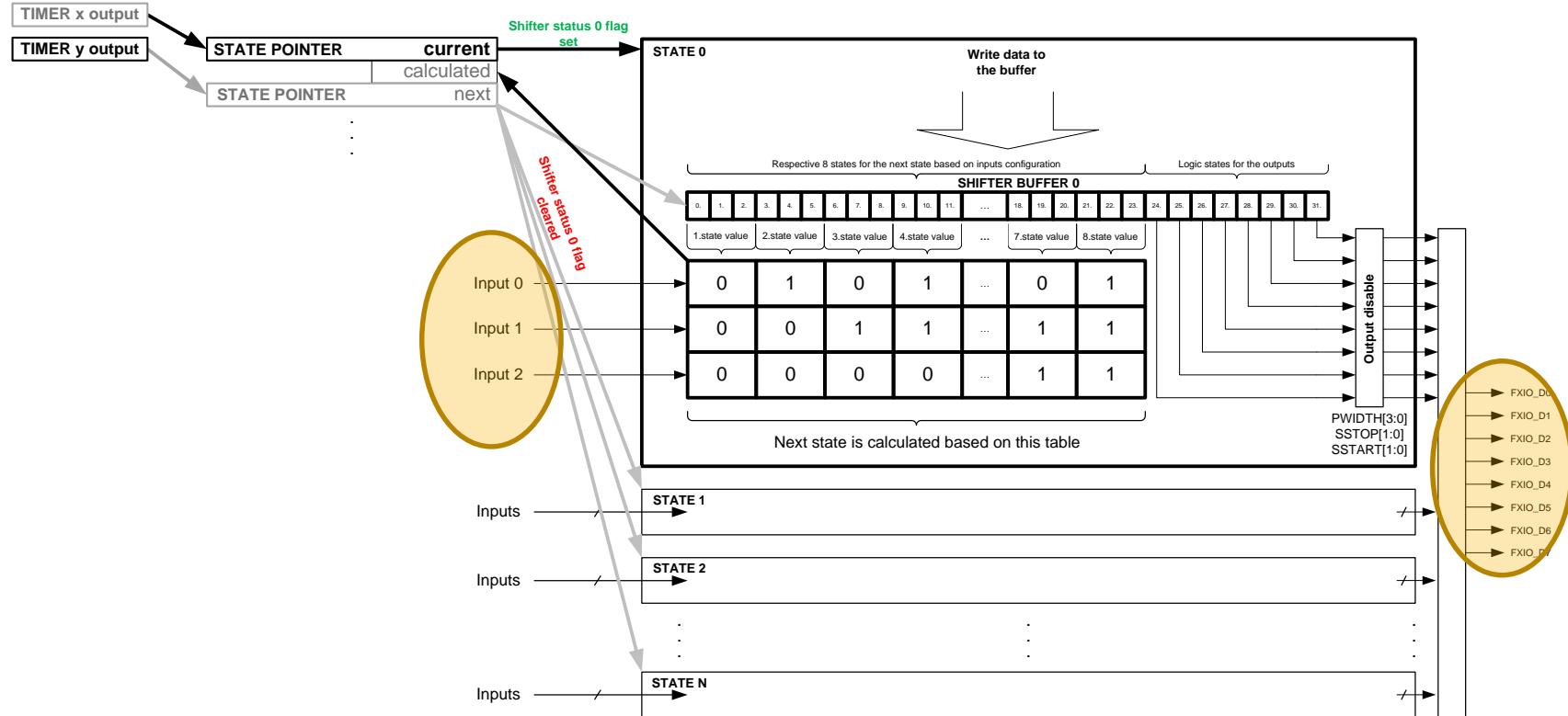
Logical Operators: Logic Mode



- In logic mode 4 FlexIO pins along with the output of one FlexIO shifter can be used to generate a 5 input by 32 output logical look up table
- A shifter holds the output value that will be placed onto the output pin based on the 5 input states
- Note that a FlexIO pin can also be a trigger to a FlexIO timer allowing this function to be expanded

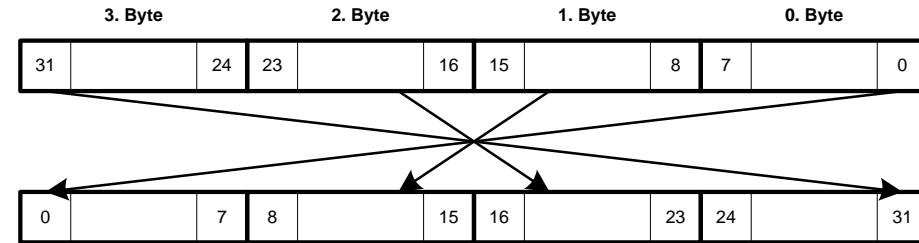
State Machine

Up to 3 input pins setting 8 states controlling up to 8 output pins

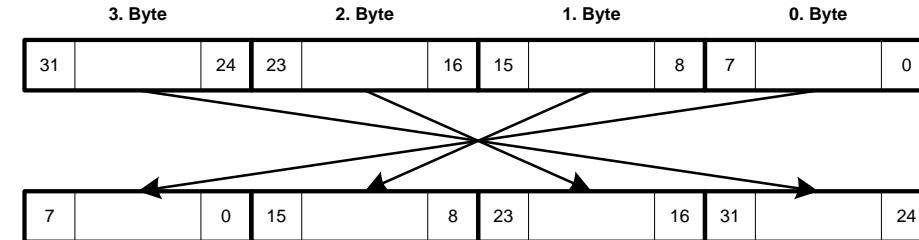


Shifters Buffer Aliases

- SHIFTBUF + BIS → Bit swapped

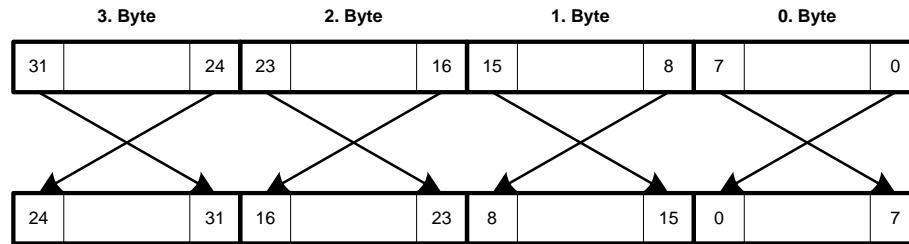


- SHIFTBUF + BYS → Byte swapped

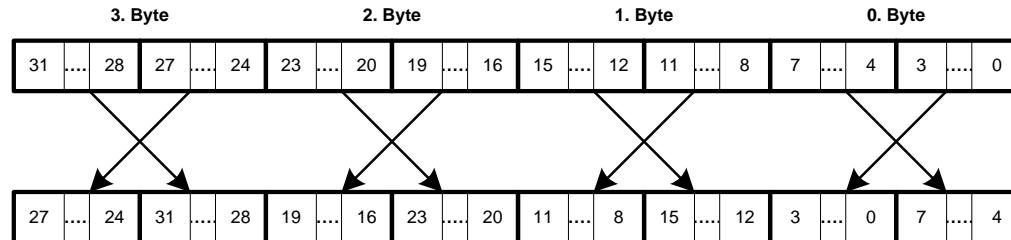


Shifters Buffer Aliases

- SHIFTBUF + BBS → Bit - byte swapped



- SHIFTBUF + NBS → Nibble byte swapped



Flexio Possibilities

- 8 shifters each with individual control and configuration
- 8 timers each with individual control and configuration
- 4 modes of operation
 - Including logic and state machine modes
- Internal and external triggers
- 32 input and output pins
- 1000s of possibilities

FlexIO Appnotes

- [AN4955: Emulating the I2S Bus Master with the FlexIO Module](#)
- [AN5034: Emulating UART by Using FlexIO](#)
- [AN5116: Emulating IRDA by Using FlexIO](#)
- [AN5133: Emulating I2C Bus Master by using FlexIO](#)
- [AN5209: AN5209:Generating PWM by Using FlexIO](#)
- [AN5239: Emulating Hardware State Machine Using FlexIO Module](#)
- [AN5242: Emulating Dual SPI Using FlexIO](#)
- [AN5275: Using FlexIO for parallel Camera Interface](#)
- [AN5280: Using Kinetis FlexIO to drive a Graphical LCD](#)

FLEXIO in SDK 2.0

Overview

The KSDK provides a generic driver for the FlexIO module of Kinetis devices, as well as multiple protocol-specific FlexIO drivers.

