AN12813

Emulating 6800 Bus with the FlexIO on RT1010

Rev. Rev 0 — 15 April 2020

Application Note

Contents

1 Introduction

This application note describes how to use the FlexIO module to emulate the 6800 parallel bus.

The 6800 bus, usually called Motorola 68K bus, is a parallel bus interface used by smart and asynchronous LCD controllers.

FlexIO is an on-chip peripheral available on NXP i.MXRT series. It is a high-configurable module capable of emulating a variety of serial/parallel communication protocols, such as UART, I2C, and SPI. Users can also use FlexIO to generate 6800 bus.

The i.MX RT1010 processor is based on the ARM Cortex-M7 platform. The processor has rich peripheral devices and provides high CPU performance and best real-time response. To verify the 6800 bus emulated via FlexIO, a simple application is implemented on the RT1010-EVB board.

2 FlexIO overview

This section describes features of FlexIO and parallel transfer modes.

2.1 Features

The FlexIO module of the i.MXRT1010 provides the following key features:

- Array of 32-bit shift registers with transmit, receive, and data match modes
- Double buffered shifter operation for continuous data transfer
- Shifter concatenation to support large transfer sizes
- Automatic start/stop bit generation
- 1, 2, 4, 8, 16 or 32 multi-bit shift widths for parallel interface support
- Interrupt, DMA, or polled transmit/receive operation
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes
- Highly-flexible 16-bit timers with support for various internal or external trigger, reset, enable and disable conditions
- Programmable logic mode for integrating external digital logic functions on-chip or combining pin/shifter/timer functions to generate complex outputs
- Programmable state machine for offloading basic system control functions from CPU with support for up to 8 states, 8 outputs, and 3 selectable inputs per state

The following figure shows a high-level overview of the configuration of FlexIO timers and shifters.

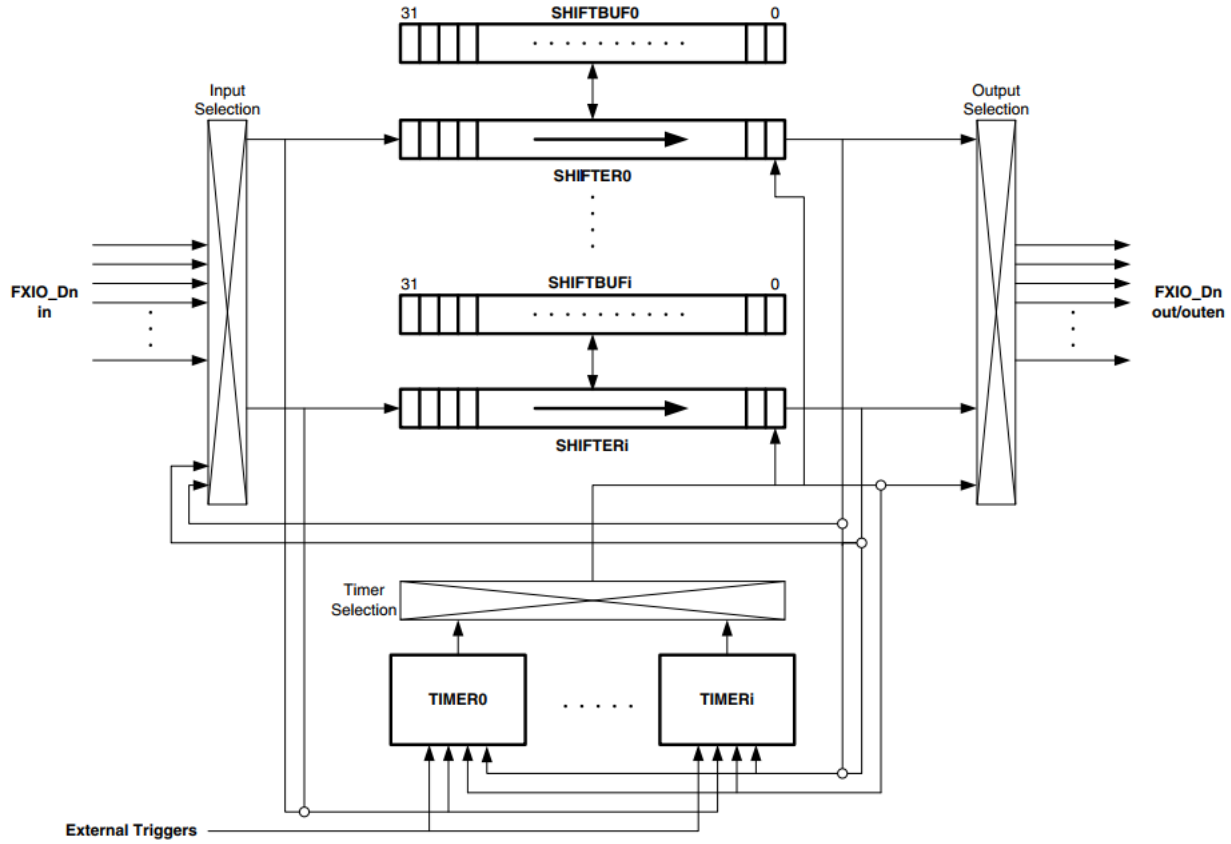

Figure 1. FlexIO block diagram

FlexIO on i.MXRT1010 has 8 shifters, 8 timers, and only 27 pins. Shifters are responsible for buffering and shifting data into or out of the FlexIO. The 16-bit timers control the loading, shifting, and storing of the shift registers. The pin configuration for each timer and shifter can be configured to use any FlexIO pin with either polarity.

