AN12788

使用 I.MXRT 系列 MCU 的 FlexIO 模块来模拟 LIN 总线主机及从机

Rev. 0 — March 31 2020

Application Note

目录

1 概述

本应用笔记介绍了如何使用 I.MXRT 系列 MCU 中的 FlexIO 模块来实现 LIN 总线。

LIN 是一种广泛应用于车内电子设备组网的低成本串行总线协议。虽然 RT 系列 MCU 中并未直接支持 LIN 外设，但是 RT 所包含的 FlexIO 模块可以很好的模拟出 LIN 总线的通信。

FlexIO 是 I.MXRT 系列 MCU 上板载的一种外设。它是一个非常灵活并且可以随意配置的模块，它不仅可以模拟出类似于 UART，I^2C，SPI，I^2S 等常用通信接口外，用户还可以使用它来模拟 LIN 总线。

本应用笔记中实现了一个基于 I.MXRT 系列平台的简单的软件程序，以供用户熟悉 FlexIO 模块模拟 LIN 主机及从机的应用。

2 LIN 概述

LIN 是一种基于 UART/SCI 的低成本串行通信协议，采用一主多从的通信方式，传输速率最高可达 20 kpbs。一个 LIN 的通信网络里最多可以连接 16 个节点，1 个主机节点以及 1 到 15 个从机节点。LIN 总线的拓扑为线型，即所有的节点设备均通过单线连接。图 1 即为 LIN 总线的拓扑结构。

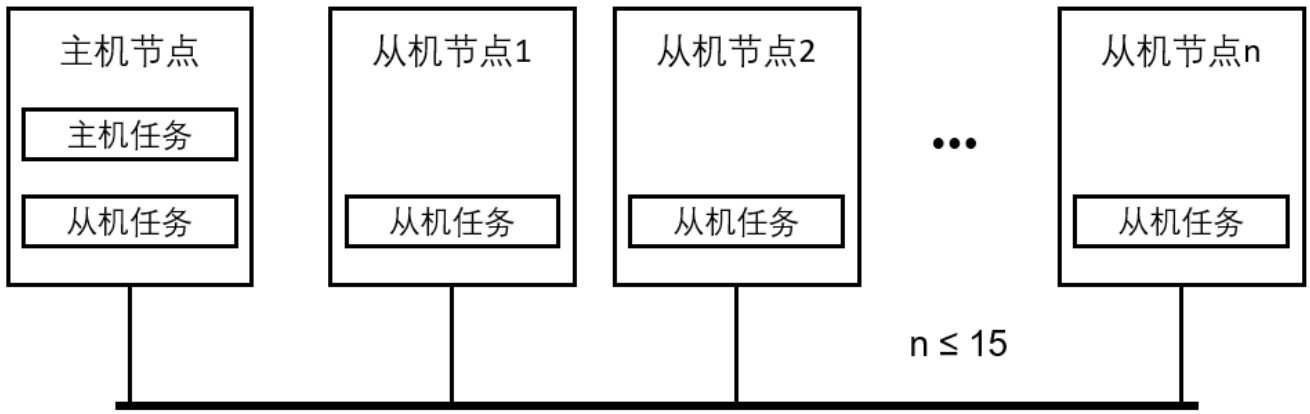

图 1. LIN 总线拓扑结构

主机节点中包含主机任务及从机任务，从机节点中只包含从机任务。主机任务主要负责发送帧头（Header）以发起 LIN 总线的通信。从机任务负责对帧头做出应答并根据帧头中的受保护 ID 段（PID）决定数据接收或发送。

LIN 的数据帧结构包含帧头（Header）和应答（Response）两部分。帧头部分由主机任务发出，从机任务根据帧头的信息决定发送应答还是接收应答。总线上的数据传输过程如 图 2 所示。