Modules

[FlexIO Camera Driver](#)

[FlexIO Driver](#)

[FlexIO I2C Master Driver](#)

[FlexIO I2S Driver](#)

[FlexIO SPI Driver](#)

[FlexIO UART Driver](#)

Start with the Kinetis SDK to leverage examples
and drivers

FlexIO Driver

- [fsl_flexio.h](#)
- [flexio_config_t](#)
- [flexio_timer_config_t](#)
- [flexio_shifter_config_t](#)
- [FSL_FLEXIO_DRIVER_VERSION](#)
- [FLEXIO_TIMER_TRIGGER_SEL_PININPUT](#)
- [flexio_isr_t](#)
- [flexio_timer_trigger_polarity_t](#)
- [flexio_timer_trigger_source_t](#)
- [flexio_pin_config_t](#)
- [flexio_pin_polarity_t](#)
- [flexio_timer_mode_t](#)
- [flexio_timer_output_t](#)
- [flexio_timer_decrement_source_t](#)
- [flexio_timer_reset_condition_t](#)
- [flexio_timer_disable_condition_t](#)
- [flexio_timer_enable_condition_t](#)
- [flexio_timer_stop_bit_condition_t](#)
- [flexio_timer_start_bit_condition_t](#)
- [flexio_shifter_timer_polarity_t](#)
- [flexio_shifter_mode_t](#)
- [flexio_shifter_input_source_t](#)
- [flexio_shifter_stop_bit_t](#)

► [flexio_shifter_start_bit_t](#)

► [flexio_shifter_buffer_type_t](#)

[FLEXIO_GetDefaultConfig](#)

[FLEXIO_Init](#)

[FLEXIO_Deinit](#)

[FLEXIO_Reset](#)

[FLEXIO_Enable](#)

[FLEXIO_SetShifterConfig](#)

[FLEXIO_SetTimerConfig](#)

[FLEXIO_EnableShifterStatusInterrupts](#)

[FLEXIO_DisableShifterStatusInterrupts](#)

[FLEXIO_EnableShifterErrorInterrupts](#)

[FLEXIO_DisableShifterErrorInterrupts](#)

[FLEXIO_EnableTimerStatusInterrupts](#)

[FLEXIO_DisableTimerStatusInterrupts](#)

[FLEXIO_GetShifterStatusFlags](#)

[FLEXIO_ClearShifterStatusFlags](#)

[FLEXIO_GetShifterErrorFlags](#)

[FLEXIO_ClearShifterErrorFlags](#)

[FLEXIO_GetTimerStatusFlags](#)

[FLEXIO_ClearTimerStatusFlags](#)

[FLEXIO_EnableShifterStatusDMA](#)

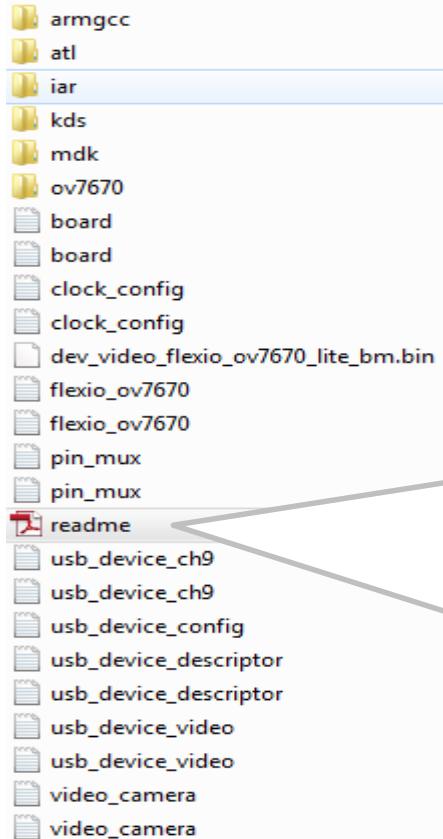
[FLEXIO_GetShifterBufferAddress](#)

[FLEXIO_RegisterHandleIRQ](#)

[FLEXIO_UnregisterHandleIRQ](#)

Example Showcase: FlexIO->USB Video Device

- SDK_2.0_FRDM-K82F.zip\boards\frdmk82f\usb examples\usb device video flexio ov7670



Overview

The USB video FlexIO camera application is a simple demonstration program that uses the KSDK software. It is enumerated as a camera and users can see the video of the device by using a PC test tool.

System Requirement

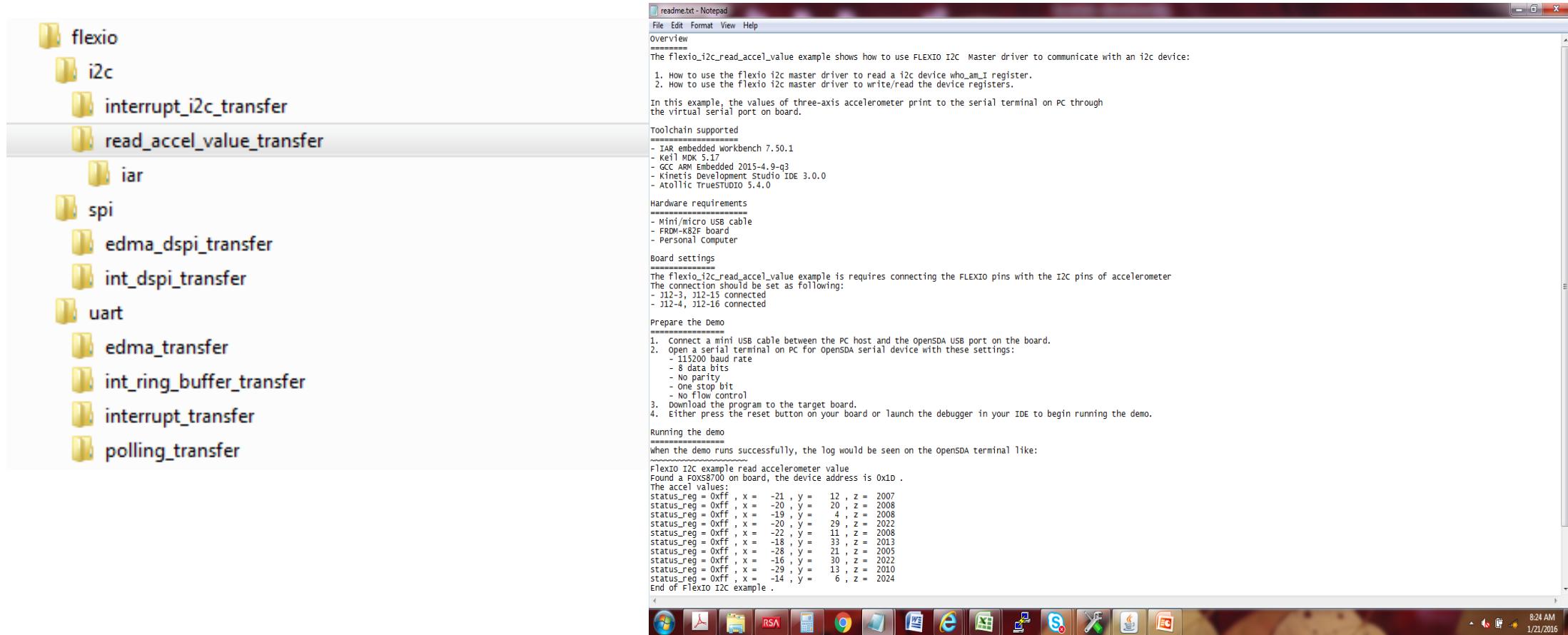
Hardware requirements

- J-Link ARM
- P&E Micro Multi-link universal
- Mini/micro USB cable
- USB A to micro AB cable
- Hardware (tower/base board, OV7670 camera module, ...) for a specific device
- Personal Computer(PC)