2.2 Parallel transfer

The FlexIO of i.MXRT1010 supports both serial and parallel transfer modes. Shifters can be configured to use multiple FlexIO pins in parallel using the SHIFTCFG[PWIDTH] field. PWIDTH is the bus width that configures the following settings of a shifter:

- Number of bits shifted per shift clock
- Number of pins driven by the shifter per shift clock
- Number of pins sampled by the shifter per shift clock

When configured for parallel shift, either 4, 8, 16, or 32-bits can be shifted on every shift clock. To support large transfer sizes, multiple shifters can be combined together for concatenation. The DMA method is used to access the shifter buffer registers for high-speed transfers.

For parallel transmit, only SHIFTER0 and SHIFTER4 support outputting to FlexIO pins. However, all shifters, except the SHIFTER0, support outputting to the adjacent low-order shifters.

Similarly, for parallel receive, only SHIFTER3 and SHIFTER7 support 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 6800 bus timing

The 6800 bus is also called Motorola 68K bus. The 6800 bus interface consists of one chip-select line (CS), one data/command-select line (RS), one write/read-latch line (EN), one write/read-select line (R/W) and 8 or 16 bi-directional data lines (Data Bus).

CS and RS are all low-level active. Low level of the CS selects the slave device. The rising or falling edge of the EN line is a data write/read latch signal (clock). 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. R/W is a write/read select signal. A low level of R/W indicates the write operation, while a high level of R/W indicates the read operation.

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.

The figure below shows the 6800 bus writing timing. A data transfer occurs during the writing timing under a 0-beat command type.

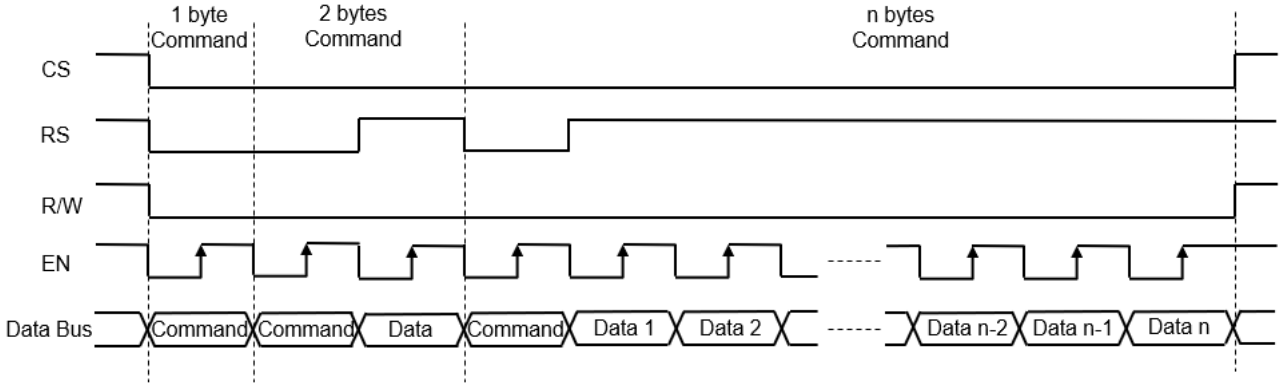

Figure 2. 6800 bus writing timing diagram

The following figure shows the reading timing. A dummy reading beat can occur between the command-writing beat and the first data-reading beat, depending on the bus slave.

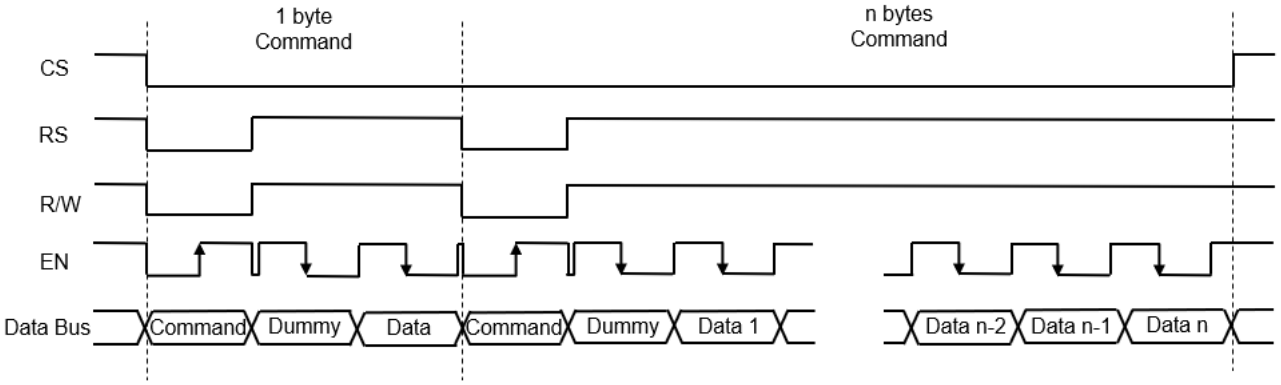

Figure 3. 6800 bus reading timing diagram

In general, any operation on the 6800 bus slave starts with a command write cycle followed by one or more data read or write cycles. To accomplish this, use the following program flow:

1. Configure FlexIO with single-beat write configuration.
2. Configure GPIO to assert CS, RS pins, and deassert R/W pin.
3. Write command data to SHIFTBUF.
4. Configure GPIO to deassert RS pin (and assert R/W pin for data read).

