AN12822
Emulating 8080 Bus with the FlexIO on RT1050

Rev. 0 — April, 2020

Application Note

Contents

1 Introduction

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

FlexIO is an on-chip peripheral available on NXP i.MXRT series. It is a highly configurable module which is capable of emulating various serial/parallel communication protocols, such as UART, I2C, and SPI. Users can also use FlexIO to generate 8080 bus.

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.

The i.MX RT1050 is a new processor family featuring NXP's advanced implementation of the high performance Arm Cortex-M7 Core. It offers high-performance processing optimized for lowest power consumption and best real-time response. In order to verify the 8080 bus emulated via FlexIO, an application was implemented on the i.MX RT1050-EVB board.

2 FlexIO overview

2.1 Features

The FlexIO module of the i.MX RT1050 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 triggers, 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 gives a high-level overview of the configuration of FlexIO timers and shifters.

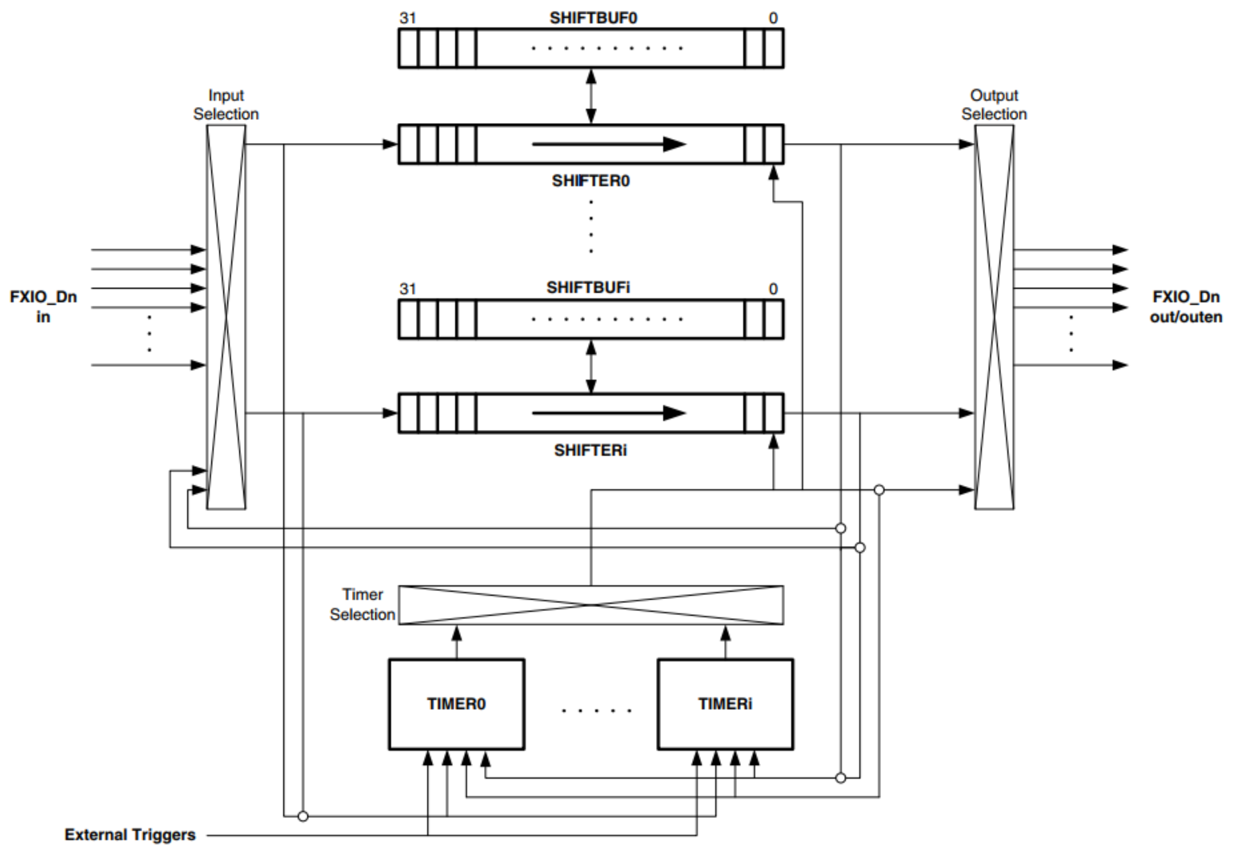

Figure 1. FlexIO block diagram

There are both FlexIO1 and FlexIO2 on i.MX RT1050. FlexIO1 and FlexIO2 parameters are not the same. FlexIO1 has 16 pins, while FlexIO2 has 32 pins. Also, both FlexIO1 and FlexIO2 have 4 shifters and 4 timers. Shifters are responsible for buffering and shifting data into or out of the FlexIO. 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.MXRT1050 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 and used to configure 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 for concatenation. DMA method is used to access the shifter buffer registers for high-speed transfers and low-power operations.

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

Similarly, for parallel receive, only SHIFTER3 supports inputting from FlexIO pins. However, all shifters, except the SHIFTER3, 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 pin15, or pin1 to pin16, and so on for 16-bit width bus.