Software requirements

Example Showcase: FlexIO->Read Accel Value Transfer

...SDK_2.0_FRDM-K82F\boards\frdmk82f\driver_examples\flexio\





HANDS ON

- **FlexIO Smart I2C Bubble Level**

FlexIO is used to monitor X, Y and Z Axis data. If negative values are detected, the RGB LED is controlled



Building from a KSDK example, FlexIO emulates an I2C master to collect data from the 6 axis sensor

Negative X axis's acceleration turns on **BLUE**

Negative Y axis acceleration turns on **RED**

Negative Z axis acceleration turns on **GREEN**

Read Accel Value Transfer: HW Setup

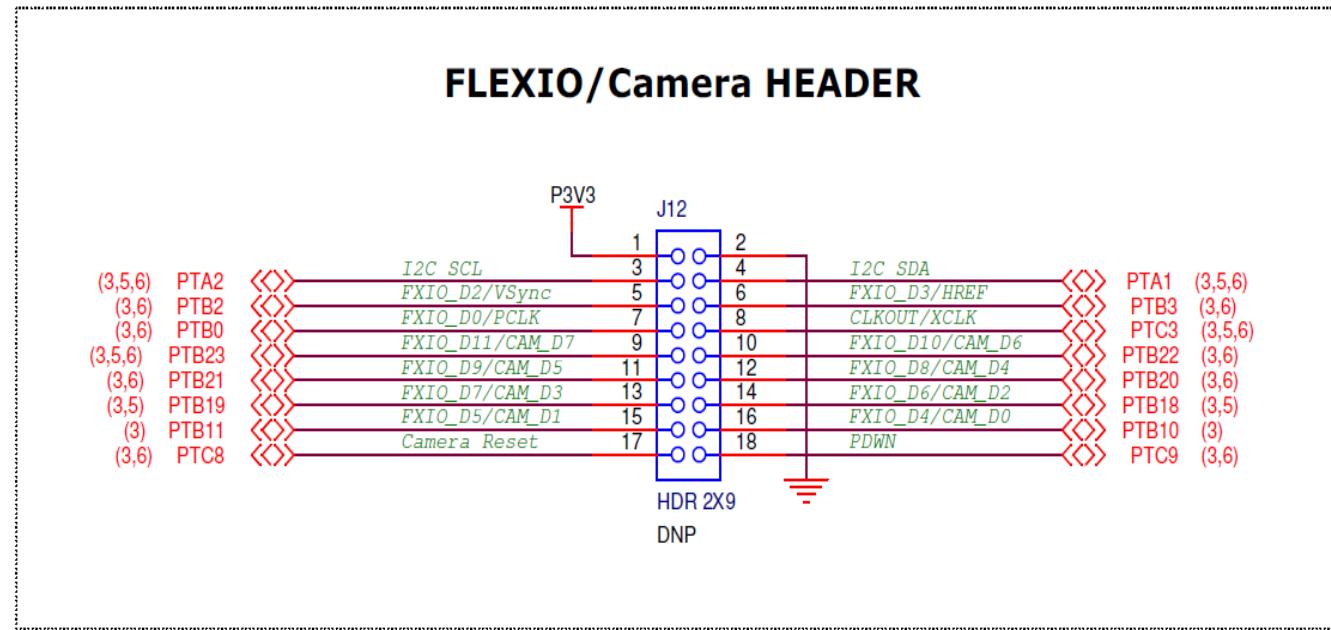
Board settings

=====

The flexio_i2c_read_accel_value example requires connecting the FLEXIO pins with the I²C pins of the accelerometer

The connection should be set as follows:

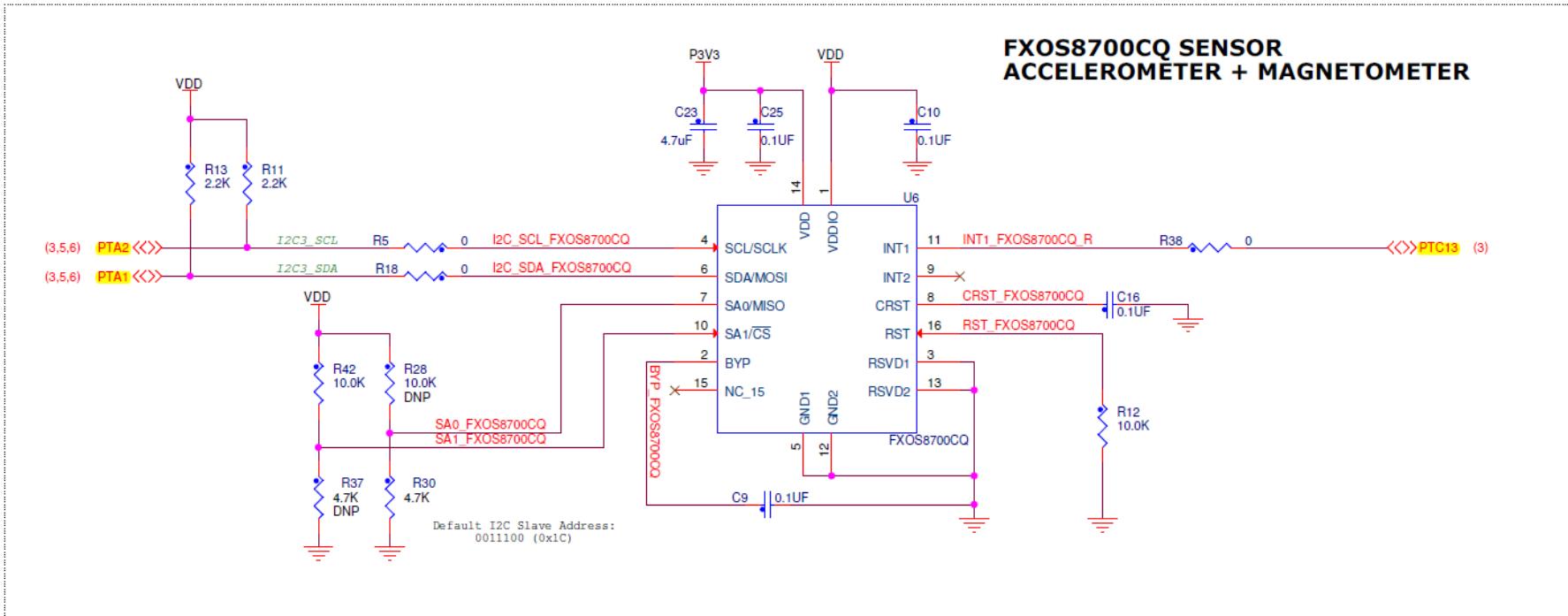
- J12-3, J12-15 connected
- J12-4, J12-16 connected



FXIO0_D5 Connected to PTA2 is the I²C SCL signal connected to FXOS8700

FXIO0_D6 Connected to PTA1 is the I²C SDA signal connected to FXOS8700

Updating the Example Using the Flexibility of FlexIO



FXIO0_D12 MULTIPLEXED to PTA2 is the I2C SCL signal connected to FXOS8700
FXIO0_D11 MULTIPLEXED to PTA1 is the I2C SDA signal connected to FXOS8700

Optimized Example (Less HW Configuration)

```
*****  
 * Definitions  
*****  
#define BOARD_FLEXIO_BASE FLEXIO0  
//#define FLEXIO_I2C_SDA_PIN 4U  
//#define FLEXIO_I2C_SCL_PIN 5U  
  
#define FLEXIO_I2C_SDA_PIN 11U  
#define FLEXIO_I2C_SCL_PIN 12U
```

Board settings

The flexio_i2c_read_accel_value example requires connecting the FLEXIO pins with the I²C pins of the accelerometer