5. Configure FlexIO with desired read or write configuration (for example, single or 32-beats).
6. Use the Shifter Status Flag to trigger interrupt or DMA driven data transfers to/from SHIFTBUF registers.
7. Configure GPIO to deassert CS pin.

4 6800 bus emulation

This section introduces how to use the FlexIO module to emulate the 6800 parallel bus writing/reading timing diagrams shown in Figure 2 and Figure 3.

4.1 Development platform

The i.MXRT1010 EVB board is used as an example in this application to emulate the 6800 parallel bus. A pin header, J46, on the board is connected to a part of the FlexIO pins, which makes hardware connections of this application convenient. The following figure shows the i.MXRT1010-EVB development platform.

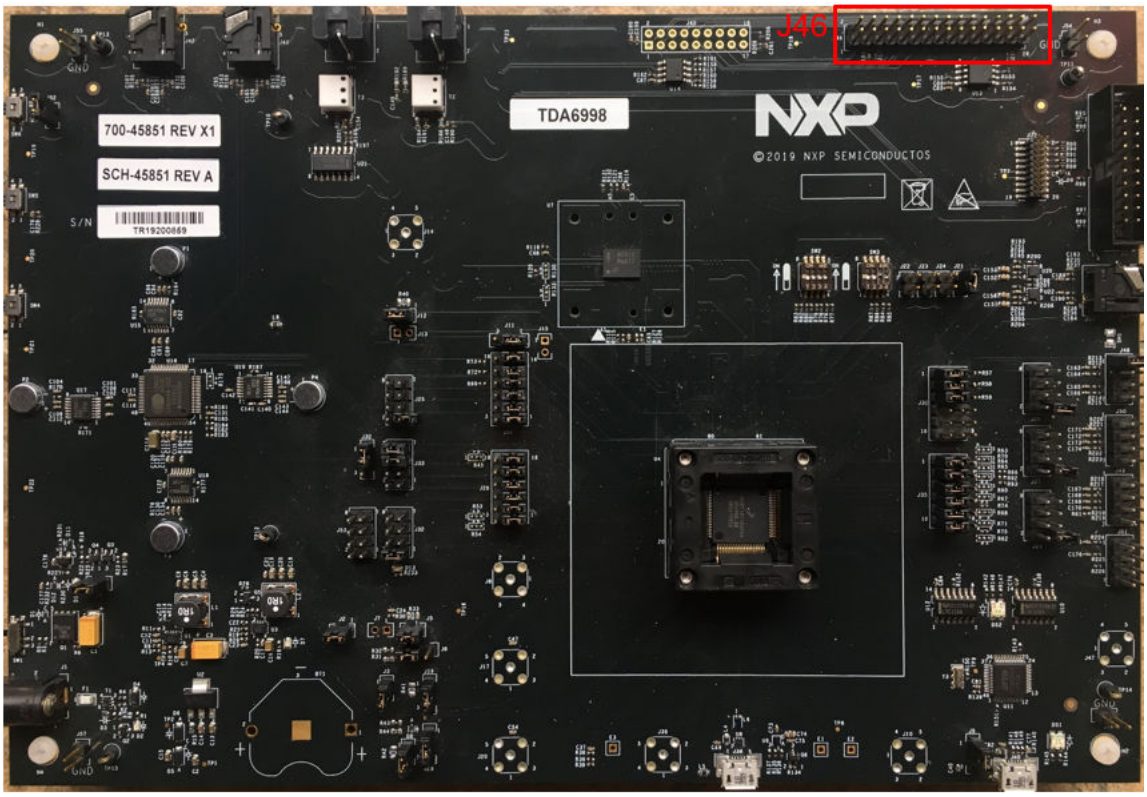

Figure 4. i.MXRT1010-EVB board

On the i.MXRT1010 board, FlexIO has a total of 27 pins. In this application, there are 9 FlexIO pins used to emulate as EN and D0~D7 pins. Also, there are 3 GPIO pins used to generate RS, CS, and R/W signals.

The table below shows the detailed pin assignment to emulate 6800 bus on the i.MXRT1010-EVB board.

Table 1. Pin assignment

Pins	Board Connector	6800 Bus Signal
FlexIO1_IO3	J46-6	D0

Table continues on the next page...

