# INTRODUCTION TO FLEXIO

Osvaldo Romero
Applications Engineer

# Agenda

- Introduction to FlexIO
- FlexIO Main Features
- FlexIO Applications
- NXP Products with FlexIO
- Collaterals\Tools for FlexIO
- FlexIO Demo
- Summary and Q&A

# Session Objective

- Lear what is the FlexIO
- Learn where the FlexIO module can be use.
- Know the NXP products that include FlexIO
- Show FlexIO in action

# Introduction

# Scenario

- Your application needs more UART modules?
- Do you need more PWM channels for your application?
- Need more SPI modules ?
- Need an I2C module ?

FlexIO can help you!

# What is FlexIO

- “FlexIO” - **Flexible input and output peripheral**
- Highly configurable module providing a wide range of functionality including:

  -Emulation of a variety of  communication protocols: UART, I2C, SPI, I2S, etc

  -Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions

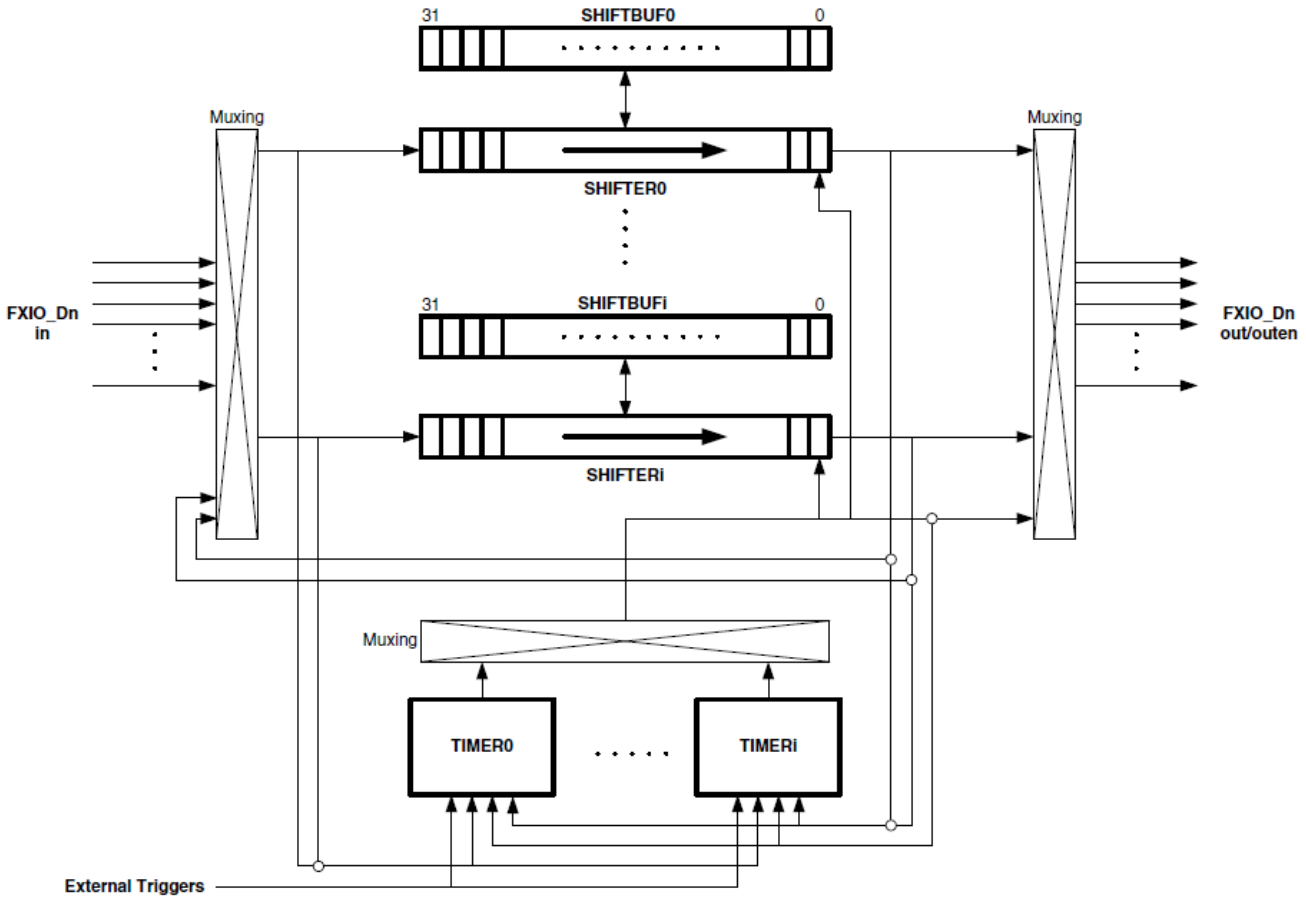
  -Programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules

  -Programmable state machine for offloading basic system control functions from CPU

- The FlexIO peripheral was initially introduced on the NXP Kinetis KL43 family

# FlexIO

- FlexIO main components
- Shifter
- Timer
- Pin



# Agenda

- Introduction to FlexIO
- FlexIO Main Features
- FlexIO Applications
- NXP Products with FlexIO
- Collaterals\Tools for FlexIO
- FlexIO Demo
- Summary and Q&A

# FlexIO Features

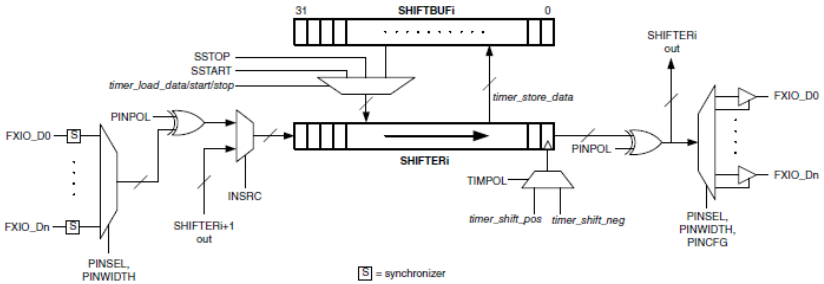
# FlexIO Features

- Allows emulation of standard communication interfaces.
- Supports a wide range of protocols and peripherals including:
  - UART
  - I2C
  - SPI
  - I2S
  - LCD RGB
  - CMT (carrier modulator transmitter)
  - PWM/waveform generation
  - SWD (single wire debug)
  - LIN

- Creates an interlink between GPIO method of software emulation and exact hardware peripheral module.

# FlexIO: Shifter Features

- Array of 32-bit shift registers with transmit, receive and data match modes
- Responsible for buffering and shifting data into or out of the FlexIO
- Each shifter has a timer that control the timing shift, load and store events.
- Support DMA interrupt or polled operation.
- Receive/transmit/match store/match continuous modes support
- Input source – pin or shifter N+1 output
- Start/Stop bit generation support
- Pin selection for shifter (with polarity option)
- Timer selection for timing of shift/load/store events (positive/negative edge option)
- Shifter buffer [illegible] t-byte swap



# FlexIO: Shifter operation

- Transmit mode

The shifter will load data from the buffer and shift data out when a load event is signaled by the assigned Timer.

- Receive Mode

The shifter will shift data in and store data into the SHIFTBUF register when a store event is signalled by the assigned Timer.

- Match Store Mode

Data is shifted  in, check for a match result and store matched data into the buffer register when a store event is signalled by the assigned Timer.

Up to 16 bits of data can be compared. When match occurs  a flag will set and any enabled interrutps or DMA requests

Flag will be cleared when reading buffer.

- Match Continuous

Data is shifted and continuously check for a match result whenever a shift event is signalled by the assigned Timer.

# FlexIO: Shifter operations

- State Mode

Implement any state machine with up to 8 states, 8 outputs and 3 selectable inputs per state.

Basic control functions to be offloaded from the CPU using FlexIO hardware

- Logic Mode

Implement a small amount of programmable digital logic within a FlexIO Shifter.