3 8080 Bus Timing

8080 bus is also called Intel 8080 bus. Generally, the 8080 bus interface consists of one chip-select line (CS), one writing-latch line (WR), one reading-latch line (RD), one data/command-select line (RS), and 8 or 16 bidirectional data lines (Data Bus).

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 timing is in progress. Similarly, WR should be high-level when a reading timing 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.

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 2 shows the 8080 bus writing timing. A data transfer occurs during the writing timing under a 0-beat command type.

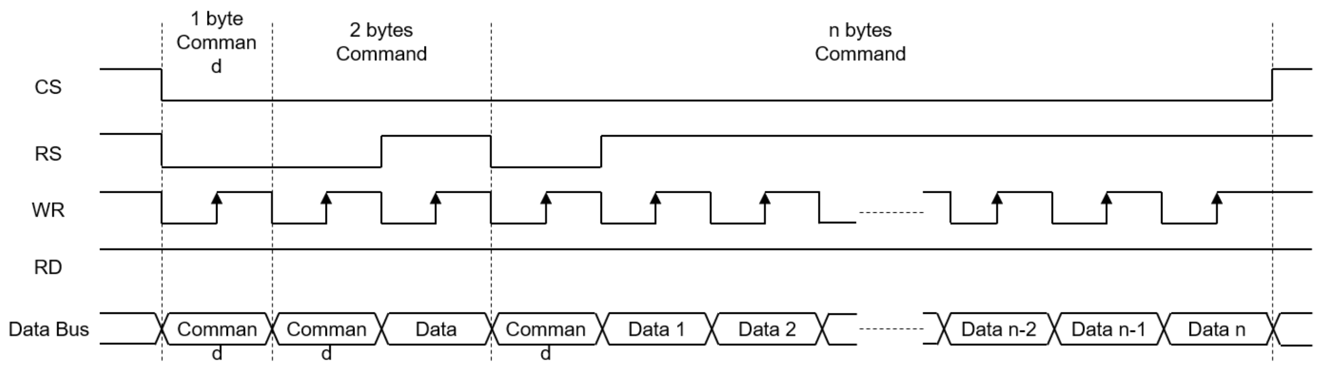

Figure 2. 8080 bus writing timing diagram

Figure 3 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.

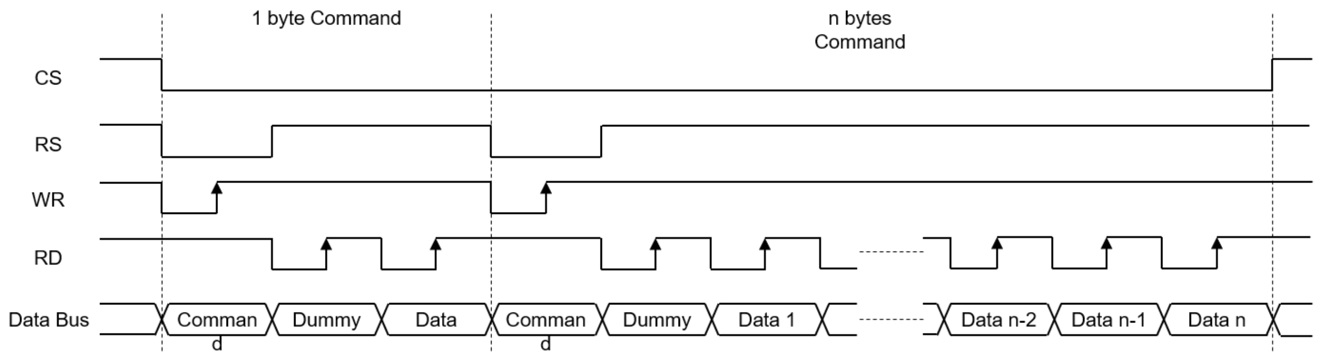

Figure 3. 8080 bus reading timing diagram

In general, any operation to the 8080 bus slave begins with a command write cycle followed by one or more data read or write cycles. To accomplish this, the following program flow is used:

1. Configure FlexIO with single-beat write configuration
2. Configure GPIO to assert CS, RS pins
3. Write command data to SHIFTBUF

4. Configure GPIO to deassert RS pin
5. Configure FlexIO with desired read or write configuration (for example, single or 8-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 8080 bus emulation

This chapter introduces the use of the FlexIO module to emulate the 8080 parallel bus writing/reading timing diagrams shown in Figure 2 and Figure 3 and to drive a TFT LCD module.

4.1 Development platform

In order to emulate the 8080 parallel bus, i.MXRT1050 EVB board is used as an example in this application. There are two pin headers, J60 and J74, on the board, connected to the FlexIO pins, which makes it convenient for the hardware connections of this application. The following figure shows the i.MX RT1050-EVB development platform.