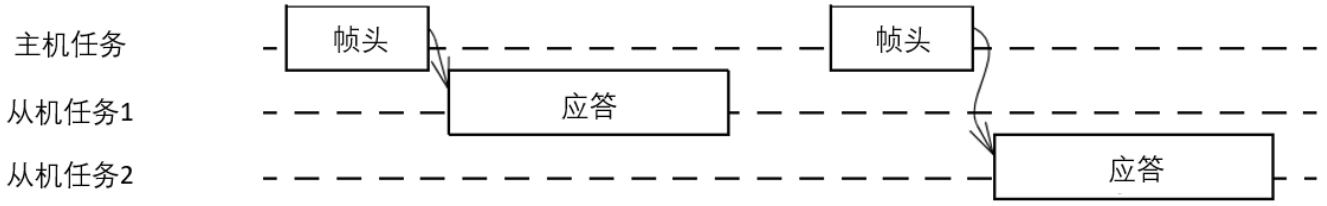

图 2. LIN 数据的传输流程

主机任务发出的帧头包含了三个场域，分别是同步间隔场（Break Field）、同步场（Sync Byte Field）以及标识符场即 PID（Protected Identifier）；应答包括数据段和校验和段。图 3 展示了一个完整的数据帧的结构。

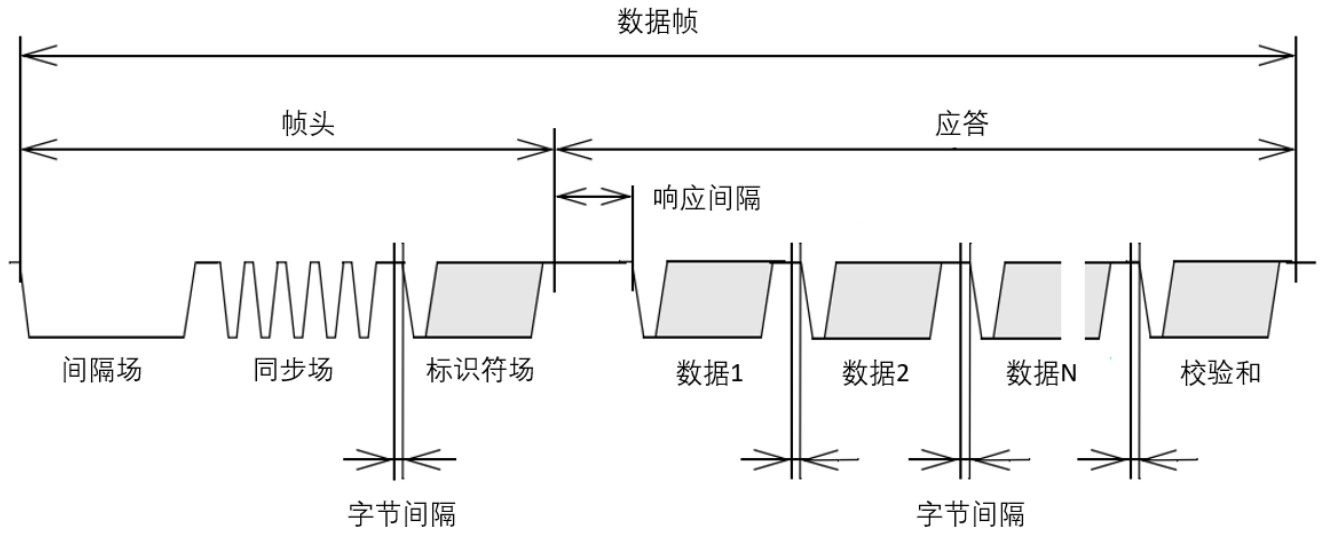

图 3. LIN 帧的结构

帧中的间隔场为至少连续 13 位的显性电平，接下来从同步场到校验和场的其余数据的格式都与标准异步串口的数据格式相同，为每一字节数据都有 1 位低电平起始位，1 位高电平停止位，无校验位。LIN 帧的数据发送中都是先发送最低有效位（LSB），最后发送最高有效位（MSB）。同步场为固定值 0x55。

3 使用 FlexIO 模拟 LIN

本章主要讲述使用 FlexIO 模块模拟 LIN 总线的主机及从机的详细配置，LIN 总线是一种低成本的串行总线，由于数据格式大部分与异步串口相同，所以在 FlexIO 的配置上，LIN 与 UART 非常相似。

3.1 LIN 发送端配置

LIN 总线的发送端需要用到如下的一些资源：

- 一个定时器 – 配置为 8-bit baud counter 模式来控制移位器的移位
- 一个移位器 – 根据定时器的配置从 SHIFTBUF 加载数据并移出
- 一个引脚 – 与上面使用到的移位器相连以输出数据

