NXP Semiconductors
Application Note

Document Number: AN5313
Rev. 0, 07/2016

Using FlexIO to Drive 8080 Bus Interface LCD Module

Using the FlexIO module to emulate 8080 bus interface

1. Introduction

The FlexIO module is a highly configurable peripheral, which allows users to implement a variety of functions. Depending on the module version, it is able to do the following:

- Emulate serial communication protocols, such as UART, SPI, I2C, I2S, and so on.
- Emulate parallel communication protocols, such as camera interface, Motorola 68 K bus, Intel 8080 bus, and so on.
- Generate the PWM waveform.
- Implement logic functions.
- Implement state machine functions.

Graphic TFT LCD modules are widely used in embedded applications which require GUI functions. The 8080 parallel bus is a common interface of a TFT LCD module.

This document describes how to use the FlexIO module to emulate the 8080 parallel bus and to drive a graphical TFT LCD with the 8080 bus interface.

The application is based on the recently launched MCU KL28Z. This Kinetis sub family is ARM® Cortex®-M0+ based microcontroller, which supports ultra-low-power and integrates crystalless USB, large memories (512 KB flash and 128 KB RAM), evolutionary low power peripherals, security features, and so on.

Contents

2. FlexIO Overview

2.1. Features and module block diagram

The FlexIO module version of the KL28Z provides the following key features:

- An array of 32-bit shift registers (also known as shifters) with transmit, receive, and data match modes, a double buffered structure for continuous transfer, and a concatenation mechanism to support large transfer sizes.
- Highly flexible 16-bit timers with support for a variety of internal or external trigger, reset, enable, and disable conditions.
- An automatic start/stop bit generation and check.
- 4, 8, 16, or 32 multibit shift widths for parallel interface support.
- An interrupt, DMA, or polled transmit/receive operation.
- Programmable baud rates supporting asynchronous operations during stop modes.
- A programmable logic mode for integrating external digital logic functions on-chip or combining pin/shifter/timer functions to generate complex outputs.
- A programmable state machine for offloading basic system control functions from the CPU.

The following diagram shows a high-level overview of the module.

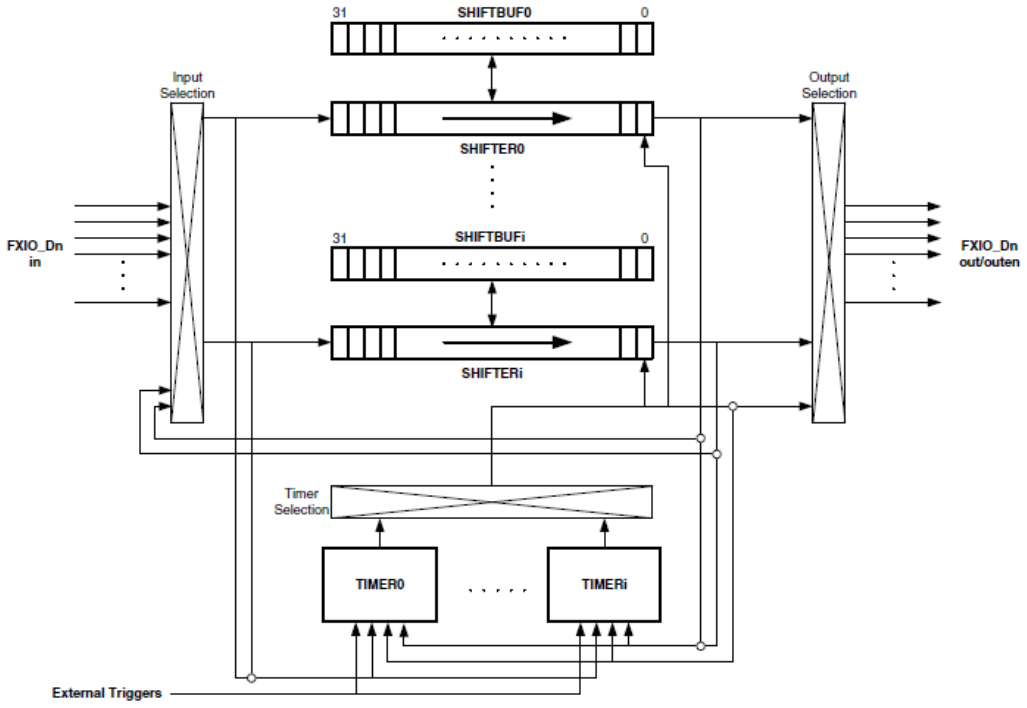


Figure 1. FlexIO block diagram

2.2. Internal logic connection

To satisfy various requirements, the internal logic connections are very flexible and intricate. Below are some capabilities.

- Any pin can be assigned to a shifter for input or output.
- Any timer can be assigned to a shifter for shift control.
- Any pin can be assigned to a timer for timer output or input.
- Timer triggers can originate from shifter flags, pins, or outside the FlexIO module.
- A shift can be on a rising or a falling edge of the shift clock.
- A pin direction and polarity is configurable.
- The trigger polarity is configurable.
- A timer's enable, disable, decrement, and reset conditions may originate from the trigger, pin, adjacent timer, and so on.

For detailed information, see the appropriate MCU reference manual.

2.3. Shifters and timers

The above description shows that FlexIO hardware resource consists of shifters, timers, and pins. The amount of these resources for a given MCU can be read from the PARAM register. For example, the FlexIO module of KL28Z has 8 shifters, 8 timers, and 32 pins.

Transmit and receive are two basic modes of the shifters. When a shifter is in transmit mode, it loads data from its buffer register and then shifts the data out to its assigned pin/pins. When a shifter is in receive mode, it shifts data in from its assigned pin/pins and then stores the data into its buffer register. Loading, storing, and shifting operations are controlled by the shifter's assigned timer.