# FlexIO: Timer Operation

- Highly flexible 16-bit timers with support for a variety of internal or external trigger,reset, enable and disable conditions
- 16-bit timers control the loading, shifting and storing of the shift registers, the counters load the contents of the compare register and decrement down to zero on the FlexIO clock
- Can generate a PWM waveform,
- Each timer operates independently.
- Can be configured to enable or disable at the same time as the previous timer.
- Timer output can trigger any other timer.
- 8 bit PWM mode

# FlexIO: Pin Operation

- Each timer and shifter can be configured to use any FlexIO pin as:
  - Input
  - Output data
  - Output enable (Open Drain)
  - Bidirectional output
  - Shifters can also be configured to use multiple FlexIO pins in parallel.
    - 4,8,16 or bits can be shifted on every Shift clock.
    - Adjacent shifter can be selected as the input source. The least significant 4, 8, 16 or 32-bits from the adjacent shifter will be sampled on Shift clock.

# Agenda

- Introduction to FlexIO
- FlexIO Main Features
- FlexIO Applications
- NXP Products with FlexIO
- Collaterals\Tools for FlexIO
- FlexIO Demo
- Summary and Q&A

# FlexIO applications

# UART application

- Two independent parts **Transmit** & **Receive**
  - Transmit is supported by using one Timer, one Shifter and one Pin
  - Receive part 1xTimer, 1x Shifter and one Pin
  - The start and stop bit insertion is handled automatically

- Allows polling and interrupt mode
- Break and idle characters require software intervention
- Configurable bit order (bit swapped buff MSB first)
- Multiple transfers can be supported using DMA controller
- Does not support automatic insertion of parity bits

# SPI Master

- Supported using:
  - Two timers
  - Two shifters
  - Four pins

  - CPHA=0 and CPHA=1 can be supported.
  - Writing to the transmit buffer by either core or DMA is used to initiate each transfer.
  - Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

# SPI Slave

- Supported using:
- One Timer
- Two Shifters
- Four pins

- CPHA=0 and CPHA=1 supported

- Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

# I2C Master

- Supported using:
- Two timers
- Two shifters
- Two pins

- FlexIO inserts a stop bit after every word to generate/verify the ACK/NACK
- Data transfers can be supported using the DMA controller
- Due to synchronization delays, the data valid time for the transmit output is 2 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

# I2S Master

- Supported using
  - Two timers
  - Two Shifters
  - Four pins
- Data transfers can be supported using the DMA
- Due to synchronization delays, the setup time for the receiver input is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

# I1S Slave

- Supported using
- Two timers
- Two shifters
- Four pins

- Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

# Agenda

- Introduction to FlexIO
- FlexIO Main Features
- FlexIO Applications
- NXP Products with FlexIO
- Collaterals\Tools for FlexIO
- FlexIO Demo
- Summary and Q&A

# FlexIO in NXP products

# NXP Families with FlexIO

- KL1x
  - MKL17Z128
  - MKL17Z256
- KL2x
  - MKL27Z128
  - MKL27Z256
- KL3x
  - MKL33Z128
  - MKL33Z256
- KL4X
  - MKL43Z128
  - MKL43Z256

# Agenda

- Introduction to FlexIO
- FlexIO Main Features
- FlexIO Applications
- NXP Products with FlexIO
- Collaterals\Tools for FlexIO
- FlexIO Demo
- Summary and Q&A

# FlexIO Demo

# FlexIO Demo: UART using KL43

- It is used PTE22(FlexIO_D6) and  PTE23 (FlexIO_D7) of KL43
- Pins are routed to Tx/Rx OpenSDA interface

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| G2 | 11 | PTE22 | ADC0_DP3/ ADC0_SE3 | ADC0_DP3/ ADC0_SE3 | PTE22 | | TPM2_CH0 | UART2_TX | | FXIO0_D6 | | |
| F2 | 12 | PTE23 | ADC0_DM3/ ADC0_SE7a | ADC0_DM3/ ADC0_SE7a | PTE23 | | TPM2_CH1 | UART2_RX | | FXIO0_D7 | | |

# UART use case

- Two independent parts **Transmit** & **Receive**
  - Transmit is supported by using one Timer, one Shifter and one Pin
  - Receive part 1xTimer, 1x Shifter and one Pin
  - The start and stop bit insertion is handled automatically