本应用笔记中使用 Timer 0 来控制 Shifter 0 模拟 LIN 总线的发射端。图 4 展示了 LIN 发射端的结构框图。

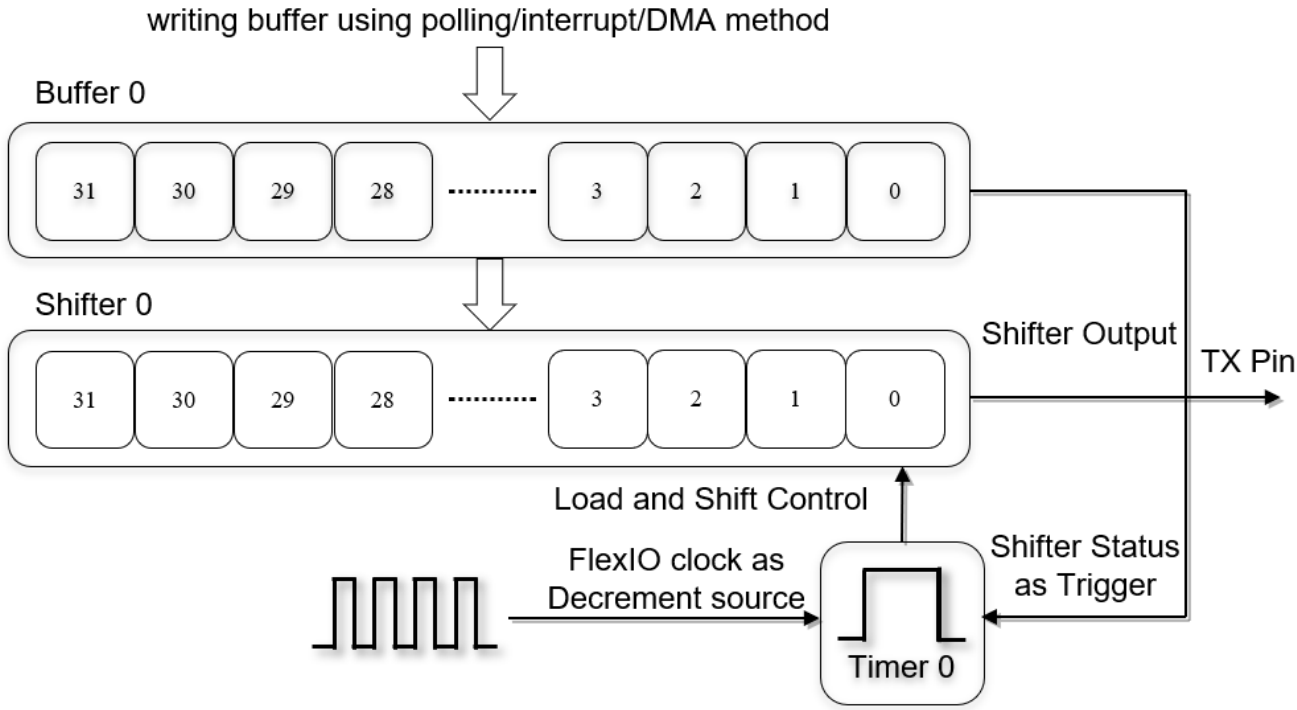

图 4. LIN 发射端结构框图

LIN 总线协议规定了传输的波特率限制为 20 Kbps，在本应用笔记中，设置波特率为 19200 以便于时钟分频的配置。 表 1 显示了 Timer 0 的各项配置。

表 1. Timer 0 配置

Items	Configurations
Trigger Select	Shifter 0 status flag
Trigger Polarity	active low
Trigger Source	internal trigger
Pin Config	output disable
Pin Select	N/A
Pin Polarity	N/A
Timer mode	dual 8-bit counters baud mode
Timer Output	Timer output is logic one when enabled and is not affected by timer reset
Timer Decrement	decrement counter on FlexIO clock, shift clock equals Timer output
Timer Reset	Timer never reset
Timer Disable	Timer disabled on Timer compare

Table continues on the next page...