The timers can also be configured in different operating modes as needed, including dual 8-bit counters baud/bit mode, dual 8-bit counters PWM mode, and single 16-bit counter mode. The dual 8-bit counters baud/bit mode usually implements a data transmitter. In this mode, the lower eight bits of the 16-bit timer divides the module clock source to generate desired baud rate, and the higher eight bits counts the shift bits of a frame. After it is enabled, the timer loads the initial value from its compare register and starts to decrement count. When the lower eight bits decrement to zero, the timer's shift clock and its output signal are toggled to generate a rising or a falling edge. The higher eight bits' decrement counts by one. The shift clock drives the shifter. The timer output signal usually drives a pin for the clock output, such as the SCK of a SPI master and WR of the 8080 bus. After that, the lower eight bits reload the initial value to start another decrement cycle. Two decrement cycles make up a shift cycle, which drives the shifter to shift one beat. When the whole 16 bits' decrement count to zero, all data bits in the shifts are shifted out. Then, the timer is disabled before another transfer frame.

2.4. General configurations and operations

The FlexIO can emulate various communication protocols. However, to emulate a dedicated peripheral and to handle transmit and receive process, FlexIO must be configured using software.

Generally, to implement a master transmitter, a shifter is configured in transmit mode and the assigned timer is configured in dual 8-bit counters baud/bit mode. The timer decrement clock originates from the

module clock. The timer trigger originates from the shifter's flag with reversed polarity. Filling the shifter's buffer via polling/interrupt/DMA clears the shifter flag, which enables the timer to start decrement count. The decrement of the timer drives the shifter to shift data out and generates the clock output signal.

To implement a receiver, a shifter is configured in receive mode. The timer is configured in dual 8-bit counters baud/bit mode for synchronous maser receiver, such as the 8080 bus reading implementation. This timer mode is also used for asynchronous receiver, such as the UART receiver. The receive process is similar to that of a master transmitter but the data is shifted in to the shifter rather than shifted out. For the synchronous receiver, the assigned timer is configured in single 16-bit counter mode, such as the SPI slave receiver. The decrement clock originates from the pin input, such as the SPI SCK signal. The timer trigger originates from another pin, such as the SPI CS signal. The master device enables the timer and control the decrement via pins. Similarly, the decrement of the timer drives the shifter to shift in the data.

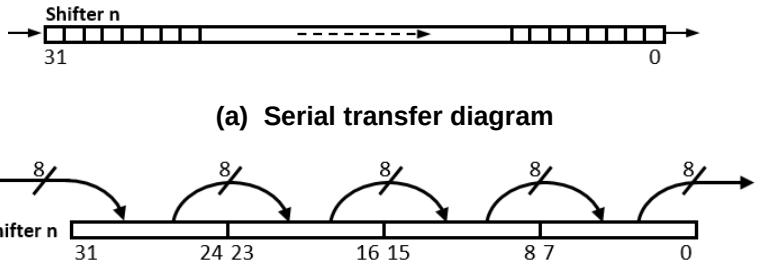
For ease of use, NXP Semiconductors provides API drivers and driver examples in the KSDK (Kinetis Software Development Kit). Users can also find detailed descriptions and similar APIs of these implementations in various application notes.

2.5. FlexIO parallel transfer

The early FlexIO IP versions, such as the version of KL17/KL27/KL33/KL43, supported serial transfers only. The later versions, however, support both serial and parallel transfers, such as KL28Z.

Data is always shifted from MSB to LSB in a shifter. In serial transfer mode, the data is shifted out bit by bit from the LSB (Bit 0) and is shifted in bit by bit from the MSB (Bit 31). This process is shown in Figure 2-a.

In parallel transfer mode, the data is shifted out from the *n* LSBs of a shifter and shifted in from the *n* MSBs, where *n* is the parallel bus width. Figure 2-b shows the use case of *n* = 8.



(a) Serial transfer diagram

(b) Parallel transfer diagram

Figure 2. Serial and Parallel transfer diagram

The following describes parallel transfer mode.

- Data is shifted *n* bits on each shift clock, where n is the configured bus width.
- 4, 8, 16, or 32 bus width is supported.
- Combine multiple shifters together for concatenation to support large transfer sizes and use DMA method to access the shifter buffer registers for high-speed transfers and low-power operations. The figure below shows the shifter concatenation diagram where additional shifters

work as FIFOs.

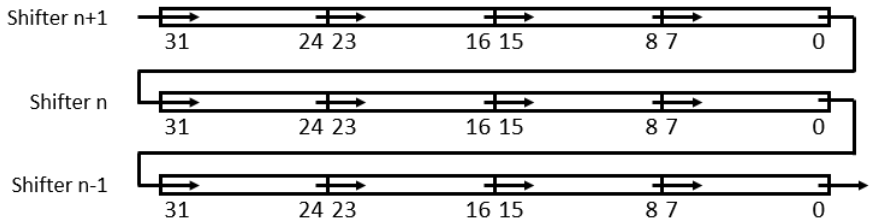


Figure 3. Shifter concatenation diagram

- Only specific shifters (SHIFTER0 and SHIFTER4) support parallel outputting to FlexIO pins. However, all shifters, except the SHIFTER0, support outputting to the adjacent low-order shifters.
- Similarly, only specific shifters (SHIFTER3 and SHIFTER7) support parallel inputting from FlexIO pins. However, all shifters, except the SHIFTER7, support inputting from the adjacent high-order shifters.
- Any FlexIO pin can be a parallel output/input pin. However, the pin indexes must be successive for a specific usage, such as pin0 to pin7, or pin1 to pin8, and so on for 8-bit width bus.

3. 8080 Parallel Bus Sequence for LCD Modules

Generally, the 8080 parallel interface used for LCD modules consists of 8 or 16 bi-directional data lines (Data Bus), one chip-select line (CS), one writing-latch line (WR), one reading-latch line (RD), and one data/command-select line (RS).

CS, WR, RD, and RS are all low-level active. Low-level of the CS selects the slave device. The rising edge of the WR line is a data write latch signal (clock). The rising edge of the RD line is a data read latch signal (clock). RD should be high-level when a writing sequence is in progress. Similarly, WR should be high-level when a reading sequence is in progress. RS is a data/command select signal. A low-level of RS indicates command (or address) transfers. A high-level of RS indicates data transfers. RS is also known as DC or ALE (Address Latch Enable).