Table 1. Pin assignment (continued)

FlexIO1_IO4	J46-7	D1
FlexIO1_IO5	J46-8	D2
FlexIO1_IO6	J46-9	D3
FlexIO1_IO7	J46-10	D4
FlexIO1_IO8	J46-11	D5
FlexIO1_IO9	J46-12	D6
FlexIO1_IO10	J46-13	D7
FlexIO1_IO1	J46-3	EN
GPIO1_10	J46-4	R/W
GPIO1_8	J46-2	RS
GPIO1_7	J46-1	CS

4.2 6800 write configuration

6800 write is implemented by configuring the shifters in transmit mode. For 6800 parallel bus, write function includes single-beat write and multi-beats write. The 6800 bus can be an 8-bit or 16-bit width bus. Because module configurations are very similar between the 8-bit and 16-bit bus implementations, the following sections only describe the 8-bit implementation.

The single-beat write transmits data in small size, such as configuring the LCD driver IC's registers, smaller frame data, and command data. Only one shifter is used to shift data, and 8 bits are shifted out to assigned FlexIO pins simultaneously. One transmit timing requires the timer to generate only one shift clock. In this case, the polling method is used to drive data transfer from SHIFTBUF for a single-beat write.

The multi-beats write transmits data in large size, such as transmitting frame data to an LCD module. Concatenated shifters are used to shift data, and the number of beats per one transmit timing is related to the number of shifters and bus width. One shifter supports, at most, a 4-beats transmit for the 8-bit width bus. In this application, all 8 shifters are used, and 32-beats are supported for the 8-bit width bus. One transmit timing requires the timer to generate multiple shift clocks. In this case, DMA method is used to drive data transfer from SHIFTBUF for multi-beats write.

To emulate single-beat write timing, one Timer and one Shifter are used. Timer 0 is used to generate shift clock and EN signal. Shifter 0 transmits the data to D0~D7 pins simultaneously on each rising edge of the shift clock.

In this application, use *FLEXIO_MCULCD_SetSingleBeatWriteConfig(FLEXIO_MCULCD_Type *base)* to configure the FLEXIO MCULCD to single beat write mode.

The table below provides detailed register configuration of the FlexIO for the single-beat write of 6800 bus.

Table 2. Single-beat write configuration

Register	Value	Comments
SHIFTCFG0	0x0007_0100	Configure parallel width as 8-bit, shifter stop bit disabled, and shifter start bit disabled.

Table continues on the next page...

Table 2. Single-beat write configuration (continued)

SHIFTCTL0	0x0003_0302	Configure transmit mode, using Timer 0 to generate shifter clock, and data output to FlexIO1_[10:3] pins on posedge of shifter clock.
TIMCMP0	0x0000_0105	TIMCMP[15:8] = (number of beats x 2) – 1= (1 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) – 1 Note: Baud rate divider = FlexIO frequency/baud rate
TIMCFG0	0x0000_2200	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, timer never reset, disabled on timer compare, enabled on trigger high, stop bit disabled, and start bit disabled.
TIMCTL0	0x01C3_0181	Configure Shifter 0 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 1 (EN), pin polarity active low, timer mode as dual 8-bit counters baud/bit.

To emulate multi-beats write timing, one Timer and 8 Shifters are used. Timer 0 is used to generate shift clock and EN signal. Shifters 0~7 transmit the data to D0~D7 pins on each rising edge of the shift clock. Additional GPIO pins drive CS, RS, and R/W signals. The figure below shows the FlexIO module configuration for multi-beats write.

