

# Emulating UART by Using FlexIO

*by Pavel Kreněk, Application Engineer*

*Freescale Roznov, Czech Republic*

## 1 Introduction

This application note shows how to use the universal peripheral module FlexIO for emulating the UART bus. The FlexIO peripheral was initially introduced on the Freescale Kinetis KL43 family.

FlexIO is a highly configurable module capable of emulating a wide range of different communication protocols: UART, I<sup>2</sup>C, SPI, I<sup>2</sup>S, etc.

Standalone peripheral module FlexIO is not used for replacement of the UART peripheral, but as an additional peripheral module of the MCU. The important feature of this peripheral is that it enables the user to build their own peripheral directly in the MCU.

This use case for the UART module creates a simple software driver based on the independent receiver and transceiver. For this demonstration, we have used the Freescale Tower system. The maximum tested baud rate of the emulated UART bus is 115200 baud.

### Contents

1. Introduction .....	1
2. Main features of the FlexIO peripheral module .....	2
3. Required hardware .....	2
4. UART in general .....	3
5. UART emulation by FlexIO module .....	3
6. Software implementation .....	5
7. Conclusion .....	8
8. References .....	8

## 2 Main features of the FlexIO peripheral module

- FlexIO means flexible input and output peripheral.
- Highly configurable module with wide range functionality.
- Allows emulation of standard communication interfaces.
- Supports a wide range of protocols and peripherals including:
  - UART
  - I<sup>2</sup>C
  - SPI
  - I<sup>2</sup>S
  - LCD RGB
  - CMT (carrier modulator transmitter)
  - PWM/waveform generation
  - SWD (single wire debug)
- Creates an interlink between GPIO method of software emulation and exact hardware peripheral module.

## 3 Required hardware

This document describes the example application based on the Freescale Tower system. The basic concept can be easily implemented on the customized hardware as well.

The application can be easily set up using the following Tower system boards:

- TWR-KL43Z48M
- TWR-ELEV (primary and secondary)
- Optionally, TWR-SER

The example shows the UART communication using the FlexIO module with following parameters:

- 8-bit communication
- One stop bit
- No parity
- No hardware flow control

Easy communication with the PC can be achieved with an OpenSDA interface using virtual CDC layer of USB standard. The TWR-SER card can also be used with the direct RS232 connection with the PC. The UART data frame emulated by FlexIO is shown in [Figure 1](#).

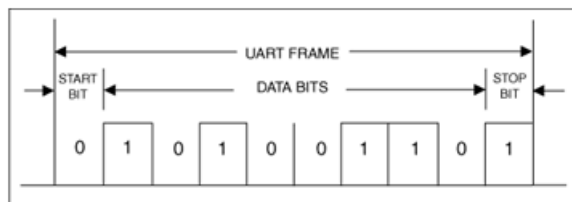


Figure 1. UART 8-bit data frame

## 4 UART in general

A universal asynchronous receiver/transmitter is a piece of computer hardware that translates data between parallel and serial forms. UARTs are commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485. The universal designation indicates that the data format and transmission speeds are configurable. The electric signaling levels and methods (such as differential signaling etc.) are handled by a driver circuit external to the UART.

A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port.

Transmitting and receiving UARTs must be set for the same bit speed, character length, parity and stop bits for proper operation. The receiving UART may detect some mismatched settings and set a "framing error" flag bit for the host system; in exceptional cases the receiving UART will produce an erratic stream of mutilated characters and transfer them to the host system.

Typical serial ports used with personal computers connected to modems use eight data bits, no parity, and one stop bit; for this configuration the number of ASCII characters per second equals the bit rate divided by 10.

## 5 UART emulation by FlexIO module

UART bus can be supported by using two timers, two shifters and two pins. The transmitter is supported by using one timer, one shifter and one pin. One timer, one shifter and one pin are used for the receiver part. Both transmitter and receiver parts can be used independently. The start and stop bit insertion is handled automatically by FlexIO peripheral. The maximum baud rate of the emulated peripheral is 115200 baud. Software implementation allows use UART in interrupt or polling modes.

Break and idle characters require software intervention and are not implemented in the example application. Configurable bit order (bit swapped buffer MSB first), and multiple transfers can be supported using DMA controller. FlexIO module does not allow automatic insertion of parity bits. [Figure 2](#) shows internal connection of FlexIO emulated UART.