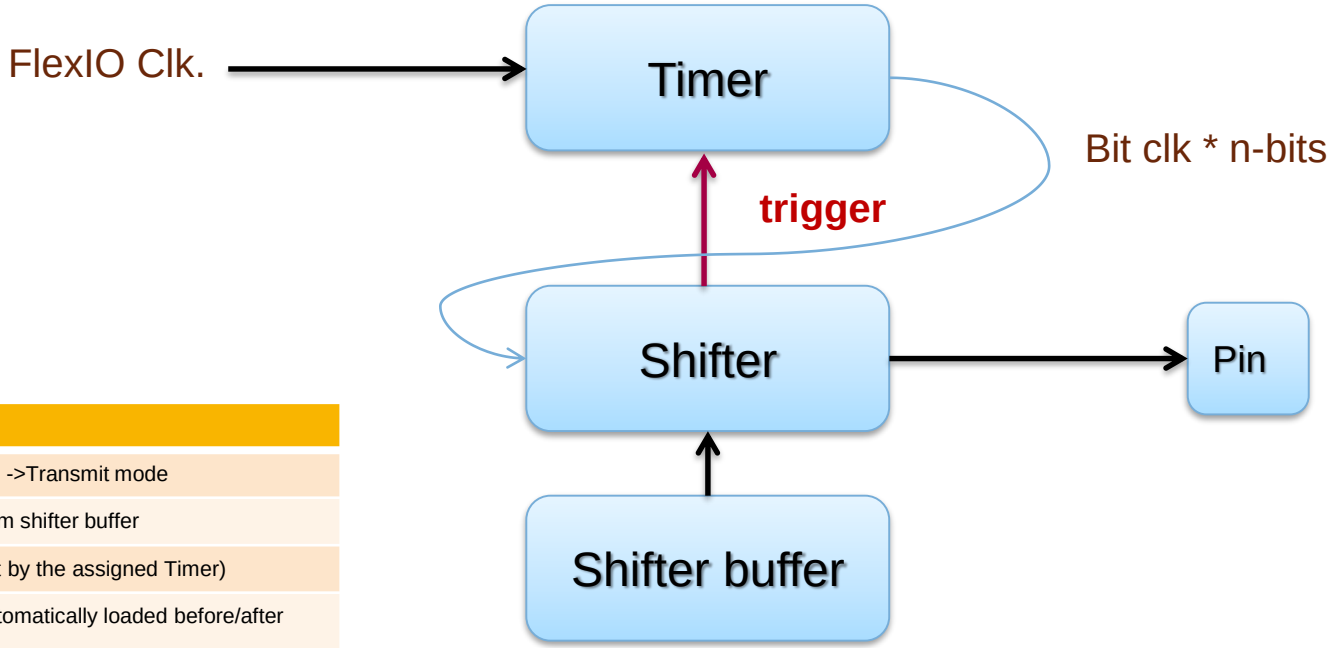
- Allows polling and interrupt mode
- Break and idle characters require software intervention
- Configurable bit order (bit swapped buff MSB first)
- Multiple transfers can be supported using DMA controller
- Does not support automatic insertion of parity bits

# FlexIO UART transmit mode

**UART Transmit procedure:**

- Shifter configuration ->Transmit mode
- Shifter load data from shifter buffer
- Shift the data (event by the assigned Timer)
- Start/stop bit are automatically loaded before/after data is loaded
- Using timer status flag for sending next data
- Difference between polling vs. interrupt mode
  - Checking the status flag (polling)
  - Timer status flag generate interrupt (interrupt)