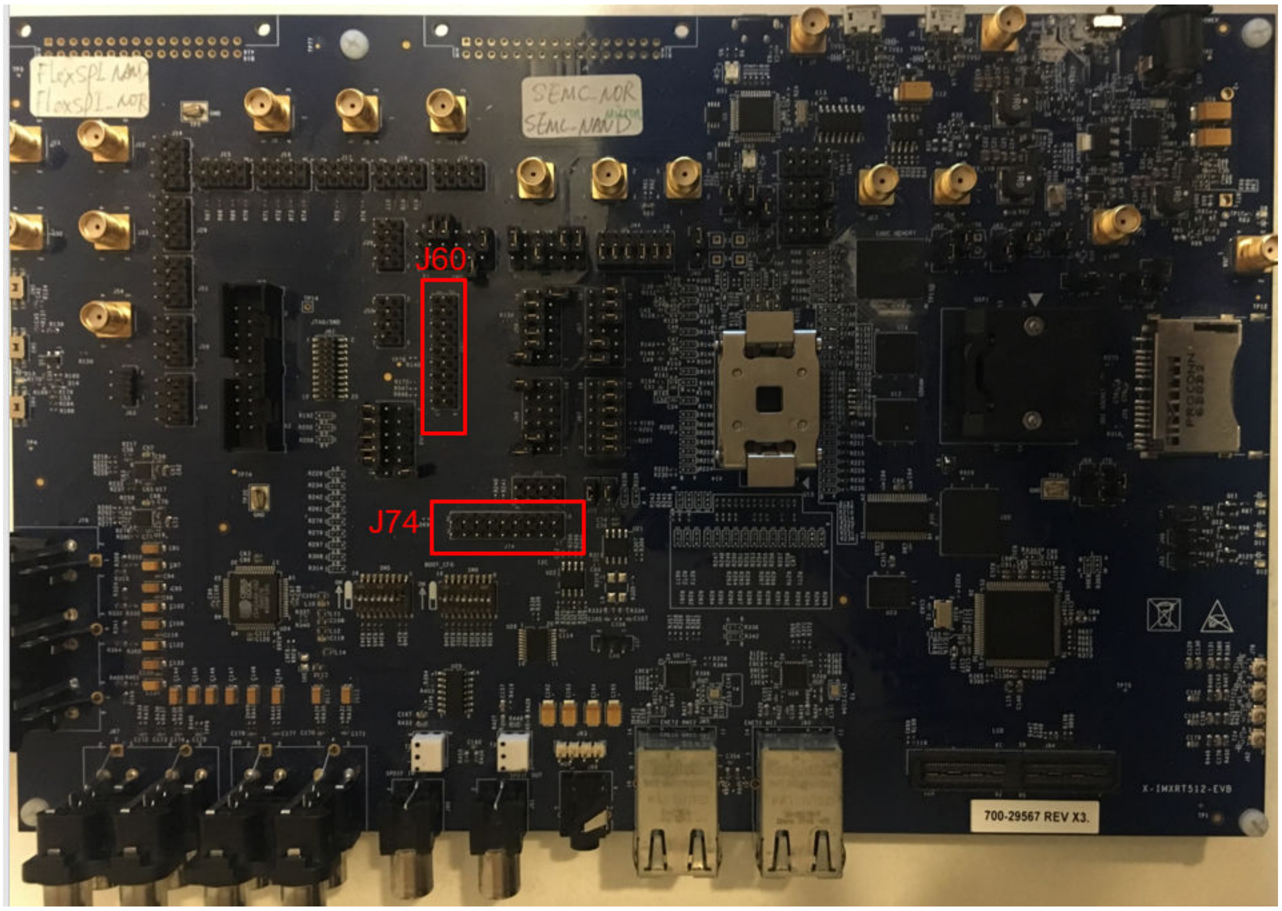

Figure 4. i.MXRT1050-EVB board

The TFT LCD module used in this application integrates a Himax LCD driver IC HX8357. It supports many data transfer interfaces. Because module configurations are similar between 8080 compatible 8-bit and 16-bit MIPI-DBI parallel bus, so the following sections only describe 16-bit bus implementation in this application. The following figure shows the system block diagram of this application.

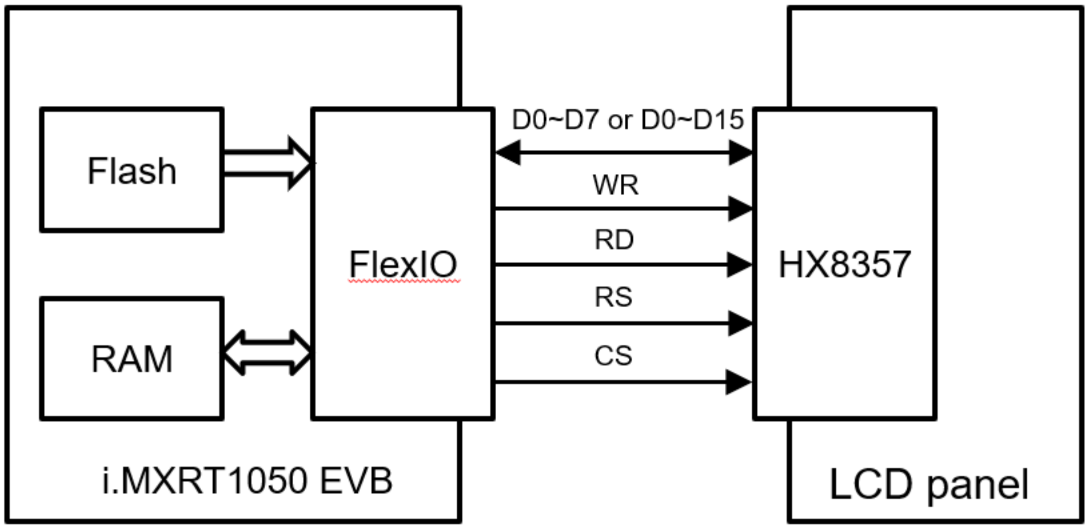

Figure 5. System block diagram

On the i.MX RT1050, FlexIO2 has a total of 32 pins. In this application, there are 18 FlexIO pins used to emulate WR, RS, and D0~D15 pins. Also, there are two GPIO pins used to generate RS and CS signals.