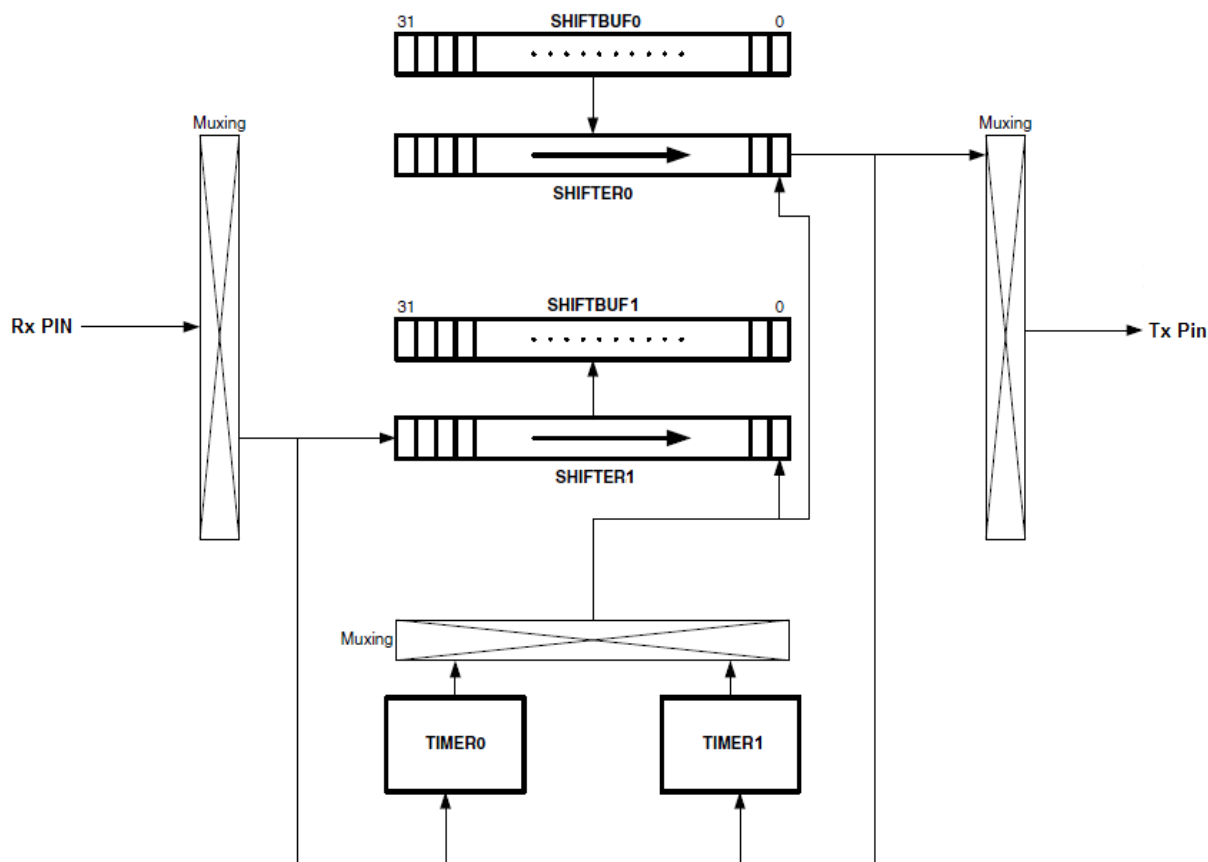


Figure 2. UART emulation block diagram

## 5.1 Transmitter

Transmitting procedure includes the following steps:

- Set the shifter to transmit mode
- Shift loaded data from the shifter buffer
- Shift the data to the pin output
- Start and stop bits are automatically loaded before or after data
- Use the timer status flag to send the next data frame

Figure 3 shows the principle of UART transmitter emulation. Timer status flag is checked in the polling mode and the module generates interrupt when the interrupt setting is enabled.

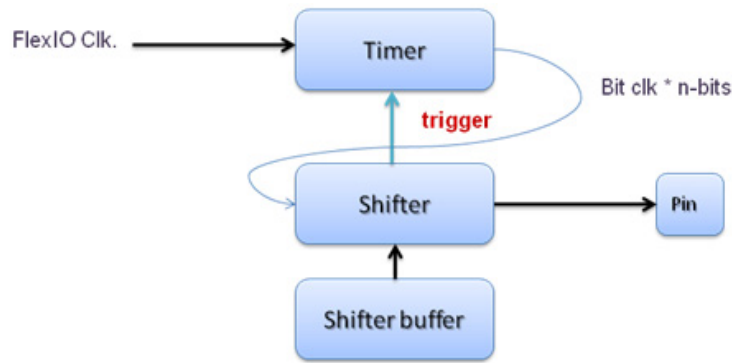


Figure 3. Block diagram of the UART transmitter on FlexIO module

## 5.2 Receiver

Receiving procedure includes the following steps:

- Set the shifter to receiver mode
- The data is shifted in when the store event is signaled
- The status flag indicates when data can be read (generate interrupt)
- Wait for the shifter status flag in polling mode
- Store into the shifter buffer
- Reading bit swaps the shifter buffer (without any logical operation)

Figure 4 shows the principle of UART receiver emulation. Shifter status flag is checked in the polling mode and the module generates interrupt when the interrupt setting is enabled.