At the beginning of a wring/reading transfer, a command/address writing sequence specifies the target address. Data transfers can be one or more beats.

Figure below shows the writing sequence. A data transfer occurs during the writing sequence under a 0-beat command type. See image below.

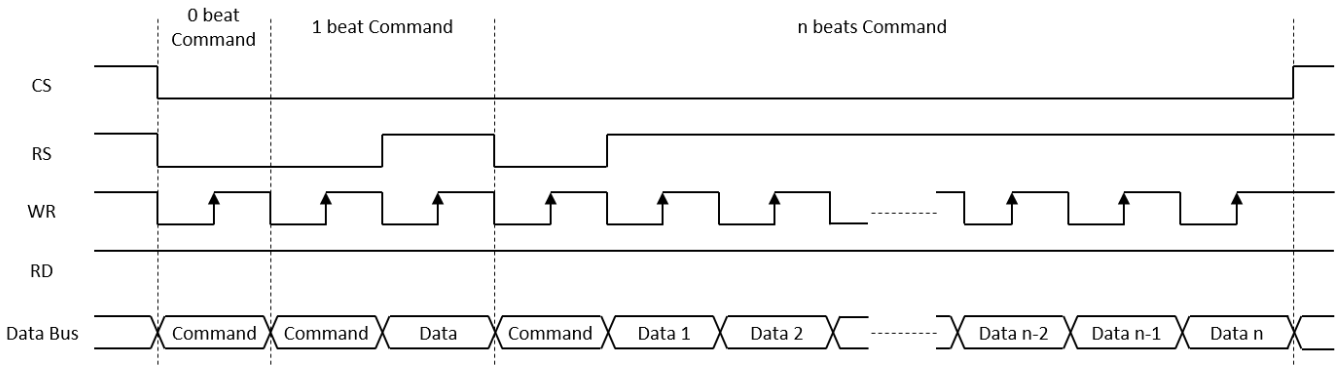


Figure 4. 8080 bus writing sequence diagram

Figure below shows the reading sequence. A dummy reading beat can occur between the command-writing beat and the first data-reading beat, depending on the bus slave.

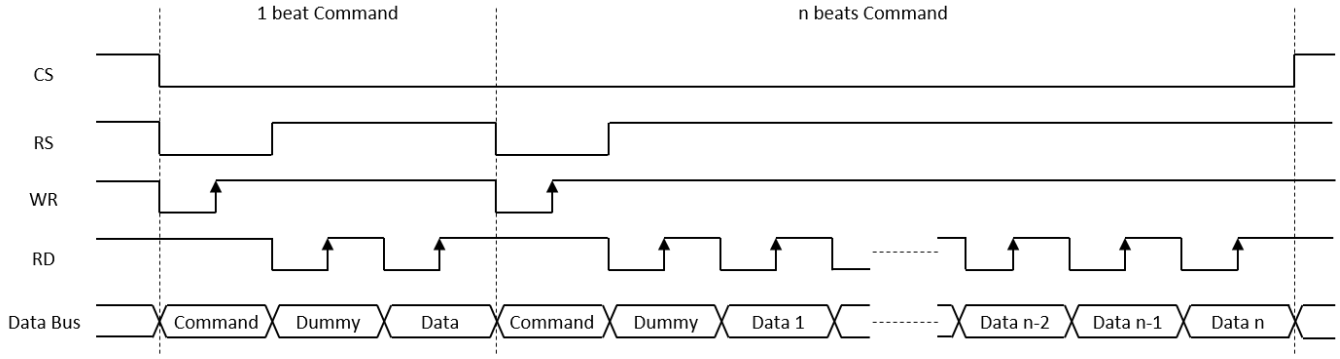


Figure 5. 8080 bus reading sequence diagram

4. Emulating 8080 Bus and Driving LCD Module

This section describes how to use FlexIO and emulate 8080 bus interface to drive a TFT LCD module.

4.1. Configure FlexIO to emulate 8080 bus

Users can configure FlexIO to emulate 8080 bus in different ways, such as different bus width, baud rate, the number of the concatenated shifters, specific shifters, pins, and timer to use, and so on. Set up proper configurations based on the application requirements.

In this application, the 8-bit and 16-bit width bus are implemented. Because module configurations and software drivers are very similar between the 8-bit and 16-bit bus implementations, the following sections only describe the 8-bit implementation.

Both writing and reading functions have been implemented. The writing function is implemented by configuring the shifters in transmit mode. The reading function is implemented by configuring the shifters in receive mode.

Both 1-beat transfer and multibeat transfer drivers have been implemented. The 1-beat transfers transfer small size data, such as configuring an LCD driver IC's registers. Shifter concatenation is not used for 1-beat transfers because only one shifter is used. One transfer sequence requires the timer to generate only one shift clock. Eight bits are transferred in/out simultaneously. In this application, a polling method accesses the shifter buffers for the 1-beat transfers.

The multibeat transfers support large transfer sizes, such as transferring frame data to a LCD module. One transfer sequence requires the timer to generate multiple shift clocks. The number of beats per one transfer sequence is related to the number of the concatenated shifters and the bus width. All shifters are 32-bit size. One shifter supports, at most, a four beat transfer for the 8-bit width bus. Two shifters support, at most, eight beats, and so on. In this application, all eight shifters are used and 32 beats are supported for the 8-bit width bus. DMA method accesses the shifter buffers for the multibeat transfers in this application.

In a real application, writing and reading does not occur at the same time. 1-beat and multibeat transfers also never operate simultaneously. In addition, the shifters and the pins are shared between different functions. Therefore, users should reconfigure the FlexIO module at each switching of functions.

There are four implementations for writing and reading, 1-beat and multibeat transfers, as follows.