# Block diagram of the UART transmitter on FlexIO



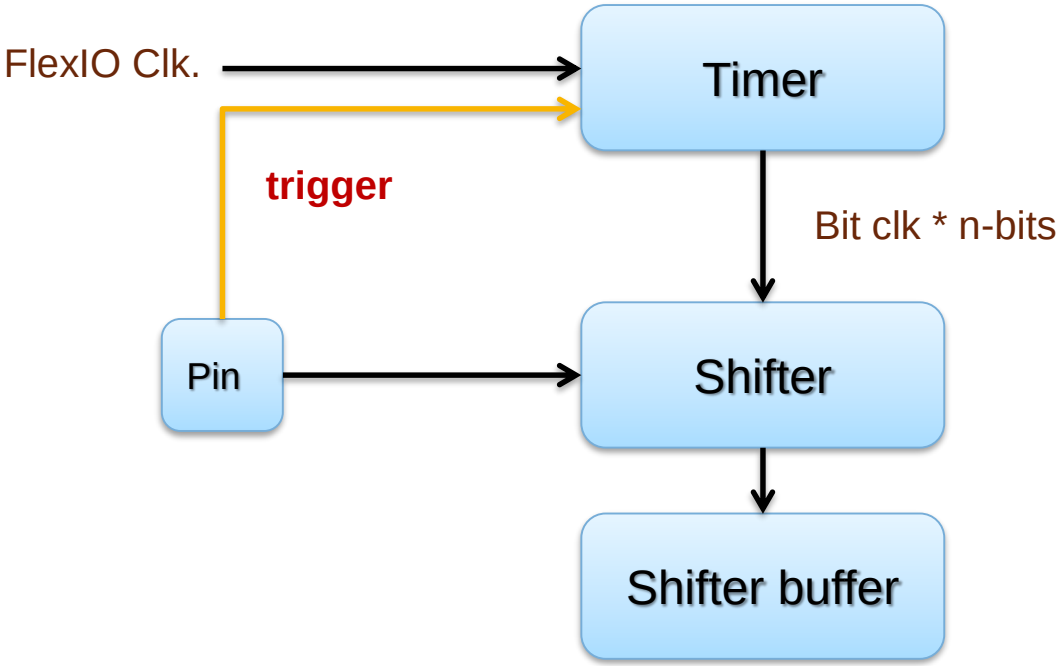

| UART Transmit procedure: |
| --- |
| Shifter configuration ->Transmit mode |
| Shifter load data from shifter buffer |
| Shift the data (event by the assigned Timer) |
| Start/stop bit are automatically loaded before/after data is loaded |
| Using timer status flag for sending next data |
| Difference between polling vs. interrupt mode<br>• Checking the status flag (polling)<br>• Timer status flag generate interrupt (interrupt) |

# FlexIO UART receive mode

| UART Receive procedure: |
| --- |
| Shifter configuration -> Receive mode |
| Shifter shifts data in when the store event is signaled |
| Status flag to indicate when data can be read (generate interrupt) |
| Waiting for shifter status flag in polling mode |
| Store into the shifter buffer |
| Reading bit swapped shifter buffer (without any logical operation) |
| |
| |

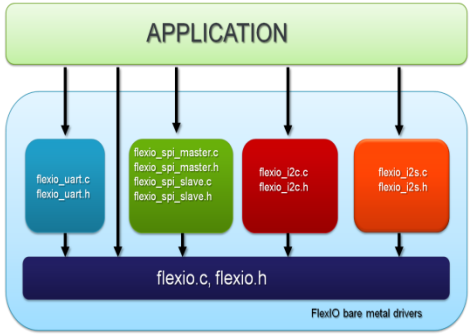
# Block diagram of the UART receiver on FlexIO

| UART Receive procedure: |
| --- |
| Shifter configuration -> Receive mode |
| Shifter shifts data in when the store event is signaled |
| Status flag to indicate when data can be read (generate interrupt) |
| Waiting for shifter status flag in polling mode |
| Store into the shifter buffer |
| Reading bit swapped shifter buffer (without any logical operation) |