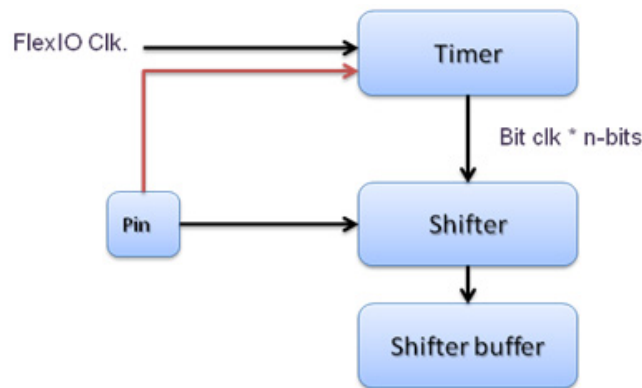


Figure 4. Block diagram of the UART receiver on FlexIO module

## 6 Software implementation

This example application is built by using software bare metal drivers. The driver is divided in two separate parts: The main low level driver is for initialization of peripherals, basic setting of the shifters and timers, and enables the user to read and write to the peripheral register. The sublayer of the bare metal driver is focused on the emulation of the current peripheral (UART). This driver can be used in other

instances. The application can use functions from both of the driver layers. Callback functionality is also implemented. Figure 5 shows the structure of the bare metal driver with all of drivers implemented for FlexIO peripheral module.

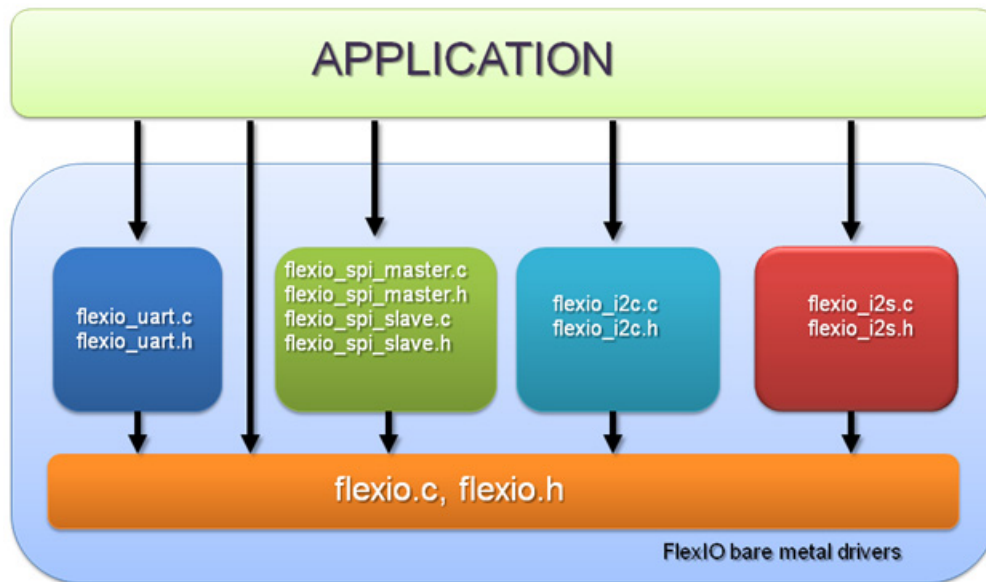


Figure 5. Block diagram of the FlexIO bare metal drivers

## 6.1 Software settings

The clocks are enabled for all required peripherals in SIM (system integration module). A high-frequency internal reference clock (HIRC) is used as a clock source, which gives 48 MHz to the system.

Initialization of the driver includes two functions. The first is for transmitting side *FLEXIO\_UART\_TxInit* and the second one for receiving side *FLEXIO\_UART\_RxInit*.

All configuration values are given from the file “*appconfig.h*”. These values are used for the calculation of the UART baudrate, but can be replaced by user definitions.