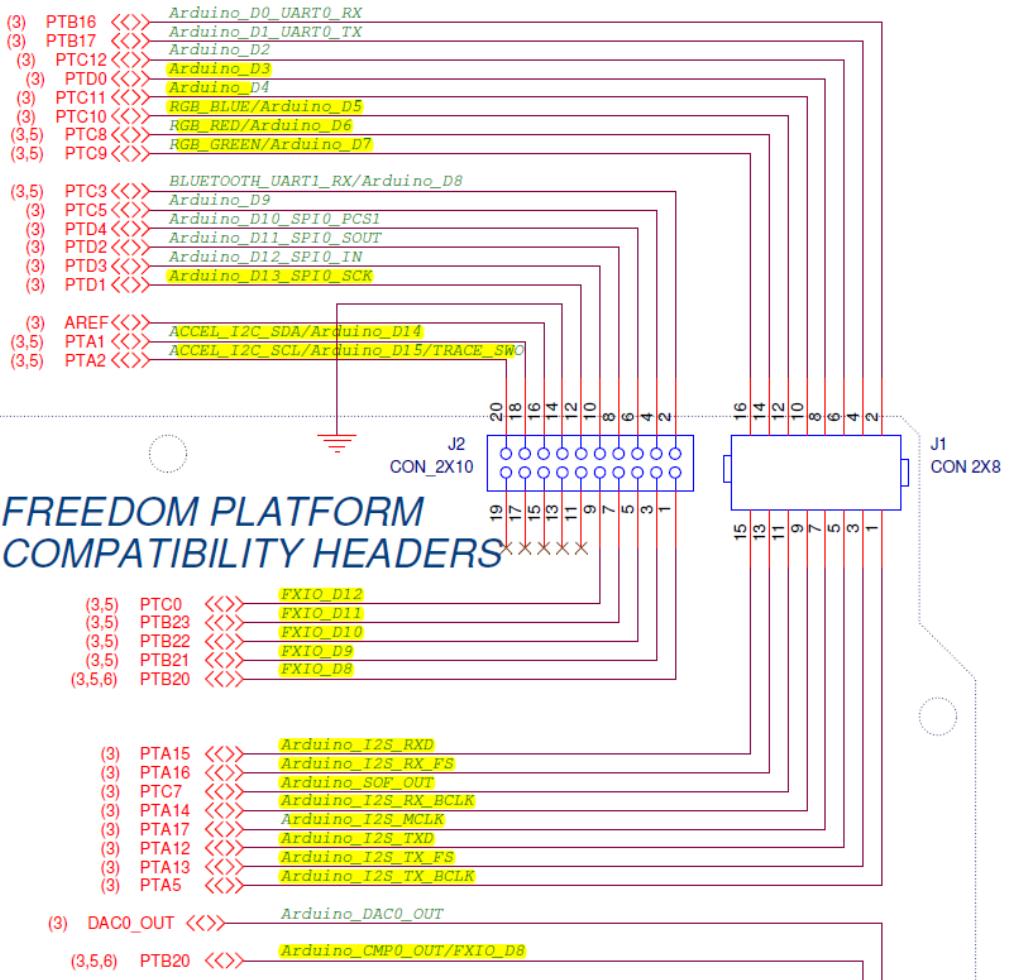
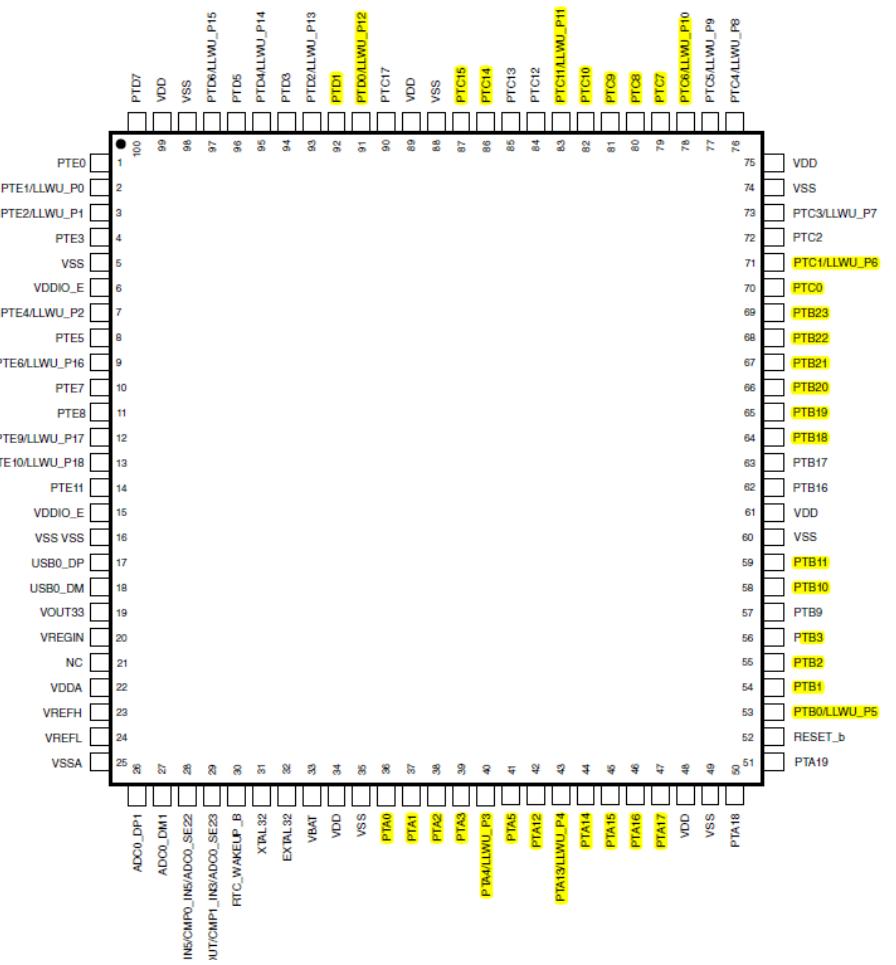
The connection should be set as follows:

- J12-3, J12-15 connected
- J12-4, J12-16 connected

```
CLOCK_EnableClock(kCLOCK_PortA);  
/* Affects PORTA_PCR1 register */  
// PORT_SetPinMux(PORTA, 1U, kPORT_MuxAlt4);  
/* Affects PORTA_PCR2 register */  
// PORT_SetPinMux(PORTA, 2U, kPORT_MuxAlt4);  
PORT_SetPinMux(PORTA, 1U, kPORT_MuxAlt5);  
/* Affects PORTA_PCR2 register */  
PORT_SetPinMux(PORTA, 2U, kPORT_MuxAlt5);
```

Hardware configuration, no longer needed

FlexIO Distribution on FRDM-K82F



Using FlexIO: Flex Your Mind and Make It Smart

I2C Master Emulation

Timer0 generates baud and bit timing

Timer1 contains data bit count

Shifter1 is used for Data (Transmit or receive)

Built from the Kinetis SDK FlexIO driver example in the below directory

SDK_2.0_FRDM-K82F\boards\frdmk82\driver_examples\flexio\i2c\read_accel_value_transfer

Uses the FlexIO I2C Master Driver in order to interface to the on board FXSO8700 6-Axis sensor