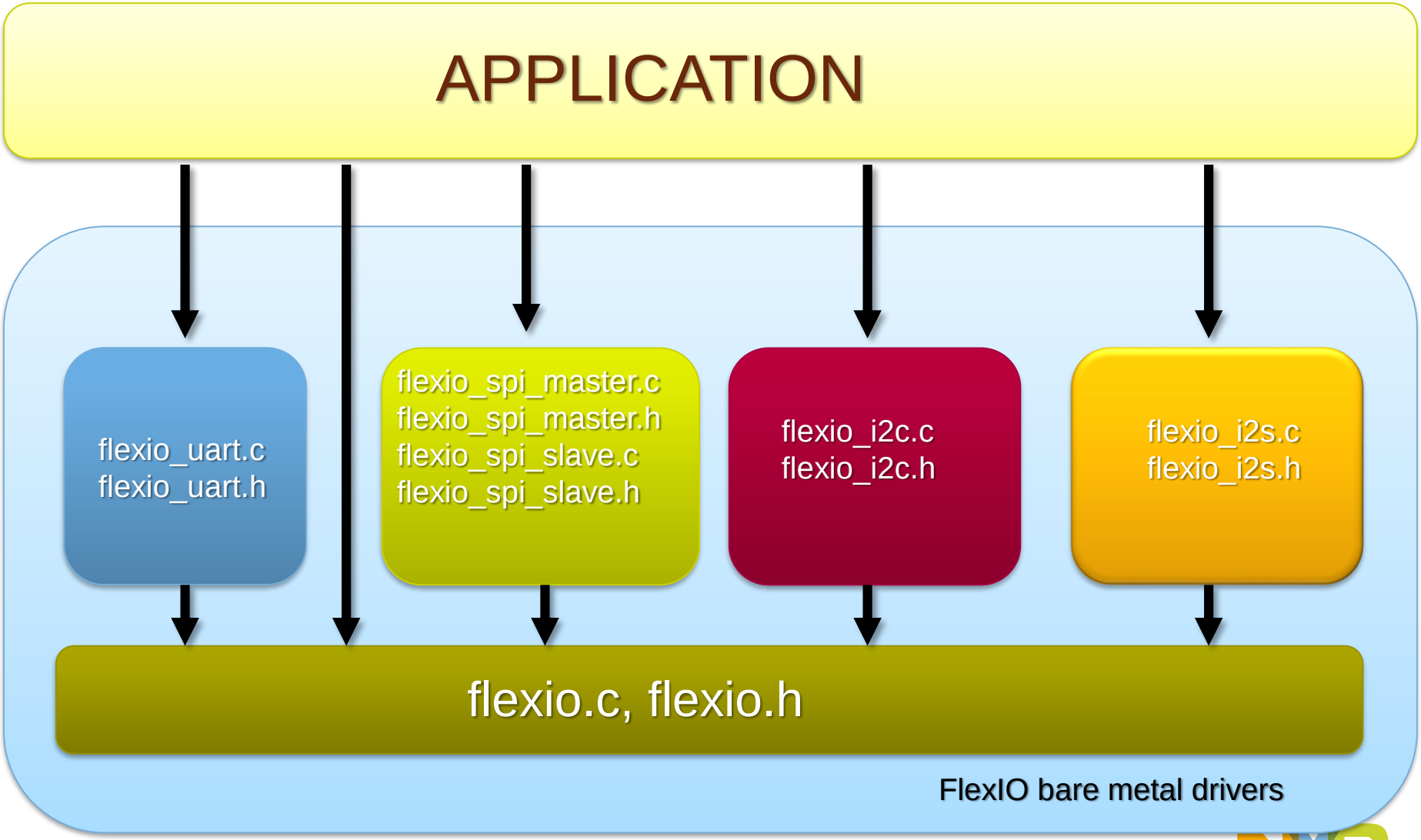


# FlexIO bare metal drivers

- Driver is divided into two parts
- Main low level driver
  - Initialization of peripheral
  - Basic setting of timers and shifters
  - Writing and reading to the register
- Sub drivers for each use case
  - Focused on emulation of the peripheral (SPI, UART …)
  - Allow using more instances of one driver
- Application can use both of driver layers
- Callback function implemented



# FlexIO bare metal drivers



# Agenda

- Introduction to FlexIO
- FlexIO Main Features
- FlexIO Applications
- NXP Products with FlexIO
- Collaterals\Tools for FlexIO
- FlexIO Demo
- Summary and Q&A

# Summary

# Summary

- FlexIO  can help you emulate
  -UART
  -I2C
  -SPI
  -I2S
  -PWM
- FlexIO Module consist on three parts: shifter, timer and I/O.
- MCUs with FlexIO are:

- Start you development with FlexIO using

# References

- AN5034: Emulating UART by Using FlexIO
- AN4955: Emulating the I2S Bus Master with the FlexIO Module

SECURE CONNECTIONS
FOR A SMARTER WORLD