**Example of macro definition in “*appconfig.h*”:**

```
#define FLEXIO_UART_RX_BAUDRATE          9600
#define FLEXIO_UART_TX_BAUDRATE          9600
#define MCU_SYSTEM_CLOCK                  2000000
```

## 6.2 API of FlexIO high level driver

```
FLEXIO_UART_RESULT FLEXIO_UART_TxInit(FLEXIO_UART_CONFIG * pConfig,
uint32_t shifterOutIx, uint32_t timerOutIx, uint32_t pinTxIx, uint32_t baudRateTx,
uint32_t flexioClk);
```

FLEXIO\_UART module Tx initialization function:

- *pConfig* - pointer to configuration structure
- *shifterOutIx* - Index of shifter for transmit
- *timerOutIx* - Index of timer for transmit
- *pinTxIx* - Index of FlexIO Pin for Tx signal
- *baudRate* - Requested baudrate
- *FlexIoClk* - Clock frequency of flexio peripheral
- Return result of Init Operation

```
FLEXIO_UART_RESULT FLEXIO_UART_RxInit(FLEXIO_UART_CONFIG *pConfig, uint32_t shifterInIx, uint32_t timerInIx, uint32_t pinRxIx, uint32_t baudRateRx, uint32_t flexioClk);
```

FLEXIO\_UART module Rx initialization function:

- *pConfig* - pointer to configuration structure
- *shifterInIx* - Index of shifter for receive and ACK
- *timerInIx* - Index of timer for clocking of shifter registers
- *pinRxIx* - Index of flexio Pin for Rx signal
- *baudRate* - Requested baudrate
- *FlexIoClk* - Clock frequency of flexio peripheral
- Return result of Init Operation

```
FLEXIO_UART_RESULT FLEXIO_UART_PutStr(FLEXIO_UART_CONFIG *pConfig, const uint8 *str);
```

FLEXIO\_UART module Write string:

- *pConfig* - pointer to configuration structure
- *str* - pointer to string
- Return result of Write Operation

```
FLEXIO_UART_RESULT FLEXIO_UART_SendBuffer(FLEXIO_UART_CONFIG *pConfig, const uint8 *pBuff, uint32 len);
```

FLEXIO\_UART module Write data function:

- *pConfig* - pointer to configuration structure
- *pBuff* - pointer to input buffer

- *Return result of write operation*

```
uint8_t FLEXIO_UART_GetChar(FLEXIO_UART_CONFIG *pConfig);
```

FLEXIO\_UART module Read data function:

- *pConfig - pointer to configuration structure*
- *Return read byte of data in interrupt mode*

The following functions are used only in interrupt mode:

```
FLEXIO_UART_RESULT FLEXIO_UART_PutChar(FLEXIO_UART_CONFIG *pConfig, uint8 ch);
```

FLEXIO\_UART module putchar function:

- *pConfig - pointer to configuration structure*
- *ch - pointer input character*
- *Return result of write operation.*

```
FLEXIO_UART_SetCallBack(FLEXIO_UART_CONFIG *pConfig, FLEXIO_UART_CALLBACK  
pCallback);
```

FLEXIO\_UART module register CallBack function:

- *pConfig - pointer to configuration structure*
- *pCallback - pointer to function CallBack function (NULL for unregister)*
- *Result of register callBack function operation*
- *Implemented as function call.*

## 7 Conclusion

This application note demonstrates the use-case which can be easily implemented by FlexIO peripheral module available on the Freescale Kinetis KL43 MCU. The example application was created with and demonstrates the use of FlexIO emulated UART and hardware UART modules in the same application using the Freescale Tower platform. Together with this application note, the application software is available on the Freescale web pages.

## 8 References

- Freescale Tower system - [freescale.com/tower](http://freescale.com/tower)
- Emulating I<sup>2</sup>S bus master by FlexIO Application Note



- TWR-KL43Z48M: Kinetis KL43, KL33, KL27, KL17 48 MHz MCUs Tower System Module - [freescale.com/webapp/sps/site/prod\\_summary.jsp?code=TWR-KL43Z48M](http://freescale.com/webapp/sps/site/prod_summary.jsp?code=TWR-KL43Z48M)



### ***How to Reach Us:***

#### **Home Page:**

freescale.com

#### **Web Support:**

freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale 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.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 Freescale 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. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [freescale.com/SalesTermsandConditions](http://freescale.com/SalesTermsandConditions).

Freescale, the Freescale logo, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Tower is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM, the ARM Powered logo and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere.

© 2015 Freescale Semiconductor, Inc.