Readings are displayed using a UART and a terminal program

Using FlexIO: Flex Your Mind and Make It Smart

I2C Master Emulation

Timer0 generates baud and bit timing

Timer1 contains data bit count

Shifter1 is used for Data (Transmit or receive)

```
/* Do hardware configuration. */
/* 1. Configure the shifter 0 for tx. */
shifterConfig.timerSelect = base->timerIndex[1];
shifterConfig.timerPolarity = kFLEXIO_ShifterTimerPolarityOnPositive;
shifterConfig.pinConfig = kFLEXIO_PinConfigOpenDrainOrBidirection;
shifterConfig.pinSelect = base->SDAPinIndex;
shifterConfig.pinPolarity = kFLEXIO_PinActiveLow;
shifterConfig.shifterMode = kFLEXIO_ShifterModeTransmit;
shifterConfig.inputSource = kFLEXIO_ShifterInputFromPin;
shifterConfig.shifterStop = kFLEXIO_ShifterStopBitHigh;
shifterConfig.shifterStart = kFLEXIO_ShifterStartBitLow;

FLEXIO_SetShifterConfig(base->flexioBase, base->shifterIndex[0], &shifterConfig);

/* 2. Configure the shifter 1 for rx. */
shifterConfig.timerSelect = base->timerIndex[1];
shifterConfig.timerPolarity = kFLEXIO_ShifterTimerPolarityOnNegative;
shifterConfig.pinConfig = kFLEXIO_PinConfigOutputDisabled;
shifterConfig.pinSelect = base->SDAPinIndex;
shifterConfig.pinPolarity = kFLEXIO_PinActiveHigh;
shifterConfig.shifterMode = kFLEXIO_ShifterModeReceive;
shifterConfig.inputSource = kFLEXIO_ShifterInputFromPin;
shifterConfig.shifterStop = kFLEXIO_ShifterStopBitLow;
shifterConfig.shifterStart = kFLEXIO_ShifterStartBitDisabledLoadDataOnEnable;

FLEXIO_SetShifterConfig(base->flexioBase, base->shifterIndex[1], &shifterConfig);
```

Using FlexIO: Flex Your Mind and Make It Smart

I2C Master Emulation

Timer0 generates baud and bit timing

Timer1 contains data bit count

Shifter1 is used for Data (Transmit or receive)

```
/*3. Configure the timer 0 for generating bit clock. */
timerConfig.triggerSelect = FLEXIO_TIMER_TRIGGER_SEL_SHIFTnSTAT(base->shifterIndex[0]);
timerConfig.triggerPolarity = kFLEXIO_TimerTriggerPolarityActiveLow;
timerConfig.triggerSource = kFLEXIO_TimerTriggerSourceInternal;
timerConfig.pinConfig = kFLEXIO_PinConfigOpenDrainOrBidirection;
timerConfig.pinSelect = base->SCLPinIndex;
timerConfig.pinPolarity = kFLEXIO_PinActiveHigh;
timerConfig.timerMode = kFLEXIO_TimerModeDual8BitBaudBit;
timerConfig.timerOutput = kFLEXIO_TimerOutputZeroNotAffectedByReset;
timerConfig.timerDecrement = kFLEXIO_TimerDecSrcOnFlexIOClockShiftTimerOutput;
timerConfig.timerReset = kFLEXIO_TimerResetOnTimerPinEqualToTimerOutput;
timerConfig.timerDisable = kFLEXIO_TimerDisableOnTimerCompare;
timerConfig.timerEnable = kFLEXIO_TimerEnableOnTriggerHigh;
timerConfig.timerStop = kFLEXIO_TimerStopBitEnableOnTimerDisable;
timerConfig.timerStart = kFLEXIO_TimerStartBitEnabled;

/* Set TIMCMP[7:0] = (baud rate divider / 2) - 1. */
timerConfig.timerCompare = (srcClock_Hz / masterConfig->baudRate_Bps) / 2 - 1;

FLEXIO_SetTimerConfig(base->flexioBase, base->timerIndex[0], &timerConfig);

/* 4. Configure the timer 1 for controlling shifters. */
timerConfig.triggerSelect = FLEXIO_TIMER_TRIGGER_SEL_SHIFTnSTAT(base->shifterIndex[0]);
timerConfig.triggerPolarity = kFLEXIO_TimerTriggerPolarityActiveLow;
timerConfig.triggerSource = kFLEXIO_TimerTriggerSourceInternal;
timerConfig.pinConfig = kFLEXIO_PinConfigOutputDisabled;
timerConfig.pinSelect = base->SCLPinIndex;
timerConfig.pinPolarity = kFLEXIO_PinActiveLow;
timerConfig.timerMode = kFLEXIO_TimerModeSingle16Bit;
timerConfig.timerOutput = kFLEXIO_TimerOutputOneNotAffectedByReset;
timerConfig.timerDecrement = kFLEXIO_TimerDecSrcOnPinInputShiftPinInput;
timerConfig.timerReset = kFLEXIO_TimerResetNever;
timerConfig.timerDisable = kFLEXIO_TimerDisableOnPreTimerDisable;
timerConfig.timerEnable = kFLEXIO_TimerEnableOnPrevTimerEnable;
timerConfig.timerStop = kFLEXIO_TimerStopBitEnableOnTimerCompare;
timerConfig.timerStart = kFLEXIO_TimerStartBitEnabled;

/* Set TIMCMP[15:0] = (number of bits x 2) - 1. */
timerConfig.timerCompare = 8 * 2 - 1;

FLEXIO_SetTimerConfig(base->flexioBase, base->timerIndex[1], &timerConfig);
```

Using FlexIO: Flex Your Mind and Make It Smart

I2C Master Emulation

Timer0 generates baud and bit timing

Timer1 contains data bit count

Shifter1 is used for Data (Transmit or receive)

I2C Sniffing

Timer[5:7] is used for setting shift count

Shifter7 is used for Z data matching

Shifter6 is used for Y data matching

Shifter5 is used for X data matching

Building from the example, other FlexIO resources can be used to monitor specific data within the I2C communications with the 6-Axis Sensor

Using FlexIO: Flex Your Mind and Make It Smart

I²C Master Emulation

Timer0 generates baud and bit timing

Timer1 contains data bit count

Shifter1 is used for Data (Transmit or receive)

I²C Sniffing

Timer[5:7] is used for setting shift count

Shifter7 is used for Z data matching

Shifter6 is used for Y data matching

Shifter5 is used for X data matching

RGB LED Control

Timer2 is used for Red LED (Waiting for Shifter6 trigger)

Timer4 is used for Blue LED (Waiting for Shifter5 trigger)

Timer3 is used for Green LED (Waiting for Shifter7 trigger)

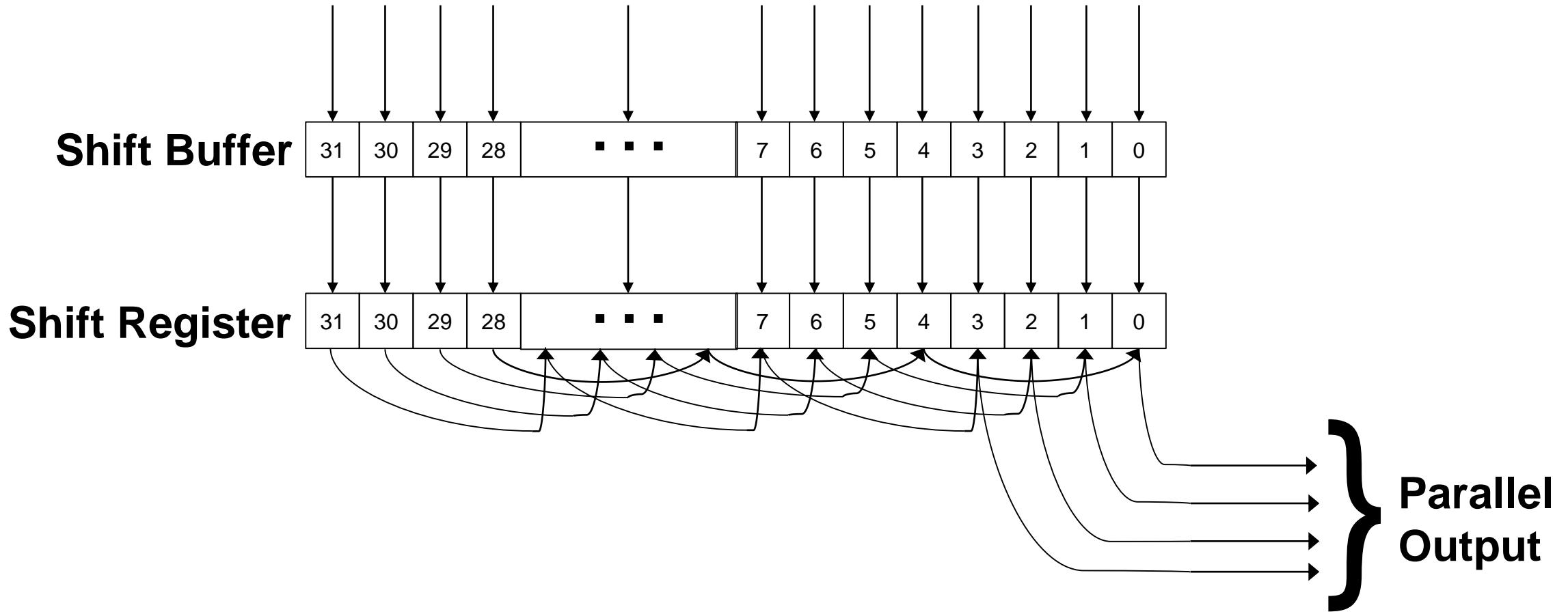
Hands On

Parallel Mode

FlexIO Parallel Mode Features

- Parallel Mode supported in FlexIO v1.1
- Supports transmit and/or receive in parallel
- Bus widths of 4, 8, 16, or 32
- Possible Applications or interfaces:
 - Graphical LCD
 - Camera interface
 - Motorola 68K bus
 - Intel 8080 bus
 - Etc.

