AN12721

Emulating IRDA with the FlexIO on RT1010

Rev. 0 — 31 January 2020

Application Note

Contents

1 FLEXIO overview....1
2 IrDA overview....1
3 IrDA by FLEXIO....2
3.1 How to generate carrier PWM by FLEXIO....3
3.2 How to configure an NRZ encoder by FLEXIO....3
3.3 How to configure an NRZ decoder by FLEXIO....4
3.4 Summary for NRZ data encoding and decoding....5
4 How to develop it based on RT1010 EVK....7
5 Revision history....8

1 FLEXIO overview

This application note describes how to use FlexIO module to emulate IrDA protocol based on RT1010. Although RT1010 LPUART supports IrDA function, FLEXIO emulation is an option when LPUART is not used.

FlexIO is an on-chip peripheral available on NXP i.MX RT series. It is a highly configurable module capable of emulating a wide range of communication protocols, such as UART, I2C, SPI, and I2S. Users can also use FlexIO to generate PWM and PFM waveform.

For this application note, it is based on the function of the FLEXIO UART, using a timer to implement the encoding and decoding of NRZ data.

2 IrDA overview

IrDA is a standardized wireless infrared data communication method. To reduce the bit error rate, NRZ format data is used. At the same time, the transmitted data should be mixed with the carrier. Carrier frequency is 38 kHz. The circuit in the Figure 2 shows how to mix the transmitted data and carrier. Infrared receiver is used to receive mixed data and then the source data is used by MCU.

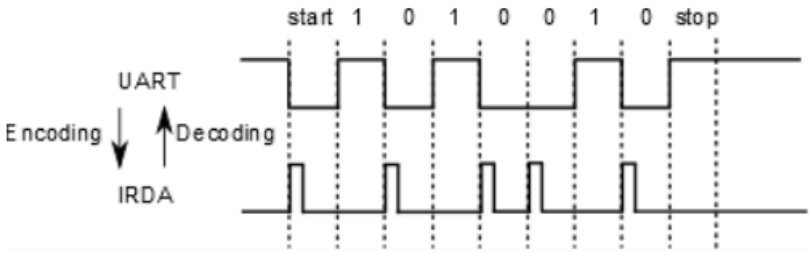

Figure 1. UART data and NRZ data

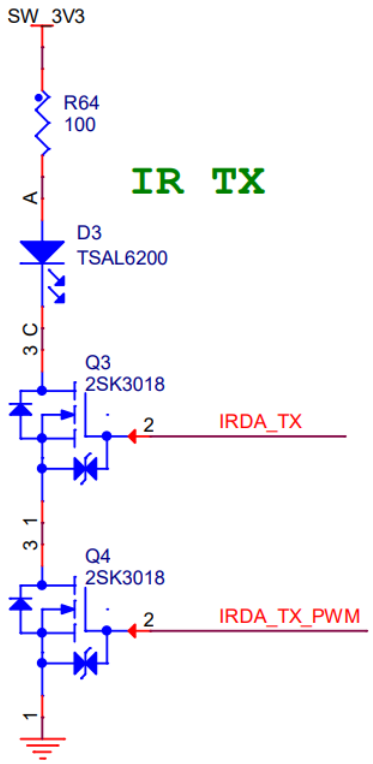

Figure 2. IrDA TX circuit

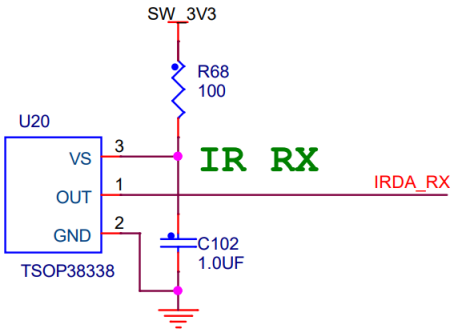

Figure 3. IrDA RX circuit

3 IrDA by FLEXIO

- How to generate carrier PWM by FLEXIO
- How to configure an NRZ encoder by FLEXIO
- How to configure an NRZ decoder by FLEXIO
- Summary for NRZ data encoding and decoding

3.1 How to generate carrier PWM by FLEXIO

As the IrDA TX Circuit shows, a PWM is needed as a carrier. The frequency of PWM can be 38 kHz and duty is 50 %. There is an Application Note **Generating PWM and PFM by using FlexIO**. User can take this AN for a reference to know how to generate 38 kHz and 50 % duty PWM.