1. 1-beat writing
2. 1-beat reading
3. multibeat writing
4. multibeat reading

In this application, the 1-beat writing is widely used, such as configuring the LCD driver IC's registers, transferring smaller frame data, and transferring the command beat in all of the four implementations, and so on. The multibeat writing mainly transfers larger frame data to an LCD. The 1-beat reading and multibeat reading are not used in this application.

Because 1-beat transfers are a simpler version of the multibeat transfers, but, multibeat writing and multibeat reading are briefly described in the following sections.

4.1.1. Multibeat writing

Figure 6 shows the FlexIO module configuration for multibeat writing transfers.

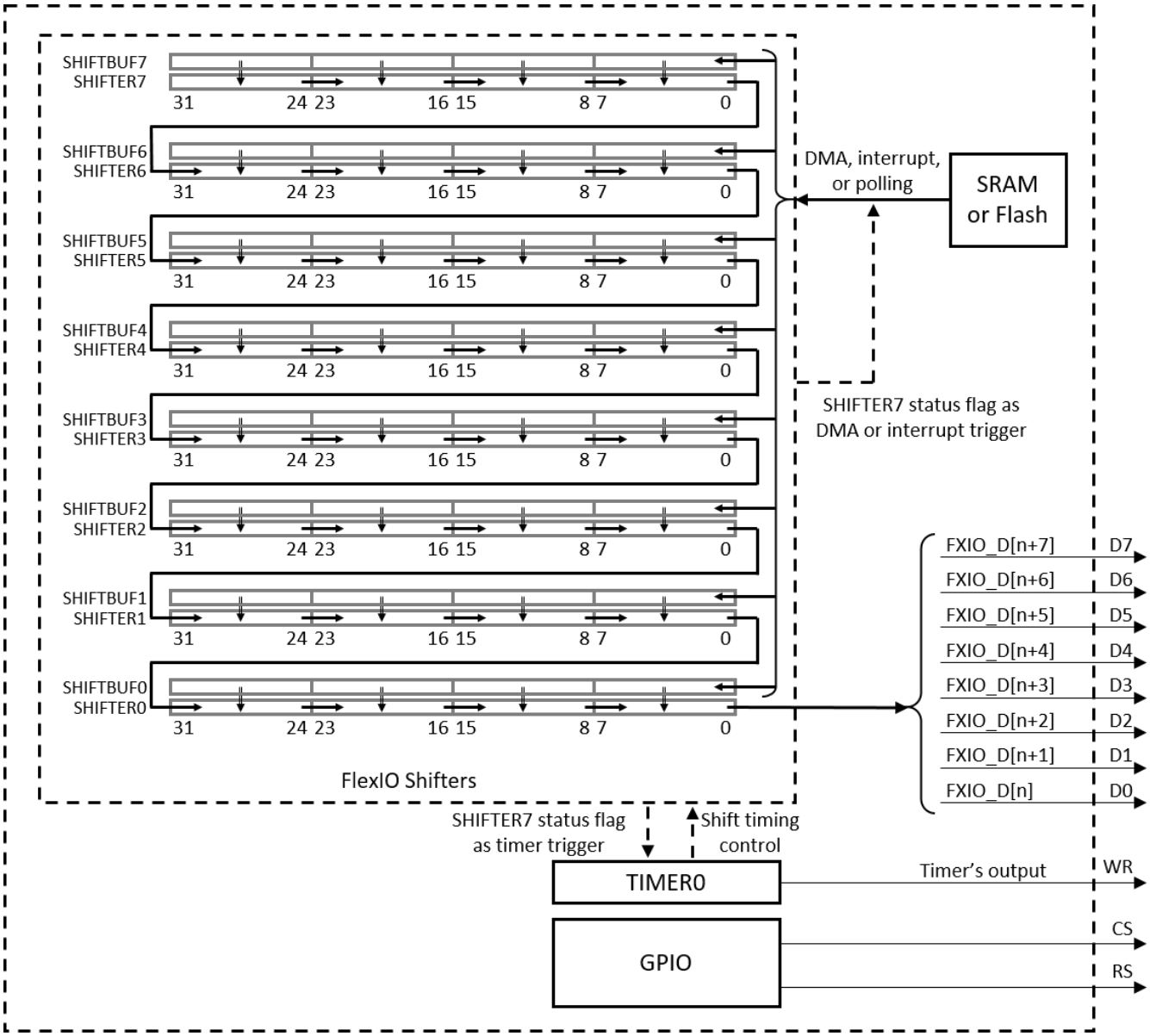


Figure 6. Multibeat writing configuration

In the configuration, all eight shifters are concatenated together. TIMER0 generates a shift clock and a WR signal. D0~D7 and WR originate from FlexIO pins. Additional GPIO pins drive CS and RS signals. SHIFTER7 status flag triggers the TIMER0 and generates the DMA request.

The steps below describe multibeat writing transfers using the DMA method.

1. Configure DMA, FlexIO module, and GPIO. Enable DMA request of the SHIFTER7 status flag.
2. The DMA request responds immediately after the enablement of the request. The DMA copies data from SRAM or flash to the shifter buffers SHIFTBUF0~SHIFTBUF7. In total, 32 bytes are

copied per one DMA request.

NOTE

Addresses of SHIFTBUF0~SHIFTBUF7 are successive.

3. Filling the shifter buffers clears the shifter status flags, which enables the selected TIMER0.
4. TIMER0 signals a loading event. Data are loaded from the shifter buffers SHIFTBUF0~SHIFTBUF7 to the shifters SHIFTER0~SHIFTER7.
5. The loading event empties the shifter buffers, which sets the shifter status flags and triggers another DMA request. The DMA fills SHIFTBUF0~SHIFTBUF7 with new data and the shifter status flags are cleared again.
6. TIMER0 starts to decrement count after the loading event. It generates the timer shift clock along with the decrement to control the shifters shifting data out and generates the timer output to drive WR signal.
7. After the configured 32 shift clocks, TIMER0 decrement counts to zero and a compare event occurs. Then TIMER0 is disabled for a short time.
8. Because the shifter buffers contain valid data and SHIFTER7 status flag is zero at this point, TIMER0 is enabled again.