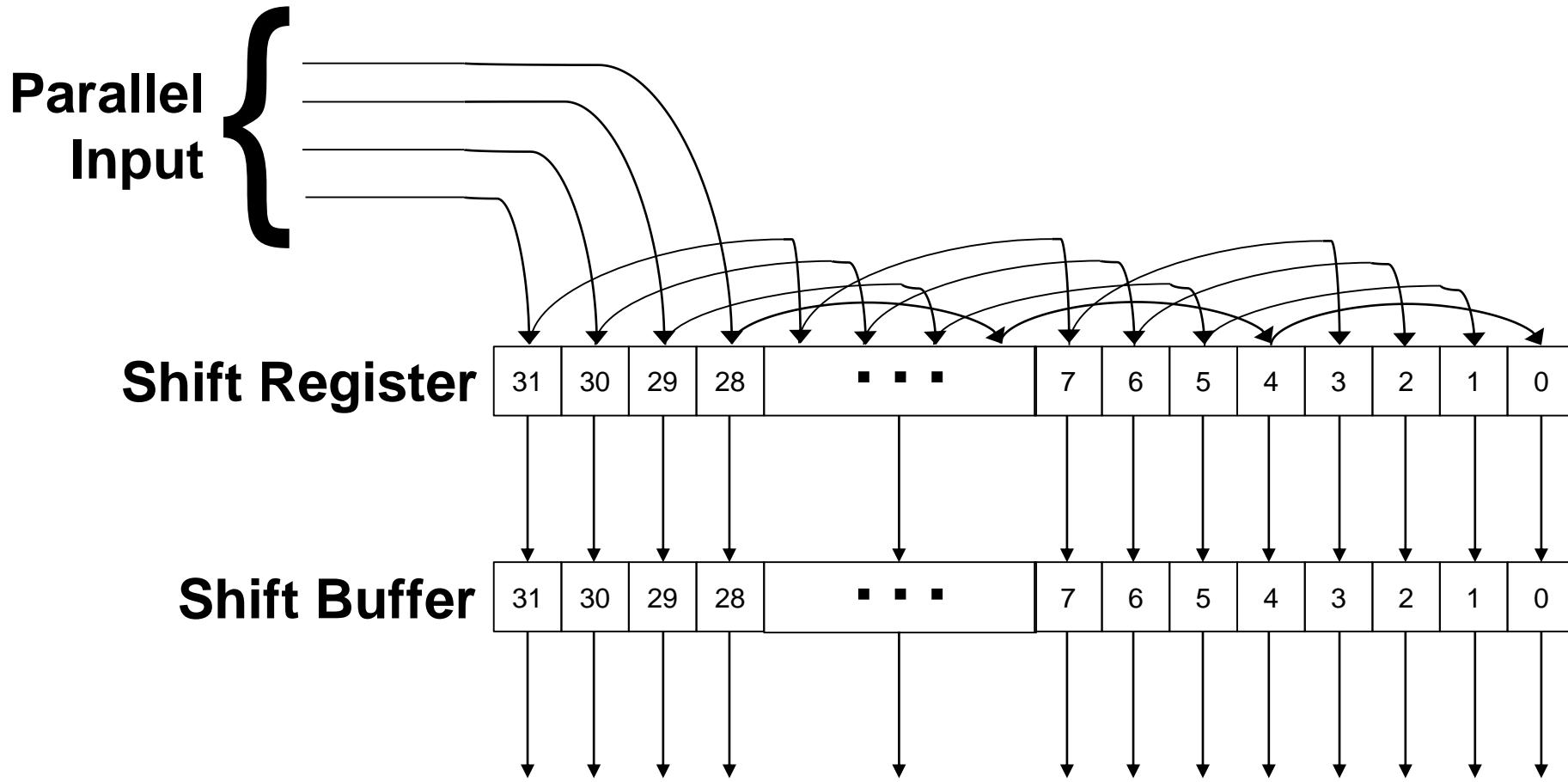
Parallel Transmit: 4-bit Bus Example



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

NOTE 2: Shifters can be chained together

Parallel Receive: 4-bit Bus Example



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

NOTE 2: Shifters can be chained together

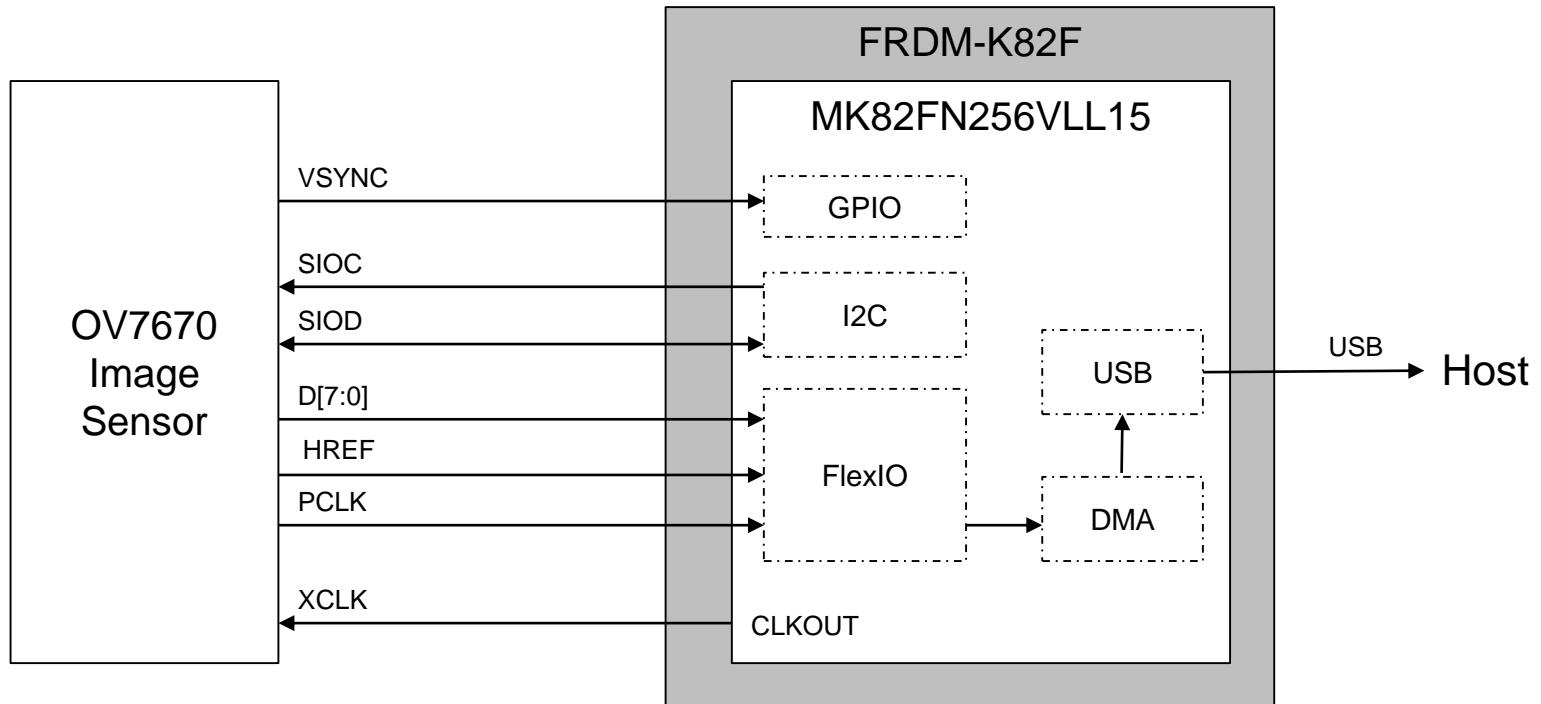
FlexIO Module Differences

FlexIO module		KL17 / KL27 / KL33 / KL43 (48MHz)	KL81Z128 / KL82Z128 (72MHz)	K80FN256 / K81FN256 (150MHz)
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
	FLEXIOx_PIN	-	x	x
Registers	FLEXIOx_SHIFTSTATE	-	x	x
	FLEXIOx_SHIFTBUFNBSn	-	x	x
	FLEXIOx_SHIFTBUFHWSn	-	x	x
	FLEXIOx_SHIFTBUFNISn	-	x	x