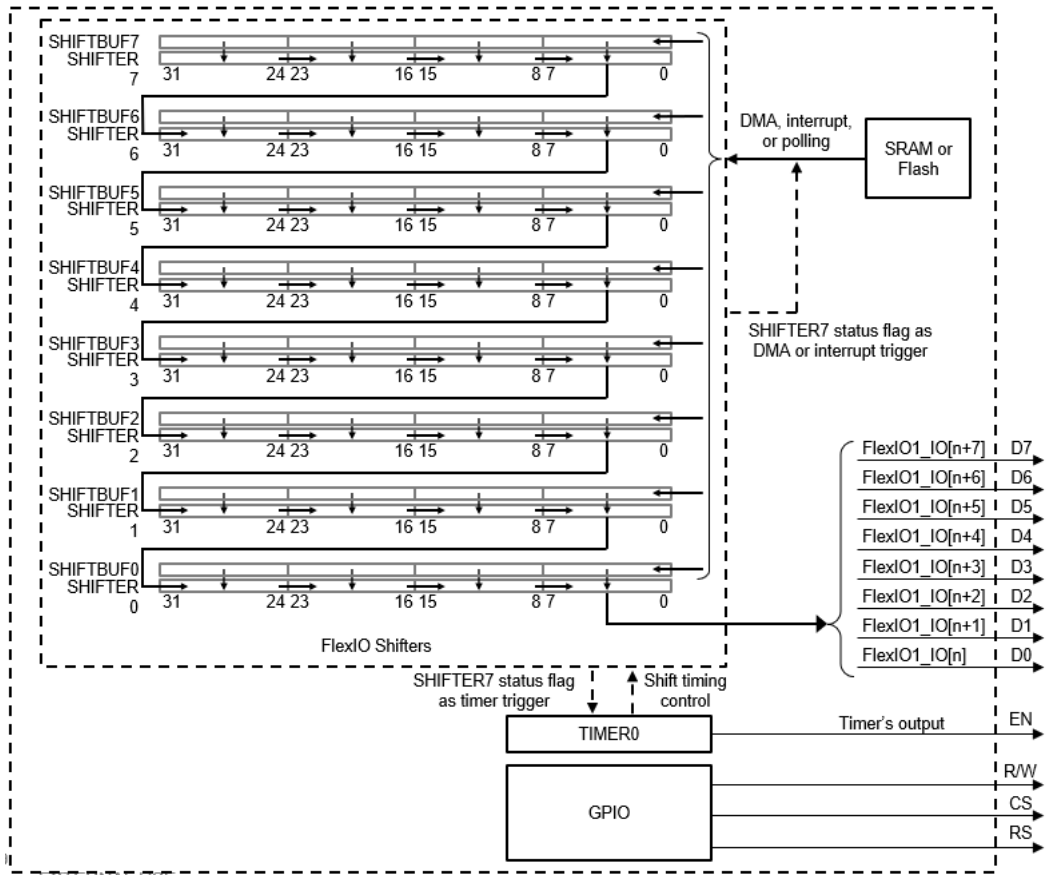

Figure 5. Multi-beats write configuration

In this application, use *FLEXIO_MCULCD_SetMultiBeatsWriteConfig(FLEXIO_MCULCD_Type *base)* to configure the FLEXIO MCULCD to multiple beats write mode.

The table below provides detailed register configuration of the FlexIO for the multi-beats write.

Table 3. Multi-beats write configuration

Register	Value	Comments
SHIFTCFG0~7	0x0007_0100	Configure parallel width as 8-bit, shifter stop bit disabled, and shifter start bit disabled, shifter input from Shifter N+1 output.
SHIFTCTL0	0x0003_0302	Configure transmit mode, using Timer 0 to generate shifter clock, and data output to FlexIO1_[10:3] pins on posedge of shifter clock.
SHIFTCTL1~7	0x0000_0002	Configure transmit mode, using Timer 0 to generate shifter clock, and shifter pin output disabled
TIMCMP0	0x0000_3F05	TIMCMP[15:8] = (number of beats x 2) – 1= (32 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) – 1 Note: Baud rate divider = FlexIO frequency/baud rate
TIMCFG0	0x0000_2200	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, timer never reset, disabled on timer compare, enabled on trigger high, stop bit disabled, and start bit disabled.
TIMCTL0	0x1DC3_0181	Configure Shifter 7 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 1(EN), pin polarity active low, timer mode as dual 8-bit counters baud/bit.

4.3 6800 read configuration

6800 read is implemented by configuring shifters in receive mode. For 6800 parallel bus, the read function includes single-beat read and multi-beats read.

Similar to 6800 write introduced before, the single-beat read receives data in small size. Only one shifter is used to shifter data, and 8 bits are shifted in from assigned FlexIO pins simultaneously. One receive timing requires the timer to generate only one shift clock. In this case, the polling method is used to load data to SHIFTBUF for a single-beat read.

The multi-beats read receives data in large size. Similar to 6800 multi-beats write, concatenated shifters are used to receive data. One receive timing requires the timer to generate multiple shift clocks. In this case, the DMA method is used to load data to SHIFTBUF for multi-beats read.

To emulate single-beat read timing, one Timer and one Shifter are used. Timer 0 is used to generate shift clock and EN signal. Shifter 7 shifts the data from D0~D7 pins on each falling edge of the shift clock.

In this application, use *FLEXIO_MCULCD_SetSingleBeatReadConfig(FLEXIO_MCULCD_Type *base)* to configure the FLEXIO MCULCD to single beat read mode.

The table below provides detailed register configuration of the FlexIO for the single-beat read.

Table 4. Single-beat read configuration

Register	Value	Comments
SHIFTCFG7	0x0007_0000	Configure parallel width as 8-bit, shifter stop bit disabled, shifter start bit disabled, shifter input from pin.

Table continues on the next page...

Table 4. Single-beat read configuration (continued)

SHIFTCTL7	0x0080_0301	Configure receive mode, using Timer 0 to generate shifter clock, and data input from FlexIO1_[10:3] pins on negedge of shifter clock.
TIMCMP0	0x0000_0105	TIMCMP[15:8] = (number of beats x 2) – 1= (1 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) – 1 Note: Baud rate divider = FlexIO frequency/baud rate
TIMCFG0	0x0000_2220	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, timer 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_0101	Configure Shifter 7 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 1 (EN), pin polarity active high, timer mode as dual 8-bit counters baud/bit.

To emulate multi-beats read timing, one Timer and 8 Shifters are used. Timer 0 is used to generate shift clock and EN signal. Shifters 0~7 shift the data from D0~D7 pins on each falling edge of the shift clock. Additional GPIO pins drive CS, RS, and R/W signals. The figure below shows the FlexIO module configuration for multi-beats read.