   Then, step 3-8 repeat.
9. After all data has been copied to the shifter buffers, the DMA completes the major loop. No more data is copied and TIMER0 is not triggered after the last compare event. A DMA interrupt is generated to indicate the completing of the major loop.
10. If the total transfer size is not divisible by 32, the additional bytes are transferred in 1-beat mode using a polling method.

In addition, the CS is pulled low before the transfer sequence by software and pulled up after the transfer sequence. RS is pulled up before the command transmission and pulled up again after that.

4.1.2. Multibeat reading

Figure below shows the FlexIO module configuration for multibeat reading transfers.

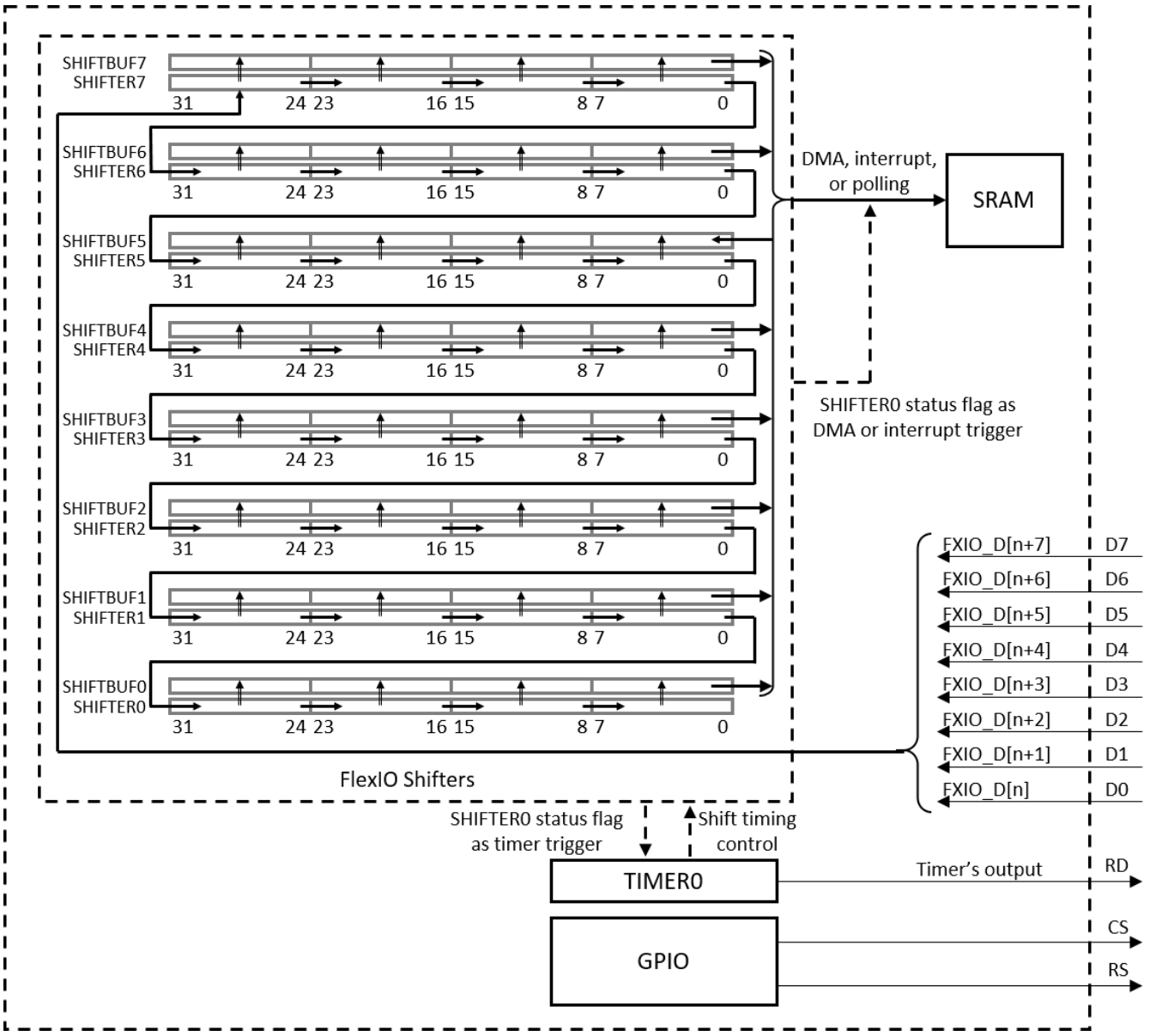


Figure 7. Multibeat reading configuration

In the configuration, all eight shifters are concatenated together. TIMER0 is used to generate the shift clock and the RD signal. D0-D7 and RD are based on FlexIO pins. Additional GPIO pins are used to drive the CS and the RS signal. SHIFTER0 status flag is used to trigger TIMER0 and generate the DMA request.

The items below describe the process of multibeat reading transfers that use the DMA method.

1. Configure DMA, FlexIO module, and GPIO. Enable the DMA request of SHIFTER0 status flag.
2. TIMER0 is enabled immediately after the completing of the configuration and starts to decrement count. It generates the timer shift clock along with the decrement to control the shifters shifting data in and generates the timer output to drive the RD signal.
3. TIMER0 decrement counts to zero and a compare event occurs after 32 shift clocks. Then, the

TIMER0 is disabled.

4. A storing event is signaled by the TIMER0 after the compare event. Data is stored from the shifters SHIFTER0-SHIFTER7 to the shifter buffers SHIFTBUF0-SHIFTBUF7.
5. The storing event fills up the shifter buffers, which sets the shifter status flags and triggers a DMA request.
6. The DMA request is responded. The DMA copies data from the shifter buffers SHIFTBUF0-SHIFTBUF7 to SRAM. 32 bytes are copied per one DMA request.
7. The DMA reading operation clears the shifter flags, which enables the TIMER0 again.