	Triggers	16	16	16
Pins		8	32	32
Timers		4	8	8
Shifters		4	8	8

FlexIO CMOS Image Sensor Interface

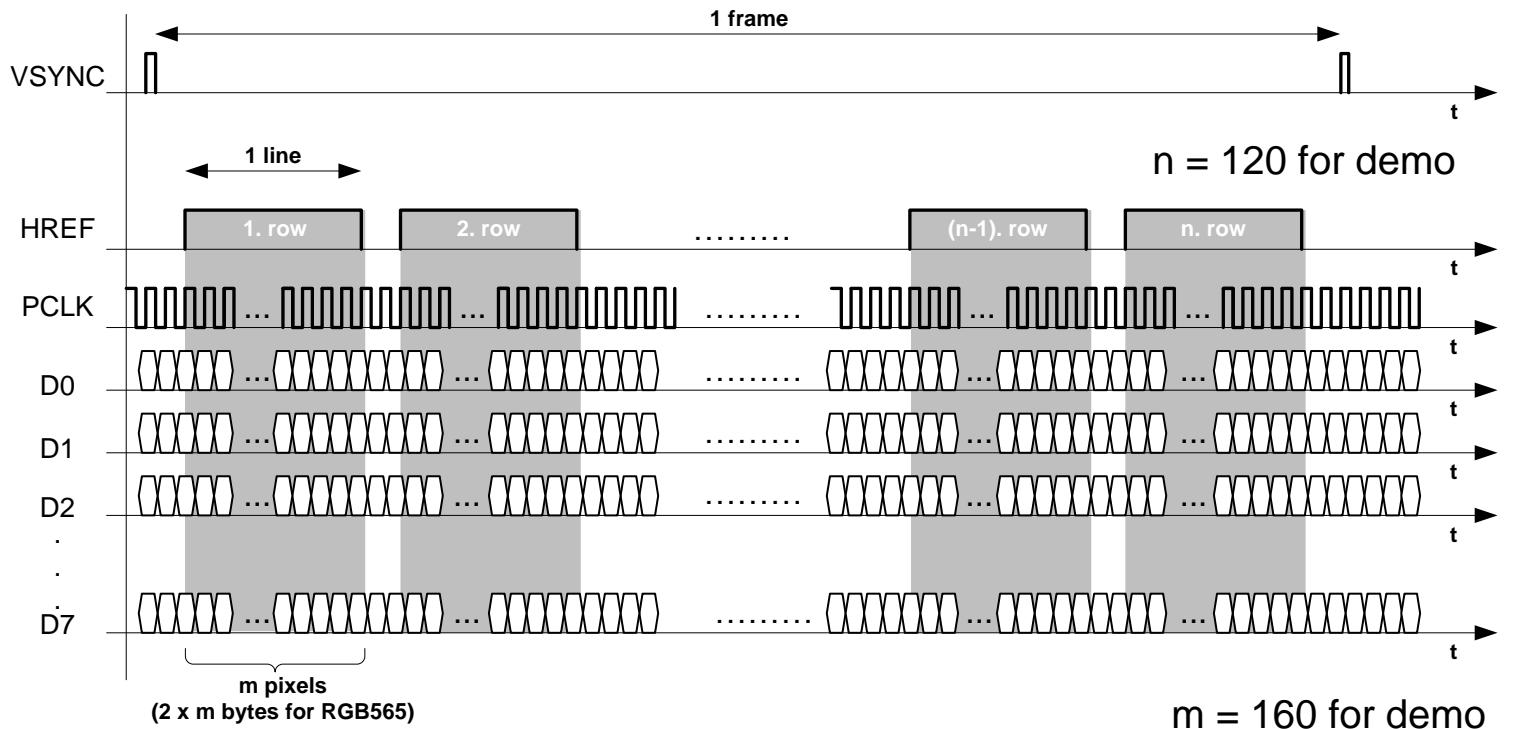
KSDK USB Camera OV7670 FlexIO Demo

- Image frames sent using USB Video Class
- Configures OV7670 for 160 x 120 resolution, 16bits/pixel
- Shifts 8-bits in parallel per PCLK using FlexIO
- FlexIO triggers DMA after frame received, moves to SRAM buffer for USB



USB Camera OV7670 FlexIO Timing

- FlexIO Timer0 trigger is HREF
 - when HREF is high, timer is enabled
- FlexIO Timer0 input is PCLK
 - rising edge on PCLK is shift clock – pixel data is sampled
- VSYNC uses GPIO interrupt to reset DMA for next frame