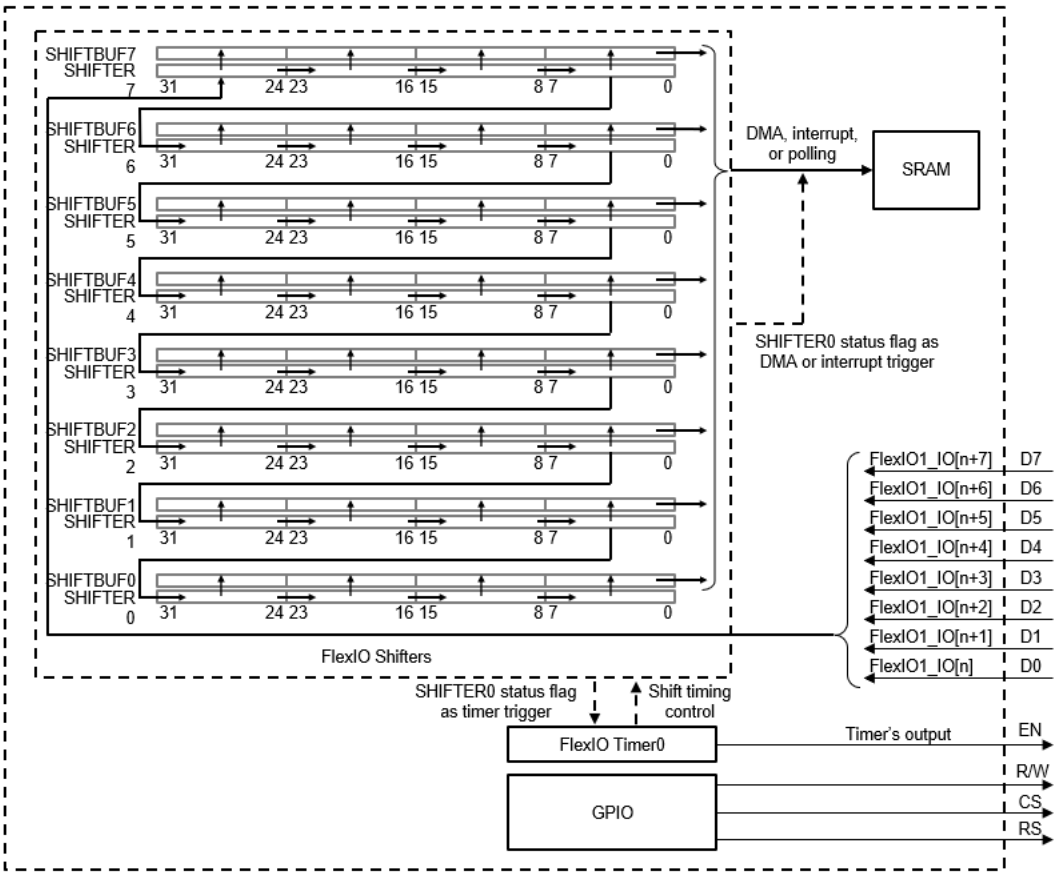

Figure 6. Multi-beats read configuration

In this application, use *FLEXIO_MCULCD_SetMultiBeatsReadConfig(FLEXIO_MCULCD_Type *base)* to configure the FLEXIO MCULCD to multiple beats read mode.

The table below provides detailed register configuration of the FlexIO for the multi-beats read.

Table 5. Multi-beats read configuration

Register	Value	Comments
SHIFTCFG0~6	0x0007_0100	Configure parallel width as 8-bit, shifter stop bit disabled, and shifter start bit disabled, shifter input from Shifter N+1 output.
SHIFTCFG7	0x0007_0000	Configure parallel width as 8-bit, shifter stop bit disabled, and shifter start bit disabled, shifter input from pin.
SHIFTCTL0~7	0x0080_0301	Configure as receive mode, using Timer 0 to generate shifter clock, and data input from FlexIO1_[10:3] pins on negedge of shifter clock.
TIMCMP0	0x0000_3F05	TIMCMP[15:8] = (number of beats x 2) – 1= (32 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) – 1 Note: Baud rate divider = FlexIO frequency/baud rate
TIMCFG0	0x0000_2220	Configure timer output logic one when enabled and not affected by reset, decrement on FlexIO clock, timer 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_0101	Configure Shifter 0 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 1 (EN), pin polarity active high, timer mode as dual 8-bit counters baud/bit.

4.4 Run the demo

Since it is difficult to get an LCD module driven by the 6800 bus, a logic analyzer is used to capture the actual bus signals to verify the 6800 bus timing. The logic analyzer is connected to RT1010-EVB board according to the connection method in Table 1.

Users can download the software in nxp.com. Find the IAR project, *flexio_6800*, and build, download, and run the demo on i.MXRT1010-EVB board. Then, you can see waveforms of all signals captured by the logic analyzer. The connection diagram between board and logic analyzer is as shown in the following figure.