   Then, steps 2-7 repeat.
8. After the major loop completes, the TIMER0 is triggered once more to shift in the last 32 bytes. A DMA interrupt is generated to indicate the completion of the DMA major loop.

NOTE

The DMA major loop size is configured not to include the last minor loop to avoid enabling the TIMER0 after completing all transfers.

9. If the total transfer size is not divisible by 32, the additional bytes are transferred in 1-beat mode using a polling method.

In addition, the CS is pulled low before the transfer sequence by software and pulled up after the transfer sequence. The RS is pulled up before the command transmission and pulled up again after that.

4.2. The driver functions

A software package is provided along with this application note. You can get the software package by searching AN5313 on the NXP Semiconductors homepage www.nxp.com. There are two IAR projects in the software package for 8-bit and 16-bit bus interface demos. This section describes several major driver functions with 8-bit implementation as an example. These driver functions are located in /boards/twrkl28z72m/driver_examples/flexio_8080_8bits/source/ flexio_8080_drv.c.

4.2.1. FlexIO API driver functions

1. void FLEXIO_8080_SglBeatWR_1Prm(uint32_t const cmdIdx, uint8_t const value)
   - Function: Write a single parameter in a single-beat writing mode using a polling method.
   - Parameter: cmdIdx – the command index
   - Parameter: value – the command value
2. void FLEXIO_8080_SglBeatWR_nPrm(uint32_t const cmdIdx, uint8_t const * buffer, uint32_t const length)
   - Function: Write multi parameters in a single-beat writing mode using a polling method.
   - Parameter: cmdIdx – the command index

- Parameter: buffer – the buffer used to store the command value
- Parameter: length – the command size

3. void FLEXIO_8080_SglBeatWR_nSamePrm(uint32_t const cmdIdx, uint16_t const value, uint32_t const length)
   - Function: Write multirepeated parameters in a single-beat writing mode using a polling method. This function is used to fill the LCD module with solid color.
   - Parameter: cmdIdx – the command index
   - Parameter: value – the command value
   - Parameter: length – the command size
4. uint8_t FLEXIO_8080_SglBeatRD_1Prm(uint32_t const cmdIdx)
   - Function: Read a single parameter in a single-beat reading mode using a polling method.
   - Parameter: cmdIdx – the command index
   - return - the read result
5. void FLEXIO_8080_SglBeatRD_nPrm(uint32_t const cmdIdx, uint8_t * buffer, uint32_t const length)
   - Function: Read multiple parameters in a single-beat reading mode using a polling method.
   - Parameter: cmdIdx – the command index
   - Parameter: buffer – the buffer used to store command value
   - Parameter: length – the command size
6. void FLEXIO_8080_MulBeatWR_nPrm (uint32_t const cmdIdx, uint8_t * buffer, uint32_t const length)
   - Function: Write multiple parameters in a multibeat writing mode using the DMA method.
   - Parameter: cmdIdx - the command index
   - Parameter: buffer – the buffer used to store command value
   - Parameter: length – the command size
7. void FLEXIO_8080_MulBeatRD_nPrm (uint32_t const cmdIdx, uint8_t * buffer, uint32_t const length)
   - Function: Read multiple parameters in a multibeat reading mode using the DMA method.
   - Parameter: cmdIdx – the command index
   - Parameter: buffer – the buffer used to store command value
   - Parameter: length – the command size

4.3. Hardware platforms

TWR-KL28Z72M Tower development module, shown in the figure below, is used as an example in this application. There is a pin header J26 on the board connected to the FlexIO pins, which makes it convenient for the hardware connections of this application.

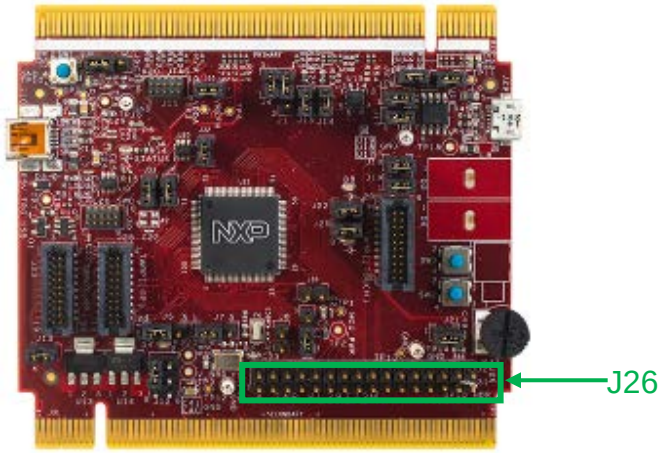


Figure 8. TWR-KL28Z72M development board

The LCD module used in this application integrates a Himax LCD driver IC HX8357. It supports many data transfer interfaces, including MIPI-DBI 8-/9-/16-/18-/24-bit parallel interface (8080 compatible), MIPI-DBI serial interface, MIPI-DPI 8-/16-/18-/24-data lines parallel (RGB) interface, and MIPI-DSI interface. The 8080 compatible 8-bit and 16-bit MIPI-DBI parallel interfaces are used in this application. The application uses the boot strapping method with several resistors on the module PCB to pull up or pull down the HX8357's interface selection pins to select a specific interface.

The system block diagram of this application demo is shown in Figure 9.

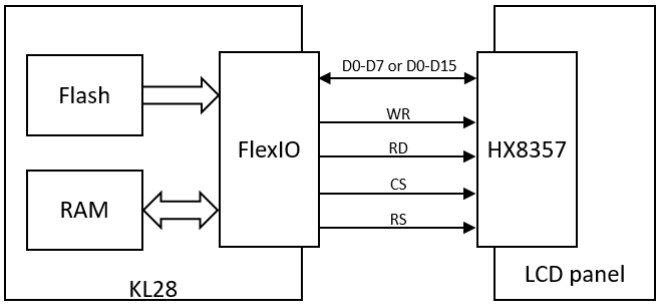


Figure 9. System block diagram

4.4. FlexIO configurations and hardware connections

This section provides detailed configurations of the FlexIO registers and hardware connections between the TWR-KL28Z72M module and the LCD module.