The following table shows the detailed hardware connections between the i.MXRT1050-EVB board and the LCD module.

Table 1. Hardware connections

Pins	Board Connector	LCD Signal
FlexIO2_12	J60-10	D0
FlexIO2_13	J60-8	D1
FlexIO2_14	J60-6	D2
FlexIO2_15	J60-4	D3
FlexIO2_16	J74-3	D4
FlexIO2_17	J74-5	D5
FlexIO2_18	J74-7	D6
FlexIO2_19	J74-9	D7
FlexIO2_20	J74-11	D8
FlexIO2_21	J74-13	D9
FlexIO2_22	J74-15	D10
FlexIO2_23	J74-17	D11
FlexIO2_24	J74-18	D12

Table continues on the next page...

Table 1. Hardware connections (continued)

Pins	Board Connector	LCD Signal
FlexIO2_25	J74-16	D13
FlexIO2_26	J74-14	D14
FlexIO2_27	J74-12	D15
FlexIO2_00	J60-3	WR
FlexIO2_01	J60-5	RD
GPIO2_02	J60-7	RS
GPIO2_03	J60-9	CS
GND	J60-2	GND
VCC	J60-1	3V3

4.2 8080 write configuration

8080 write can be implement by configuring shifters in transmit mode. For 8080 parallel bus, write function includes single-beat write and multi-beats write.

The single-beat write transmits small size data, such as configuring the LCD driver IC’s registers, smaller frame data, and command data. Only one shifter is used to shifter data, and 16 bits are shifted out to assigned FlexIO pins simultaneously. One transmit timing requires the timer to generate only one shift clock. In this case, polling method is used to drive data transfer from SHIFTBUF for single-beat write.

The multi-beats write transmits large size data, such as transmitting frame data to an LCD module. Concatenated shifters are used to shifter 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 2-beat transmit for the 16-bit width bus. In this application, all four shifters are used and 8-beats are supported for the 16-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, a total of one Timer, one Shifter are used. Timer 0 is used to generate a shift clock and a WR signal. Shifter 0 transmits the data to D0~D15 pins 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 following table provides detailed register configuration of the FlexIO for the single-beat write.

Table 2. Single-beat write configuration

Register	Value	Comments
SHIFTCFG0	0x000F_0100	Configure parallel width as 16-bit, shifter stop bit disabled, and shifter start bit disabled.
SHIFTCTL0	0x0003_0C02	Configure transmit mode, using Timer 0 to generate shifter clock, and data output to FlexIO2_[27:12] pins on positive edge of shifter clock.

Table continues on the next page...

Table 2. Single-beat write configuration (continued)

Register	Value	Comments
TIMCMP0	0x0000_0105	TIMCMP[15:8] = (number of beats x 2) – 1= (1 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) - 1
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_0081	Configure Shifter 0 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 0 (WR), pin polarity active low, timer mode as dual 8-bit counters baud/bit.

To emulate multi-beats write timing, a total of one Timer, four Shifters are used. Timer 0 is used to generate a shift clock and a WR signal. Shifters 0~3 transmit the data to D0~D15 pins on each rising edge of the shift clock. Additional GPIO pins drive CS and RS signals. The following figure shows the FlexIO module configuration for multi-beats write.