3.2 How to configure an NRZ encoder by FLEXIO

There is a demo example for FLEXIO in RT1010 SDK: *flexio_uart_interrupt_transfer*. The encoder can develop based on it. The TX data should be NRZ format, so that UART data falling edge triggers encode timer. Once there is a falling edge, the NRZ data pin sends a fix duty pulse, so that the timer mode should configure as PWM mode. The following lists show the detail about the encoder timer configuration:

- *triggerSelect:* Triggered by FLEXIO UART TX pin
- *triggerSource:* Internal trigger
- *triggerPolarity:* Trigger Polarity is Active Low
- *pinConfig:* Pin output
- *pinPolarity:* Pin Polarity is Active High
- *pinSelect:* NRZ data output pin
- *timerMode:* Dual 8-Bit PWM
- *timerOutput:* Never Reset
- *timerDecrement:* Decrement counter on FlexIO clock, Shift clock equals Timer output
- *timerDisable:* Timer disabled on Timer compares
- *timerEnable:* Timer enabled on Trigger high
- *timerReset:* Timer never resets
- *timerStart:* Start bit disable
- *timerStop :* Stop bit is enabled on timer disable

Besides these settings, the frequency of PWM and the duty of the pulse must be configured. The duty of pulse can be 3/16 period of PWM. The frequency of PWM is same with the baud rate.

The Compare Register must be configured to determine the frequency and duty of the PWM. The timer is working under Dual 8-bits mode, the lower 8-bits determines the High-level timing and the upper 8-bits determines the low-level timing. For specific calculation and configuration, FLEXIO PWM demo in the SDK can take a reference: *boards\evkmimxrt1010\driver_examples\flexio\pwm*.

Figure 4 shows a timing sequence for UART TX data(Character U) and NRZ data.