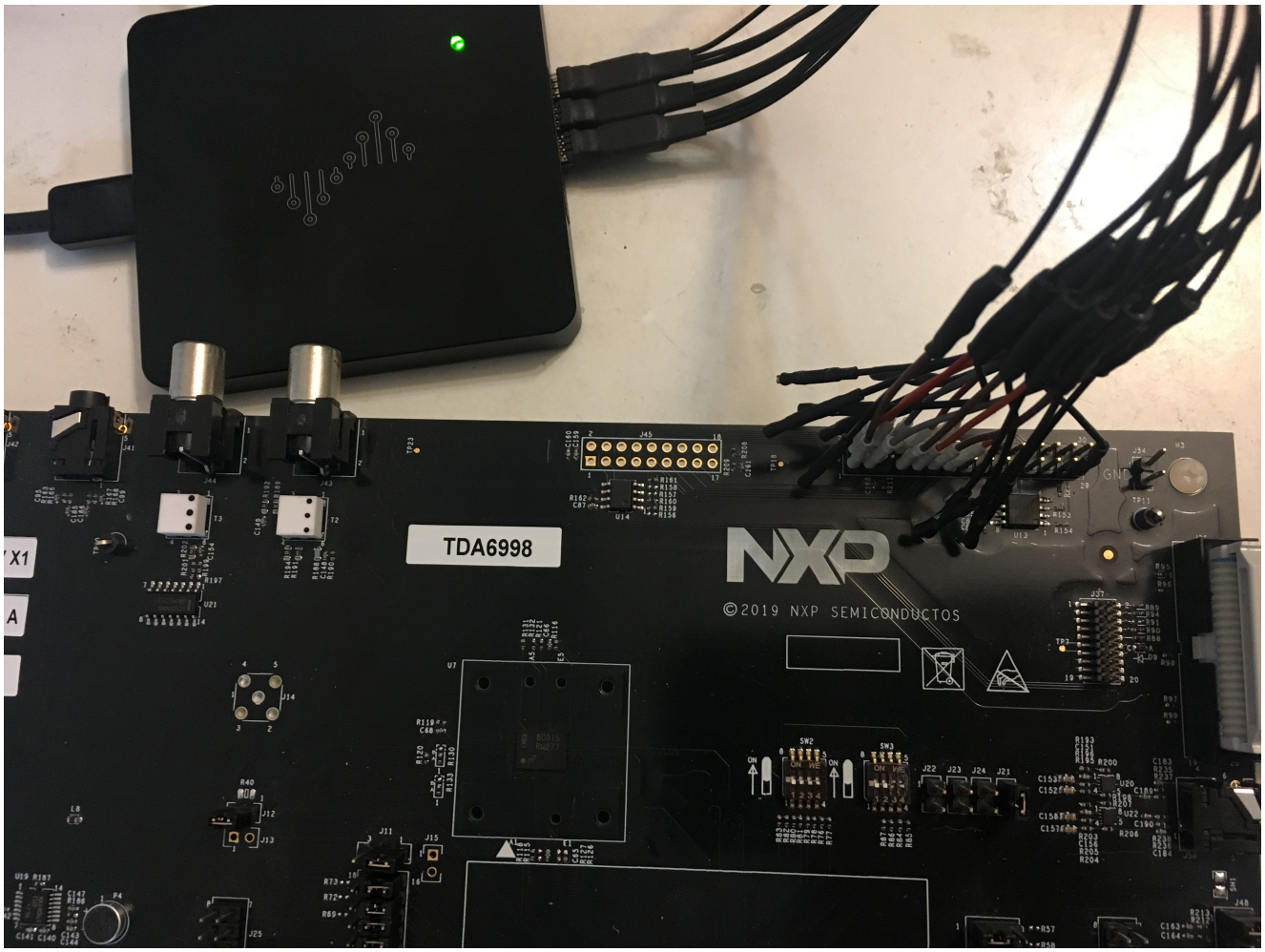

Figure 7. Hardware connection

The figure below shows the waveform of 6800 write. From the picture, you can see that the 6800 bus write begins with a command write cycle followed by one or more data write cycles.

When in write progress, the R/W signal is always low level. Before the writing of data starts, the CS signal should be low level, and after the writing of data completes, the CS should be high level. The RS signal should be low level when writing a command, while it is high level when writing data. Writing a command uses the single-beat write, command is shifted out to FlexIO1_[10:3] pins on posedge of one shift clock(generated by EN signal). Writing data uses single-beat or multi-beats write; also, data is shifted out to FlexIO1_[10:3] pins on posedge of one or more shift clocks.