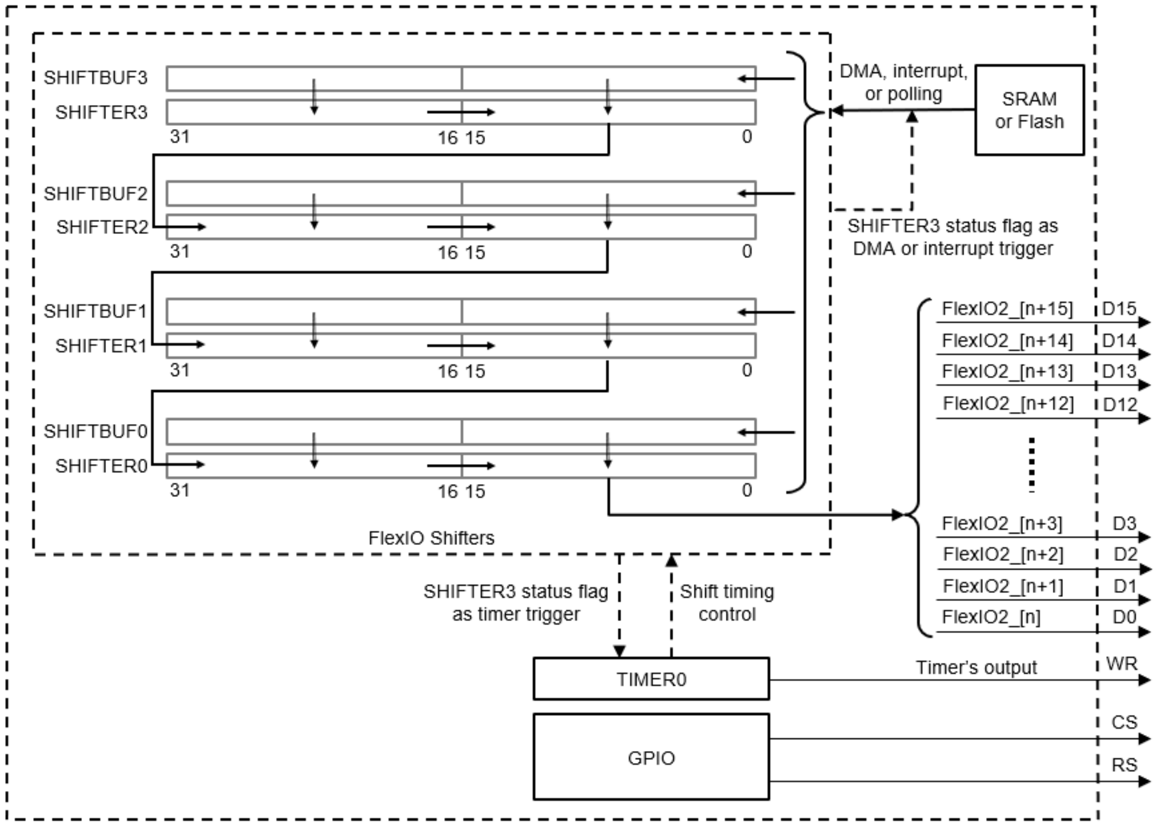

Figure 6. Multi-beats write configuration

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

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

Table 3. Multi-beats write configuration

Register	Value	Comments
SHIFTCFG0~3	0x000F_0100	Configure parallel width as 16-bit, shifter stop bit disabled, and shifter start bit disabled, shifter input from Shifter N+1 output.
SHIFTCTL0	0x0003_0C02	Configure transmit mode, using Timer 0 to generate shifter clock, and data output to FlexIO2_[27:12] pins on positive edge of shifter clock.
SHIFTCTL1~3	0x0000_0002	Configure transmit mode, using Timer 0 to generate shifter clock, and shifter pin output disabled
TIMCMP0	0x0000_0F05	TIMCMP[15:8] = (number of beats x 2) – 1= (8 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) - 1
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	0x0DC3_0081	Configure Shifter 3 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 0 (WR), pin polarity active low, timer mode as dual 8-bit counters baud/bit.

4.3 8080 read configuration

8080 read can be implement by configuring shifters in receive mode. For 8080 parallel bus, read function includes single-beat read and multi-beats read.

Similar to 8080 write introduced before, the single-beat read receives small size data. Only one shifter is used to shifter data, and 16 bits are shifted in from assigned FlexIO pins simultaneously. One receive timing requires the timer to generate only one shift clock. In this case, polling method is used to load data to SHIFTBUF for single-beat read.

The multi-beats read receives large size data. Similar to 8080 multi-beats write, concatenated shifters are used to receive data. One receive timing requires the timer to generate multiple shift clocks. In this case, DMA method is used to load data to SHIFTBUF for multi-beats read.

To emulate single-beat read timing, a total of one Timer, one Shifter are used. Timer 0 is used to generate shift clock and RD signal. Shifter 3 shifts the data in from D0~D15 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 following table provides detailed register configuration of the FlexIO for the single-beat read.

Table 4. Single-beat read configuration

Register	Value	Comments
SHIFTCFG3	0x000F_0000	Configure parallel width as 16-bit, shifter stop bit disabled, shifter start bit disabled, shifter input from pin.
SHIFTCTL3	0x0080_0C01	Configure receive mode, using Timer 0 to generate shifter clock, and data input from FlexIO2_[27:12] pins on negative edge of shifter clock.

Table continues on the next page...

Table 4. Single-beat read configuration (continued)

Register	Value	Comments
TIMCMP0	0x0000_0105	TIMCMP[15:8] = (number of beats x 2) – 1= (1 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) - 1
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	0x0DC3_0181	Configure Shifter 3 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 1 (RD), pin polarity active low, timer mode as dual 8-bit counters baud/bit.

To emulate multi-beats read timing, a total of one Timer, four Shifters are used. Timer 0 is used to generate a shift clock and an RD signal. Shifters 0~3 shift the data in from D0~D15 pins on each falling edge of the shift clock. Additional GPIO pins drive CS and RS signals. The following figure shows the FlexIO module configuration for multi-beats read.