4.4.1. FlexIO configurations

Table 1 to Table 4 provide primary register configurations of the FlexIO for the four operating modes.

Table 1. 1-beat writing

Register	Value	Comment
SHIFTCFG0	0x0007_0100	Configure shifter stop bit disabled, start bit disabled and loading data on enabled, 8-bit parallel width.
SHIFTCTL0	0x0003_0002	Configure shift clock from TIMER0, shift on posedge of shift clock. Configure shifter's pin as output, pin index starting from 0, and pin polarity active high. Configure shifter mode as transmit.
TIMCMP0	0x0000_0101	Set TIMCMP [15:8] =number of beatsx2–1=1x2-1=1 for one beat. Set TIMCMP [7:0] =baud rate divider/2–1=4/2-1=1.
TIMCFG0	0x0000_2200	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, shift clock equals timer output, never reset, disabled on timer compare, enabled on trigger high, stop bit disabled, and start bit disabled.
TIMCTL0	0x01C3_1081	Configure SHIFT0 status flag as timer trigger, and trigger polarity active low. Configure timer’s pin as output, pin index as 16 (WR), pin polarity active low. Configure timer mode as 8-bit counters baud/bit.

Table 2. 1-beat reading

Register	Value	Comment
SHIFTCFG7	0x0007_0000	Configure shifter input from pin, stop bit disabled, start bit disabled and loading data on enabled, 8-bit parallel width.
SHIFTCTL7	0x0080_0001	Configure shift clock from TIMER0, shift on negedge of shift clock. Configure shifter's pin as input, pin index starting from 0, and pin polarity active high. Configure shifter mode as receive.
TIMCMP0	0x0000_0101	Set TIMCMP [15:8] =number of beatsx2-1=1x2-1=1 for one beat. Set TIMCMP [7:0] =baud rate divider/2-1=4/2-1=1.
TIMCFG0	0x0000_2220	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, shift clock equals timer output, never reset, disabled on timer compare, enabled on trigger high, stop bit enabled on timer disabled (for internal signal synchronization to disable timer in time), and start bit disabled.
TIMCTL0	0x1DC3_1181	Configure SHIFT7 status flag as timer trigger, and trigger polarity active low. Configure timer’s pin as output, pin index as 17 (RD), pin polarity active low. Configure timer mode as 8-bit counters baud/bit.

Table 3. Multibeat writing

Register	Value	Comment
SHIFTCFG0…7	0x0007_0100	Configure shifter input from next shifter's output, stop bit disabled, start bit disabled and loading data on enabled, 8-bit parallel width.
SHIFTCTL0	0x0003_0002	Configure shift clock from TIMER0, shift on posedge of shift clock. Configure shifter's pin as output, pin index starting from 0, and pin polarity active high. Configure shifter mode as transmit.
SHIFTCTL1…7	0x0000_0002	Configure transmit using TIMER0, shift on posedge of shift clock. Configure shifter's pin as output disabled, pin index starting from 0, and pin polarity active high. Configure shifter mode as transmit.
TIMCMP0	0x0000_3F01	Set TIMCMP [15:8] =number of beatsx2-1=32x2-1=63 for 32 beats. Set TIMCMP [7:0] =baud rate divider/2-1=4/2-1=1.

Register	Value	Comment
TIMCFG0	0x0000_2200	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, shift clock equals timer output, never reset, disabled on timer compare, enabled on trigger high, stop bit disabled, and start bit disabled.
TIMCTL0	0x1DC3_1081	Configure SHIFT7 status flag as timer trigger, and trigger polarity active low. Configure timer's pin as output, pin index as 16 (WR), pin polarity active low. Configure timer mode as 8-bit counters baud/bit.

Table 4. Multibeat reading

Register	Value	Comment
SHIFTCFG0…6	0x0007_0100	Configure shifter input from next shifter's output, stop bit disabled, start bit disabled and loading data on enabled, 8-bit parallel width.
SHIFTCFG7	0x0007_0000	Configure shifter input from pin, stop bit disabled, start bit disabled and loading data on enabled, 8-bit parallel width.
SHIFTCTL0…7	0x0080_0001	Configure shift clock from TIMER0, shift on negedge of shift clock. Configure shifter's pin as input, pin index starting from 0, and pin polarity active high. Configure shifter mode as receive.
TIMCMP0	0x0000_3F01	Set TIMCMP [15:8] =number of beatsx2-1=32x2-1=63 for 32 bits. Set TIMCMP [7:0] =baud rate divider/2-1=4/2-1=1.
TIMCFG0	0x0000_2220	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, shift clock equals timer output, never reset, disabled on timer compare, enabled on trigger high, stop bit enabled on timer disabled (for internal signal synchronization to disable timer in time), and start bit disabled.
TIMCTL0	0x01C3_1181	Configure SHIFT0 status flag as timer trigger, and trigger polarity active low. Configure timer's pin as output, pin index as 17 (RD), pin polarity active low. Configure timer mode as 8-bit counters baud/bit.

4.4.2. Hardware connections

Table 5 to Table 7 provide the hardware connections between the TWR-KL28Z72M module and the LCD module. Use external wires to set up these connections.

Table 5 lists the connections of the FlexIO pins to the LCD module. Note that FXIO_D8 to FXIO_D15 pins are dedicated for 16-bit bus interface.