表 1. Timer 0 配置 (continued)

Items	Configurations
Timer Enable	Timer enabled on Trigger high
Timer Stop Bit	enabled on timer disable
Timer Start Bit	enable
Timer Compare	((bitCountPerChar[1] * 2 - 1) << 8) | (baudrate_divider[2]/ 2 - 1))

1. bitCountPerChar 为传输字长。
2. baudrate_divider 是波特率分频系数，可由 FlexIO 时钟频率除以 LIN 总线波特率得到。

Timer 0 被配置为 8-bit baud counter mode，这种模式下，16 位长度的计数器以及比较寄存器（计数器记到 0 时会重新加载比较寄存器内的值）被分割成两个 8 位长度的计数器。其中，低 8 位的计数器用来配置波特率（移位器时钟），高 8 位的计数器用来配置传输中的字长。当低 8 位的计数器自减到 0 时，Timer 的输出会翻转，然后低 8 位的计数器会重载比较寄存器中的值，此时高 8 位的计数器会自减 1。表 2 展示了 Shifter 0 的配置。

表 2. Shifter 0 配置

Items	Configurations
Timer Select	Timer 0
Timer Polarity	shift on posedge of shift clock
Pin Config	Shifter pin output
Pin Select	FlexIO_D21
Pin Polarity	active high
Shifter Mode	transmit mode
Input Source	N/A
Shifter Stop Bit	stop bit high
Shifter Start Bit	start bit low

当 SHIFTBUF 寄存器内的数据被转移到移位器以后，shifter status flag 会置位，而当朝 SHIFTBUF 寄存器内写值时，shifter status flag 会被清 0。在本应用笔记中，Shifter 0 的 shifter status flag 被配置为 Timer 0 的触发器，当 SHIFTBUF 寄存器被写入数据，shifter status flag 被清零并且 Timer 0 被使能。Shifter 0 在 Timer 0 的输出的上升沿移出数据直到 Timer 0 的计数器自减为 0 失能，一个字（此处为 8 位）的数据被传输完成。

3.2 LIN 接收端配置

LIN 总线的接收端需要用到如下的一些资源：

- 一个定时器 – 配置为 8-bit baud counter 模式来控制移位器的移位
- 一个移位器 – 根据定时器的配置从引脚接收数据并存入 SHIFTBUF
- 一个引脚 – 与移位器相连以输入数据