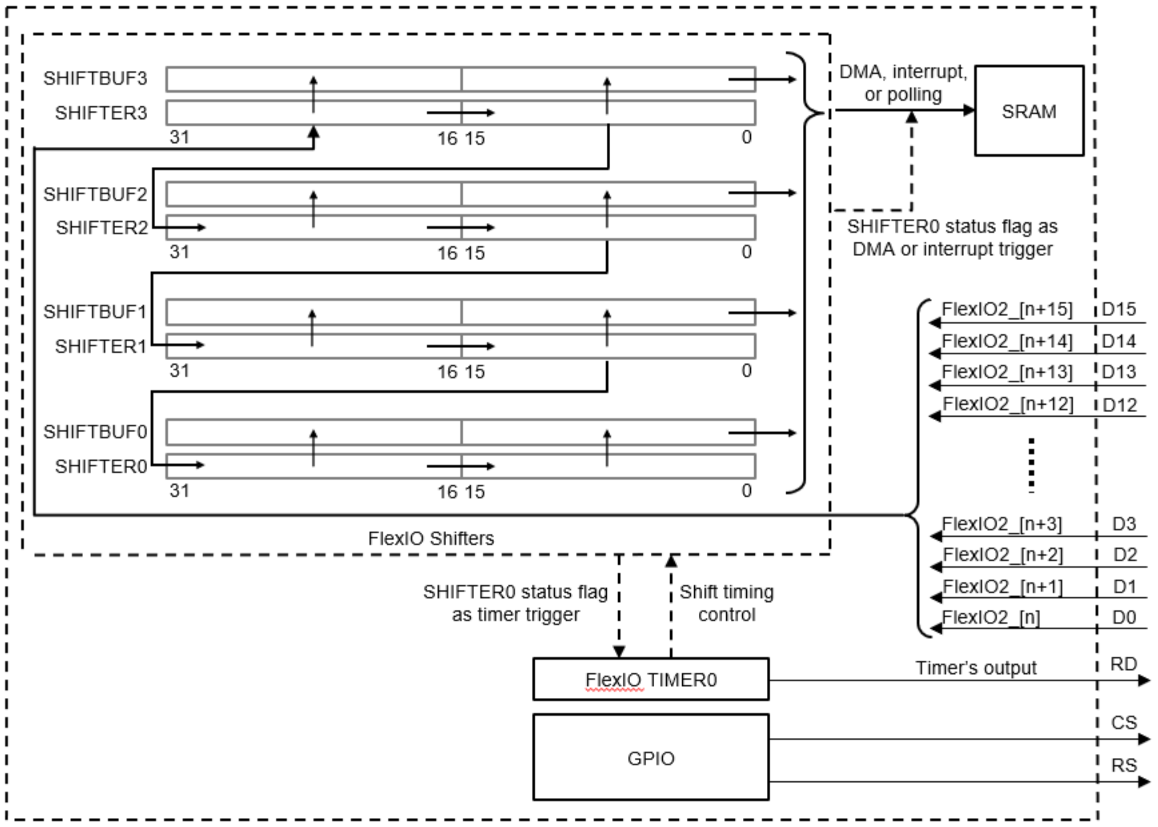

Figure 7. Multi-beats read configuration

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

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

Table 5. Multi-beats read configuration

Register	Value	Comments
SHIFTCFG0~2	0x000F_0100	Configure parallel width as 16-bit, shifter stop bit disabled, and shifter start bit disabled, shifter input from Shifter N+1 output.
SHIFTCFG3	0x000F_0000	Configure parallel width as 16-bit, shifter stop bit disabled, and shifter start bit disabled, shifter input from pin.
SHIFTCTL0~3	0x0080_0C01	Configure as receive mode, using Timer 0 to generate shifter clock, and data input from FlexIO2_[27:12] pins on negative edge of shifter clock.
TIMCMP0	0x0000_0F05	TIMCMP[15:8] = (number of beats x 2) – 1= (8 x 2) – 1 TIMCMP[7:0] = (baud rate divider / 2) - 1
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_0181	Configure Shifter 0 status flag as timer internal trigger, trigger polarity active low, timer's pin as output, pin index as 1 (RD), pin polarity active low, timer mode as dual 8-bit counters baud/bit.

In this application, the single-beat read and multi-beats read are not used.

4.4 Run the demo

Users can download the software from nxp.com. Find the IAR project *flexio_8080_lcd*, build, download, and run the demo on i.MXRT1050-EVB to drive an LCD module. Then, you can see a picture of NXP logo displayed on the LCD screen, as shown in the following figure.