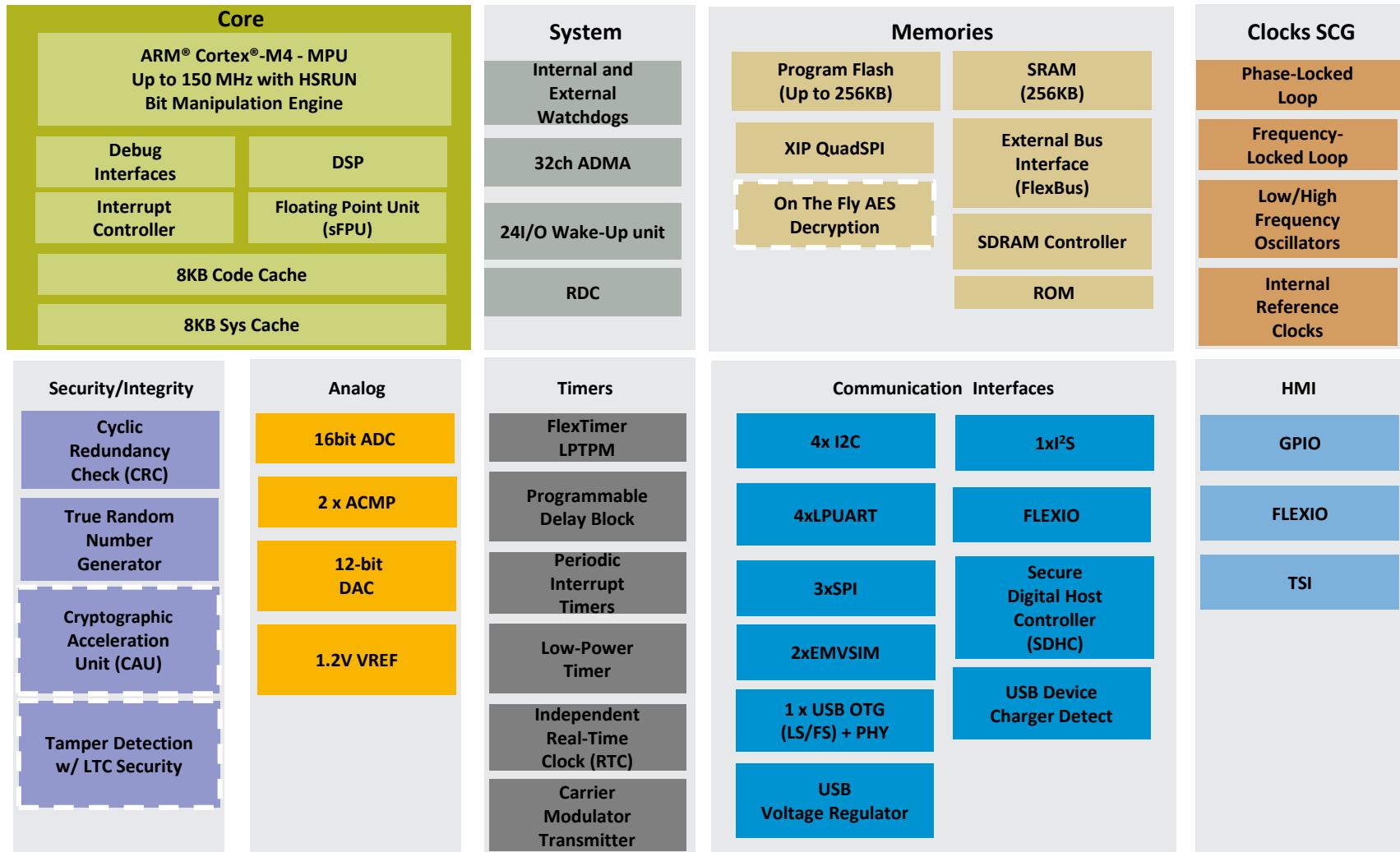
KSDK USB Camera OV7670 FlexIO Demo

- Example included in [KSDK](#), located at:
 - SDK_2.0_FRDM-
 - K82F.zip\boards\frdmk82f\usb_examples\usb_device_video_flexio_ov7670



Driving Graphical LCD with FlexIO

Using Kinetis K8x MCU to Drive Graphical LCD with FlexIO

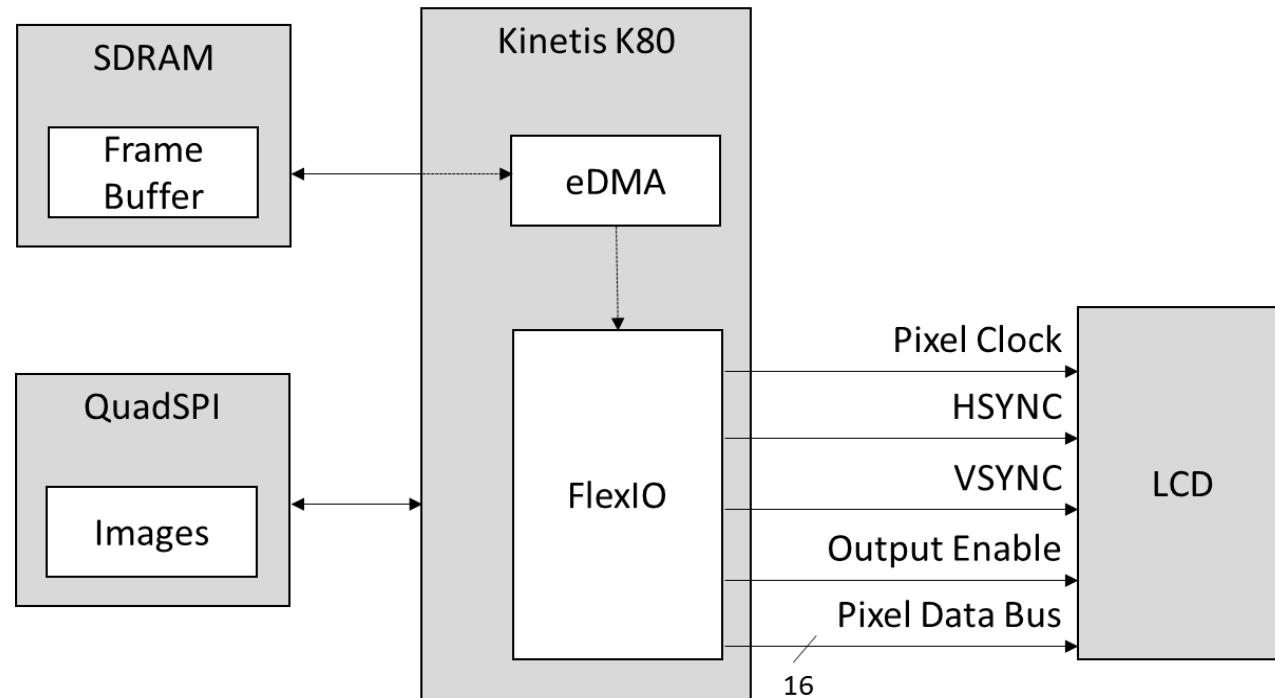


* Dashed Line Represents Optional Features

- FlexIO v1.1 with Parallel mode
- SDRAM Controller for large frame buffers
- QuadSPI Flash interface to store graphics
- High-Performance 150MHz CPU with large caches

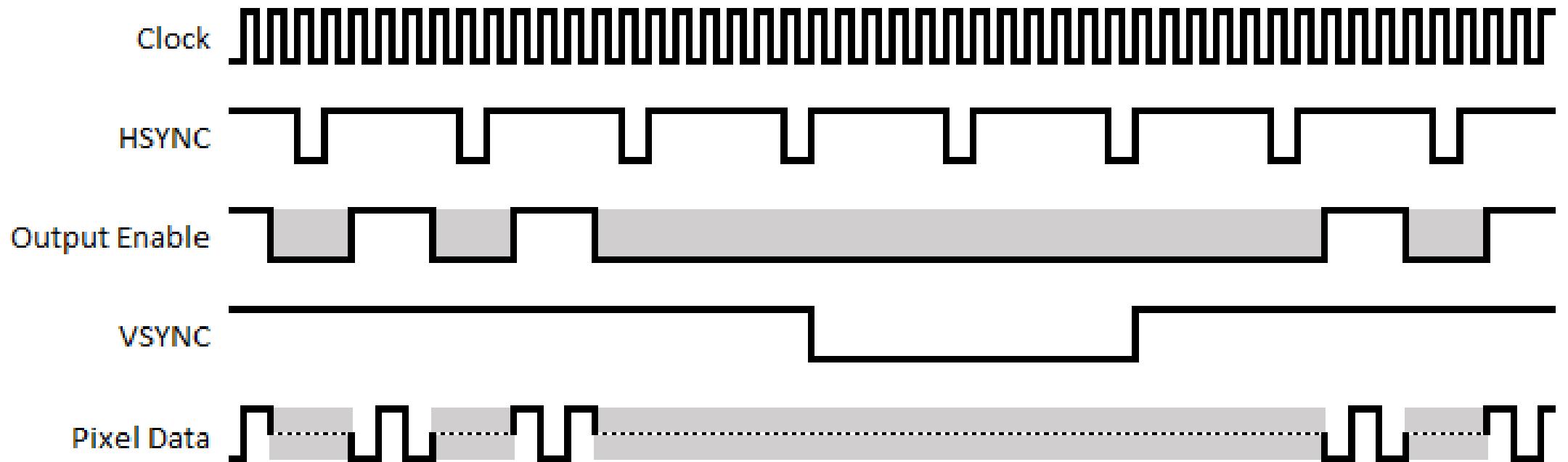
FlexIO to Graphical LCD

- FlexIO in Parallel mode for pixel data
- FlexIO timers can also drive HSYNC, VSYNC, and pixel clock
- For larger displays, frame buffer in external SDRAM
- FlexIO triggers DMA to pull data from frame buffer
- QuadSPI to store graphic images



LCD Signal Generation With FlexIO

Simplified timing diagram



FlexIO LCD Demo

Appnote [AN5280](#) available and software [AN5280SW](#)





SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, CoolFlux, EMBRACE, GREENCHIP, HITAG, I2C BUS,ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Plus, MIFARE FleX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TrenchMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. © 2015–2016 NXP B.V.