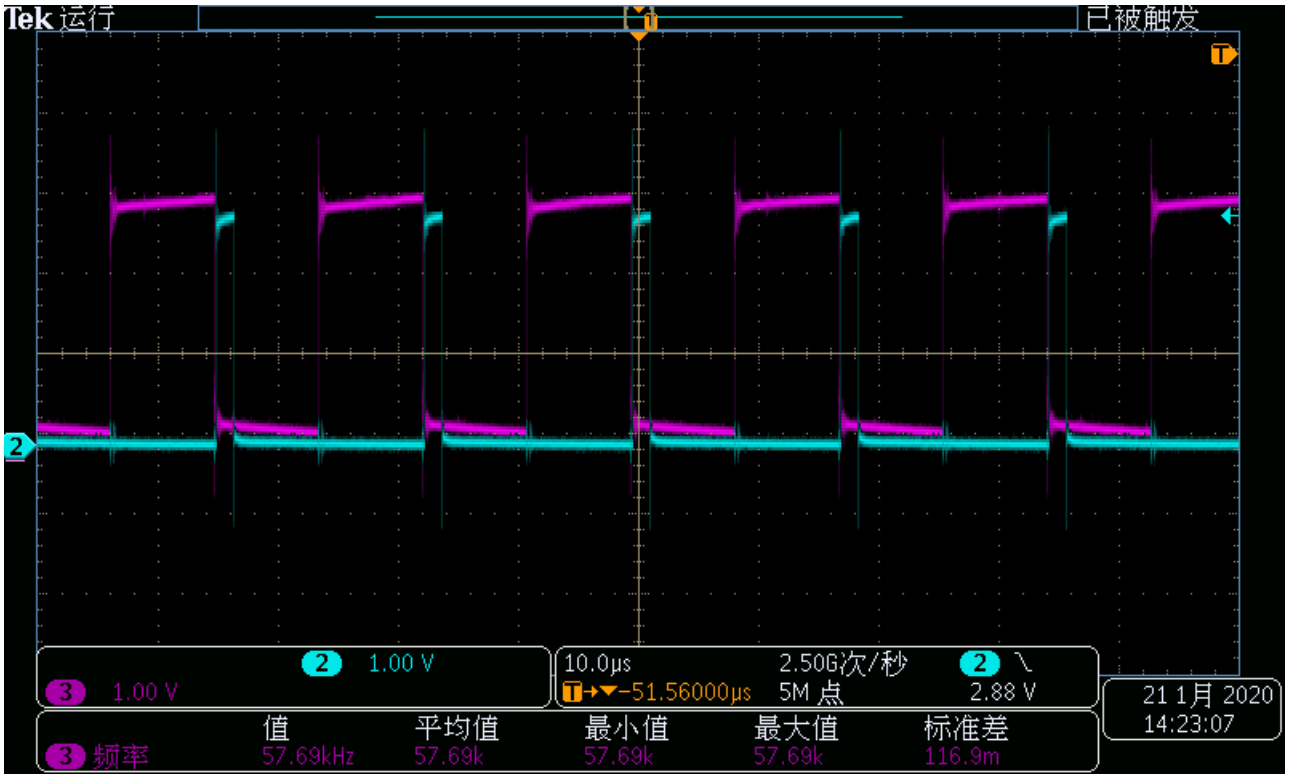

Figure 4. CH3 is the source data with UART and CH2 is the NRZ format after encoding

3.3 How to configure an NRZ decoder by FLEXIO

After the RX circuit processes the IrDA, signal the FLEXIO input pin can provide the NRZ format data.

A timer is used to convert NRZ format data to UART TTL level data. Figure 1 shows that the timer is triggered by rising edge of the NRZ data. The following settings sre used:

- *triggerSelect:* Triggered by FLEXIO IrDA RX pin
- *triggerSource:* Internal trigger
- *triggerPolarity:* Trigger Polarity is Active High
- *pinConfig:* Pin output
- *pinPolarity:* Pin Polarity is Active Low
- *pinSelect:* FLEXIO UART RX pin
- *timerMode:* Dual 8 Bit Baud Mode
- *timerOutput:* Never Reset
- *timerDecrement:* Decrement counter on FlexIO clock, Shift clock equals Timer output
- *timerDisable:* Timer disabled on Timer compares
- *timerEnable:* Timer enabled on Trigger high
- *timerReset:* Timer reset on Trigger rising edge
- *timerStart:* Start bit disable
- *timerStop:* Stop bit disable

The value of lower 8 bits of CMP is calculated using:

CMP[7:0] = FLEXIO_CLOCK_FREQUENCY / BOARD_DEBUG_UART_BAUDRATE - 1

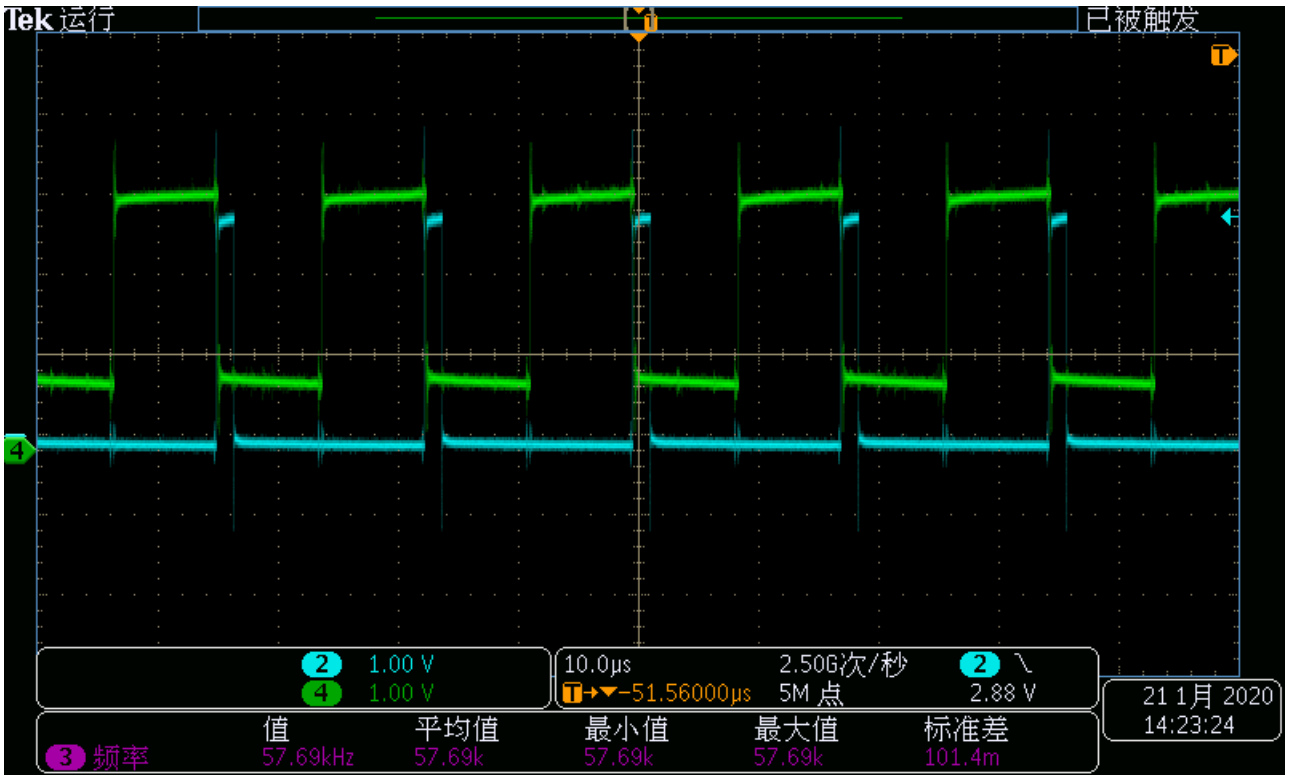

Figure 5. CH2 is the NRZ format data and CH4 is the data after decoding

3.4 Summary for NRZ data encoding and decoding

Figure 6 shows the process for NRZ data encoding and decoding. Besides this, Figure 7 and Figure 8 shows the details trigger sequence for encoding and decoding. The meaning of each channel in Figure 7 and Figure 8, can be found in Figure 6.