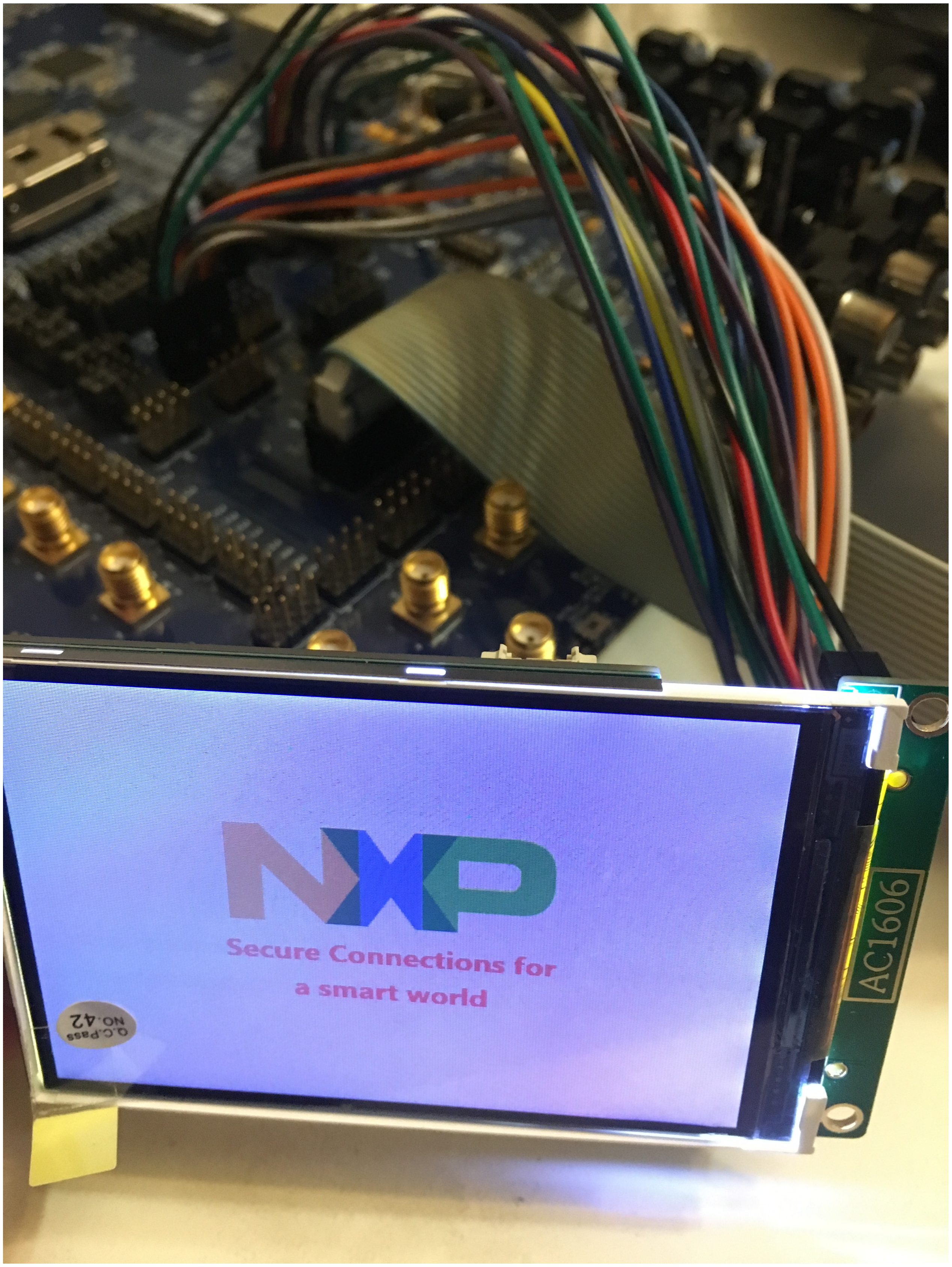

Figure 8. LCD display diagram

In order to learn 8080 bus timing, capture the actual bus signals with a logic analyzer. The following shows the waveform of single-beat write when writing a command ‘0x2C’. From the picture you can see, when in write mode, the RD signal is always high-level. Before sending the command, the CS and RS signals should be low-level. Then, data is shifted out to FlexIO2_[27:12] pins on positive edge of WR signal.