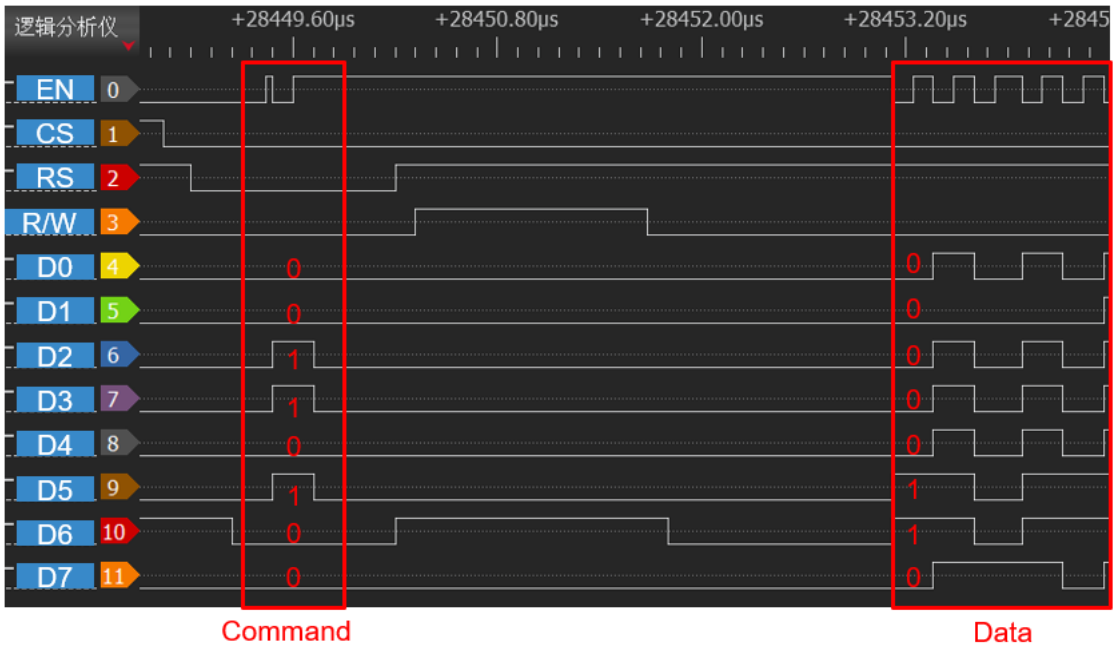

Figure 8. 6800 write

The figure below shows the waveform of 6800 read. Similarly, the 6800 bus read begins with a command write cycle followed by one or more data read cycles.

When in read progress, the R/W signal is always high level. Before the reading of data starts, the CS signal should be low level, and after the reading of data completes, the CS should be high level. The RS signal should be low level when writing a command, while it is high level when reading data. Writing a command uses the single-beat write, command is shifted out to FlexIO1_[10:3] pins on posedge of one shift clock(generated by EN signal). Reading data uses single-beat or multi-beats read; also, data is shifted in from FlexIO1_[10:3] pins on negedge of one or more shift clocks.

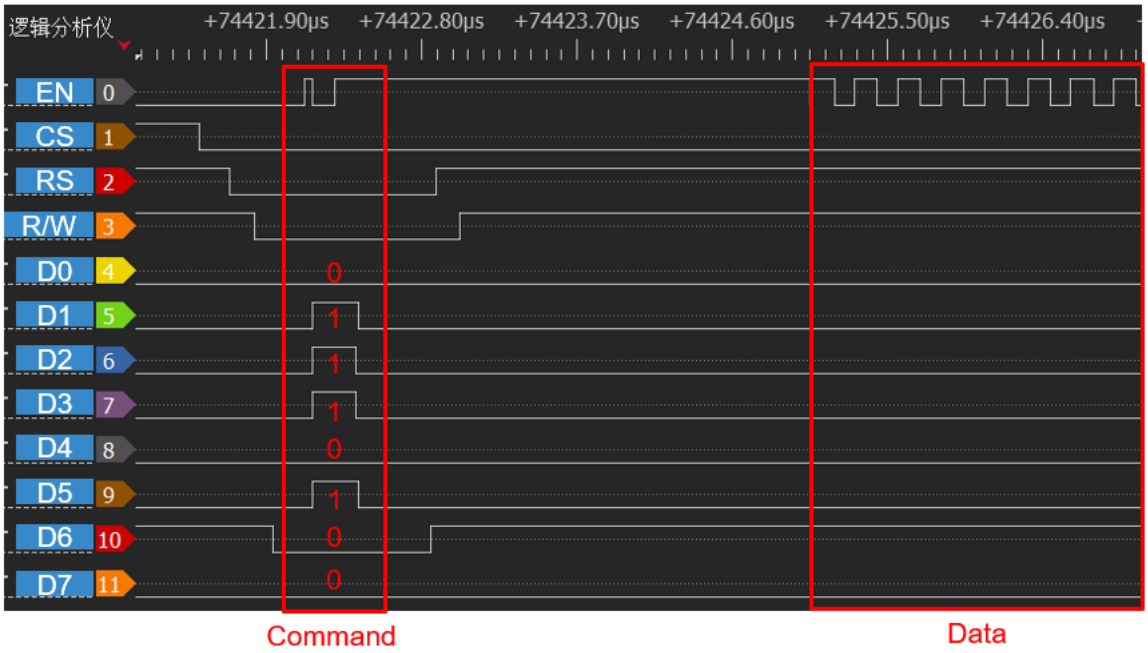

Figure 9. 6800 read

5 Conclusion

This application note introduces an example of the 6800 parallel bus that can be emulated via the FlexIO module provided by the RT1010 MCU. In addition, the emulated 6800 bus timing can be well observed by the logic analyzer.

6 References

1. i.MX RT1010 Processor Reference Manual (Rev. 0, 09/2019).
2. Using FlexIO to Drive 8080 Bus Interface LCD Module (document AN5313).

7 Revision history

Table 6. Revision history

Revision number	Date	Substantive changes
0	2/2020	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP 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. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 15 April 2020

Document identifier: AN12813