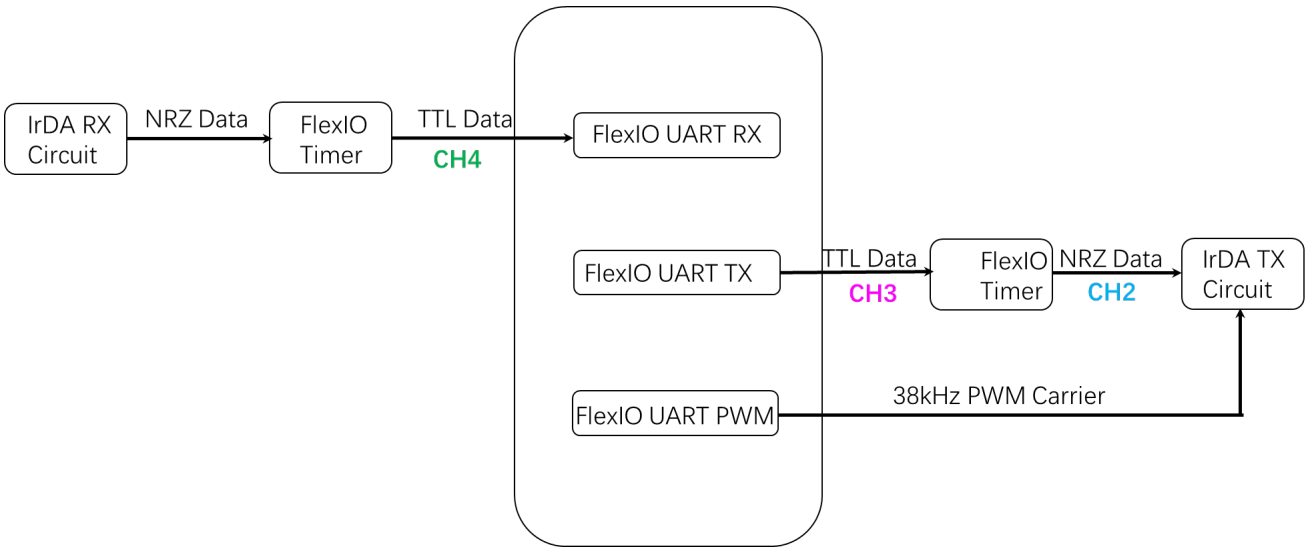

Figure 6. Entire process of NRZ data encoding and decoding

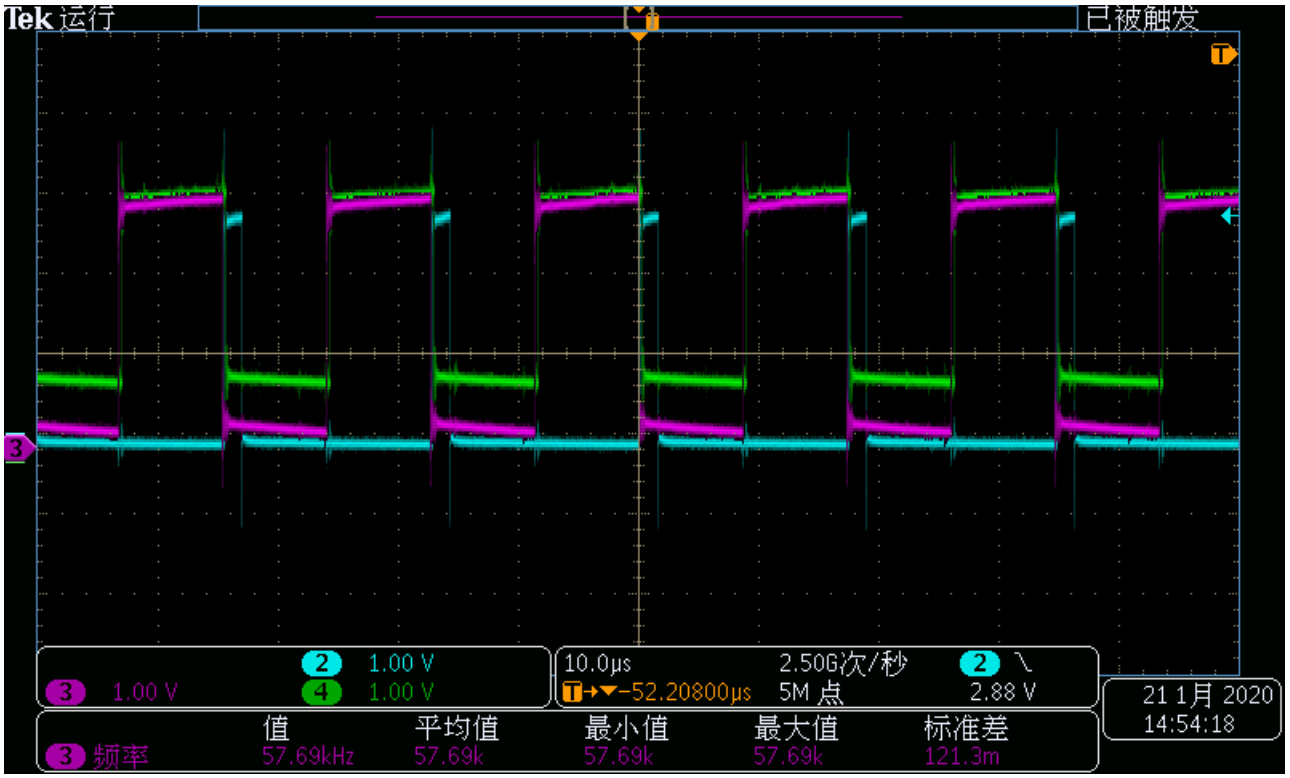

Figure 7. Entire process of NRZ data encoding and decoding trigger sequence

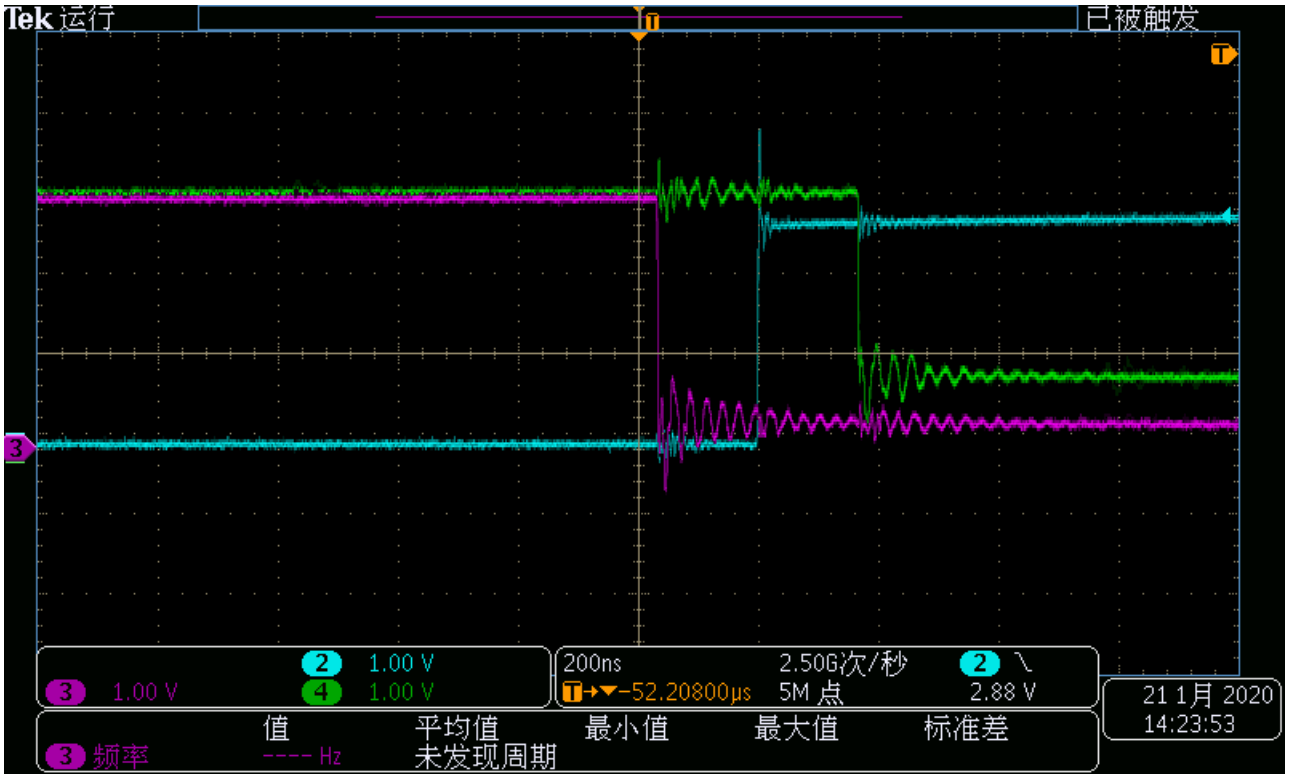

Figure 8. Entire process of NRZ data encoding and decoding trigger sequence

4 How to develop it based on RT1010 EVK

Hardware settings:

- CH2: TX NRZ data, fly a wire from the second pad of R1820 and connect to J26-8
- CH3: TX TTL data
- CH4: RX TTL data

Use the oscilloscope to connect the test points as in Figure 9 to complete the preparation for the hardware test.