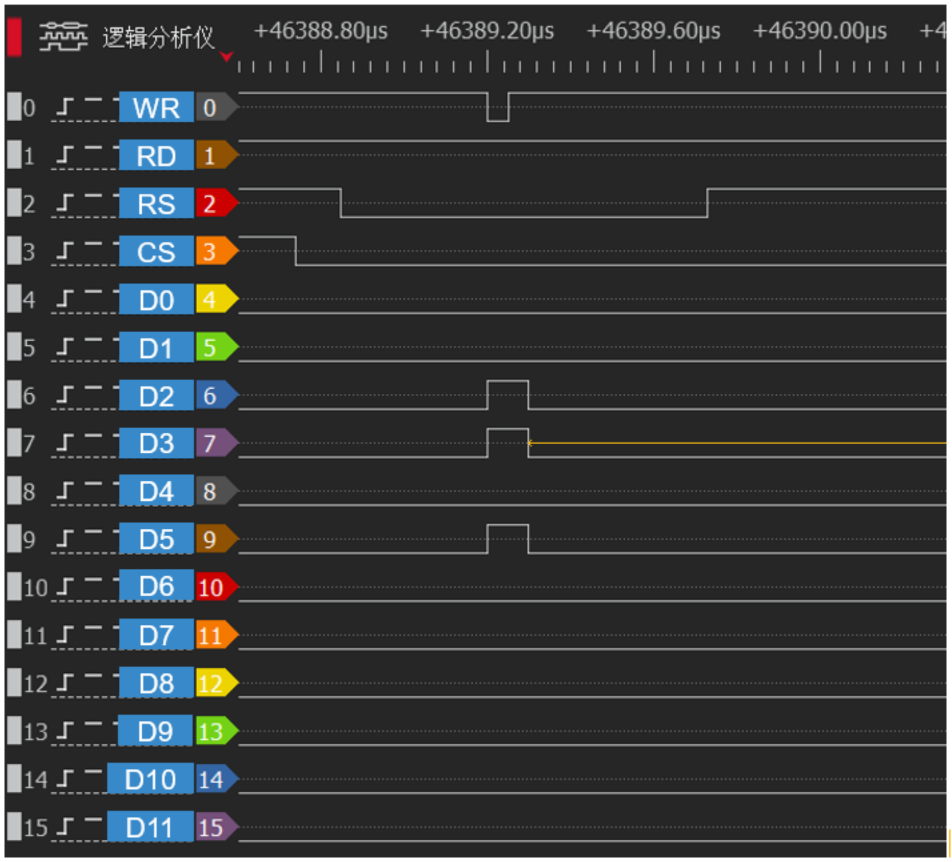

Figure 9. Command write

The following figure shows the waveform of multi-beats write when writing frame data to the LCD. Similarly, when in write mode, the RD signal is always high-level. When sending the data, the CS signal is low-level while the RS signal is high-level. Also, data is shifted out to FlexIO2_[27:12] pins on positive edge of WR signal.

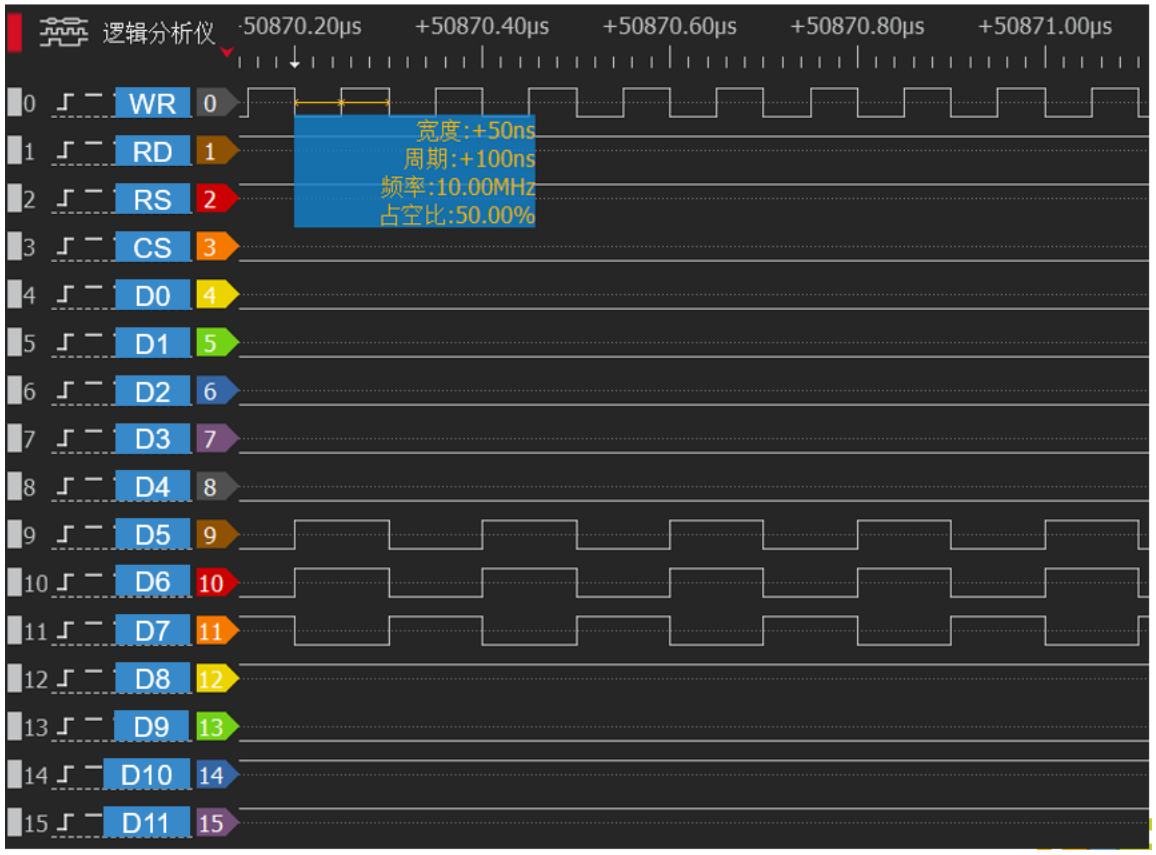

Figure 10. Frame data write

5 Conclusion

This application note introduces an example of the 8080 parallel bus that can be emulated via the FlexIO module provided by the RT1050 MCU. In addition, a TFT LCD can be driven well by the 8080 bus emulated via the FlexIO module.

6 References

1. i.MX RT1050 Processor Reference Manual (Rev. 4, 12/2019)
2. Using FlexIO to Drive 8080 Bus Interface LCD Module (AN5313)

7 Revision history

Table 6. Revision history

Revision number	Date	Substantive changes
0	04/2020	Initial release

Date of release: April, 2020

Document identifier: AN12822