Table 5. FlexIO pins assignments and connections

FlexIO Pins	Port Pins	Board Connector	LCD signal
FXIO_D0	PTD0	J26-0	D0
FXIO_D1	PTD1	J26-1	D1
FXIO_D2	PTD2	J26-2	D2
FXIO_D3	PTD3	J26-3	D3
FXIO_D4	PTD4	J26-4	D4
FXIO_D5	PTD5	J26-5	D5
FXIO_D6	PTD6	J26-6	D6
FXIO_D7	PTD7	J26-7	D7
FXIO_D8	PTB0	J26-8	D8
FXIO_D9	PTB1	J26-9	D9
FXIO_D10	PTB2	J26-10	D10
FXIO_D11	PTB3	J26-11	D11
FXIO_D12	PTB8	J26-12	D12
FXIO_D13	PTB9	J26-13	D13

Table 5. FlexIO pins assignments and connections

FlexIO Pins	Port Pins	Board Connector	LCD signal
FXIO_D14	PTB10	J26-14	D14
FXIO_D15	PTB11	J26-15	D15
FXIO_D16	PTB16	J26-16	WR
FXIO_D17	PTB17	J26-17	RD

Table 6 lists the connections of the GPIO pins to the LCD module.

Table 6. GPIO pins assignments and connections

GPIO Pins	Port Pins	Board Connector	LCD signal
GPIOB18	PTB18	J26-18	CS
GPIOB19	PTB19	J26-19	RS

Table 7 lists the power supply connections for the LCD module.

Table 7. Power supply connections

Power	Board Connector	LCD signal
VCC	J12-4	3V3
GND	J12-6	GND

4.5. Run the demo

Download and unzip the software package. Find the IAR project in */boards/twrkl28z72m/driver_examples/flexio_8080_8bits/iar*.

Build, download, and run the demo. You can see a picture of NXP logo displayed on the LCD screen, as shown in figure 10. The logo image moves around the screen and is echoed when it reaches any of the edges.

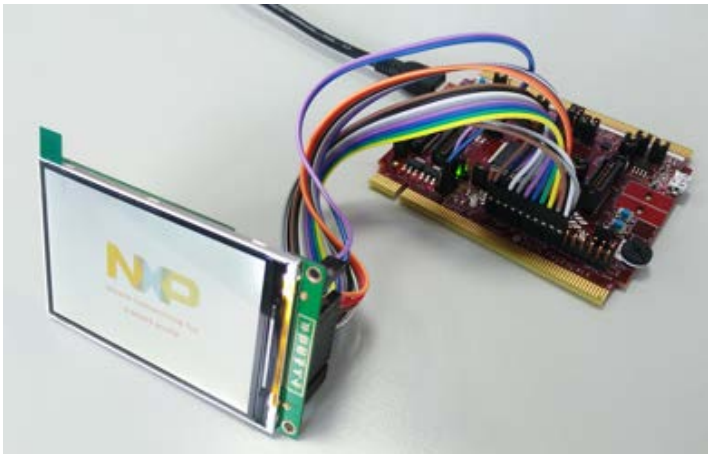


Figure 10. Run the demo

The demo implements several display modes. The user can press the on board button SW2 to switch the display modes. One mode tests the refresh rate. In this mode, the LCD is refreshed with solid colors successively and the green LED D7 toggles during the refreshing intervals. To measure the refresh rate, connect an oscilloscope probe to J20 connected to D7.

The refresh rate of the 8-bit width demo measured by an oscilloscope is about 39 fps, which is close to the theoretical value. Below is the calculation of the theoretical value.

- FlexIO clock frequency: 48 MHz from FIRC
- FlexIO baud rate divider: 4
- LCD frame size: 320x480x2 bytes
- Refresh rate: 48,000,000/4/ (320x480x2) = 39.0625 fps

To get a higher refresh rate, use a higher FlexIO clock frequency, such as 72 MHz or 96 MHz from PLL. Note that the 16-bit width almost doubles the refresh rate.

To learn 8080 bus timing sequence and the API drivers of this application well, capture the actual bus signals with a logic analyzer. Figure below shows a section of the multibeat writing waveform.

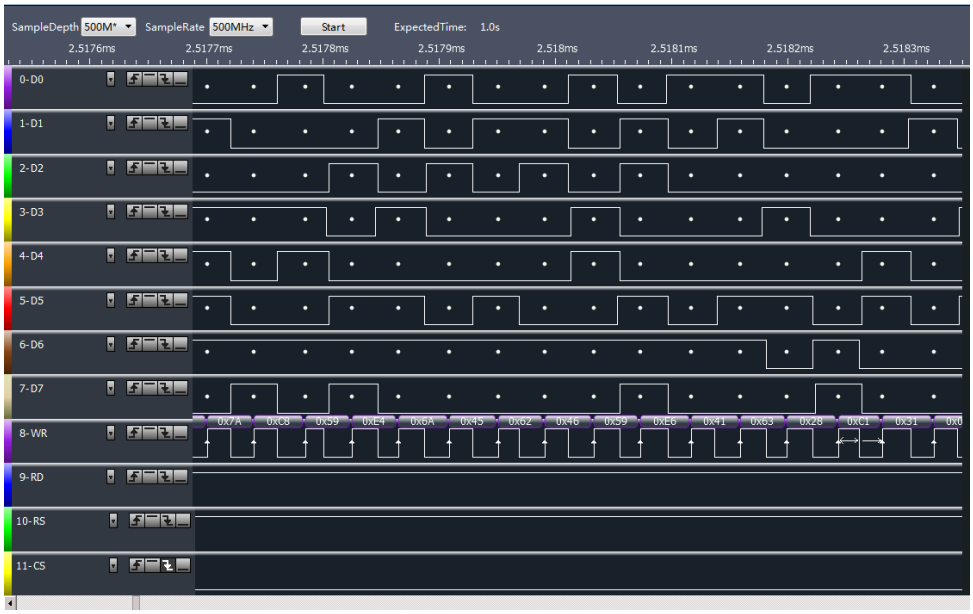


Figure 11. Actual multibeat writing waveforms

5. Conclusion

This application implements a demo that emulates 8080 bus by using FlexIO. Several driver functions are implemented for writing and reading to drive a TFT LCD module. The measurements indicate that the performance is satisfactory with 39 fps needed to refresh a 320x480x2 sized LCD module.

6. Revision History

The revision history tracking for this document is in the table below.

Table 8. Revision history

Revision number	Date	Substantive changes
0	07/2016	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN5313
Rev. 0
07/2016