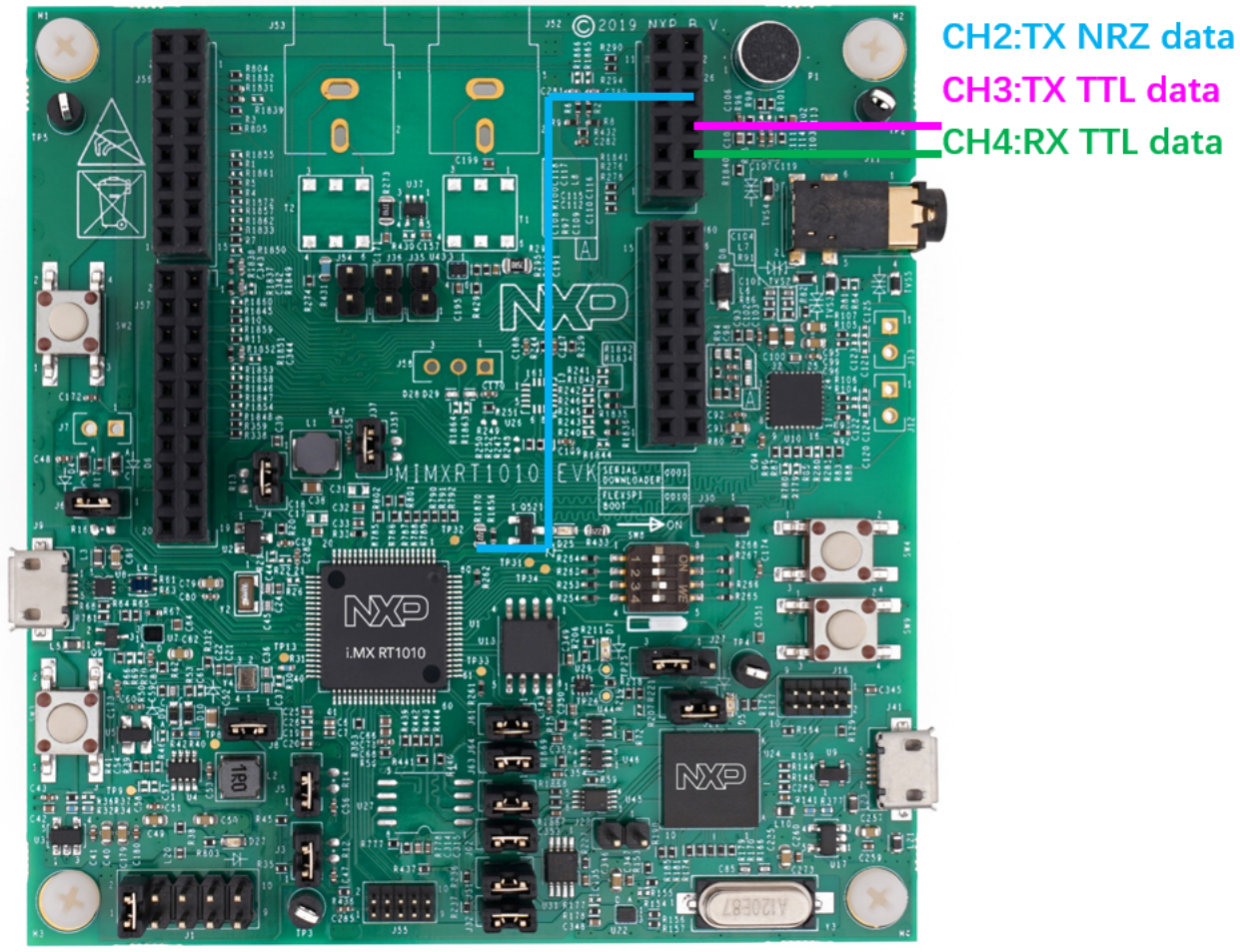

Figure 9. Entire process of NRZ data encoding and decoding trigger sequence

Software settings:

The software is based on *flexion_uart_polling_transfer* in SDK 2.6.1 with IAR Tool Chain.

Replace the pin_mux.c and flexio_uart_polling_transfer.c with the attached files. Download it to the flash, change the value of **IrDA** by IAR watch feature, then run the code, the value of **ch** changes and is same for IrDA. There is another way to check the value, connect J26-4 to the J31-1. Open a terminal with baud rate 115200, the value of ch prints in the terminal.

```
173 {
174     FLEXIO_UART_ReadBlocking(&uartDev, &ch, 1);
175     FLEXIO_UART_WriteBlocking(&uartDev, &IRDA, 1);
176
177 }
```

Figure 10. How to change TX data

5 Revision history

This table summarizes revisions to this document.

Table 1. Revision history

Revision number	Date	Substantive changes
0	01/2020	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: 31 January 2020

Document identifier: AN12721

arm