本应用笔记中使用 Timer 1 来控制 Shifter 1 模拟 LIN 总线的接收端。图 5 展示了 LIN 接收端的结构框图。

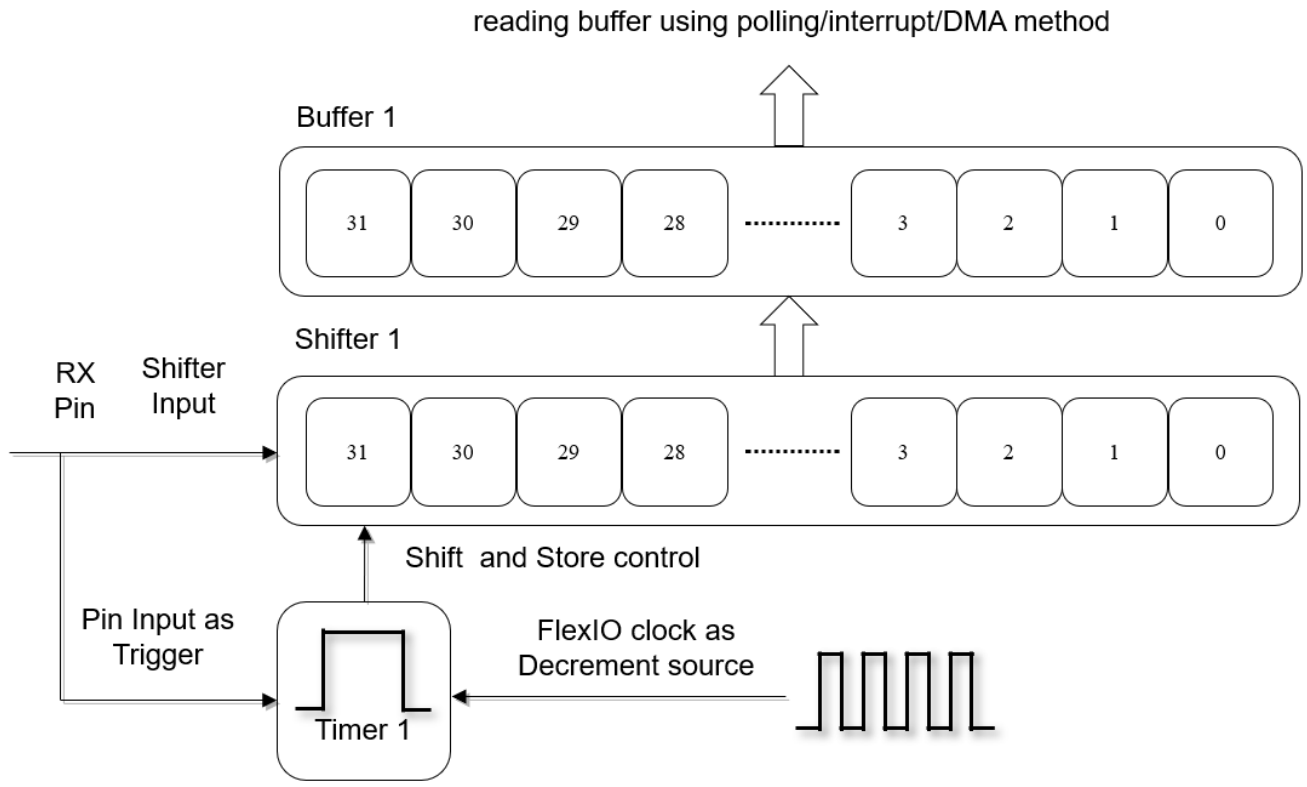

图 5. LIN 接收端框图

表 3 展示了 Timer 1 的配置。

表 3. Timer 1 配置

Items	Configurations
Trigger Select	trigger from pin FlexIO_D26
Trigger Polarity	active high
Trigger Source	external trigger
Pin Config	output disable
Pin Select	N/A
Pin Polarity	N/A
Timer mode	dual 8-bit counters baud mode
Timer Output	Timer output is logic one when enabled and on timer reset
Timer Decrement	decrement counter on FlexIO clock, shift clock equals Timer output
Timer Reset	Timer reset on Timer Pin rising edge

Table continues on the next page...

表 3. Timer 1 配置 (continued)

Items	Configurations
Timer Disable	Timer disabled on Timer compare
Timer Enable	Timer enabled on Pin rising edge
Timer Stop Bit	enabled on timer disable
Timer Start Bit	enable
Timer Compare	((bitCountPerChar * 2 - 1) << 8) | (baudrate_divider / 2 - 1))

表 4 展示了 Shifter 1 的配置。

表 4. Shifter 1 配置

Items	Configurations
Timer Select	Timer 1
Timer Polarity	shift on negedge of shift clock
Pin Config	output disable
Pin Select	FlexIO_D26
Pin Polarity	active high
Shifter Mode	receive mode
Input Source	input from pin
Shifter Stop Bit	stop bit high
Shifter Start Bit	start bit low

FlexIO_D26 引脚的上升沿配置为 Timer 1 的使能信号，Shifter 1 在 Timer 1 输出信号的下降沿开始接收数据，直到 Timer 1 失能时停止。

3.3 LIN 任务模型

以上两节的内容所使用的 FlexIO 的配置与 FlexIO 模拟串口的配置基本相同，因为 LIN 总线的数据传输格式与串口基本相同，但是这样的配置并不能满足 LIN 协议中主机发起通信时发送的帧头中持续 13 位显性电平的要求。本节主要介绍 LIN 协议相关的具体任务模型的实现。

3.3.1 帧头配置

帧头的间隔场表示一帧报文的起始，由主机节点发出，间隔至少 13 个显性电平（低电平），间隔场是唯一一个不符合串口数据格式的场，帧中的所有其他任何字段都不会发出大于 9 位的显性电平，所以常规的 FlexIO 配置无法满足这个要求。由于间隔场与同步场每次发送的数据都是相同的，所以定义一组常量数组来模拟这两个场域的发送。图 6 就是这两个场域的结构图。

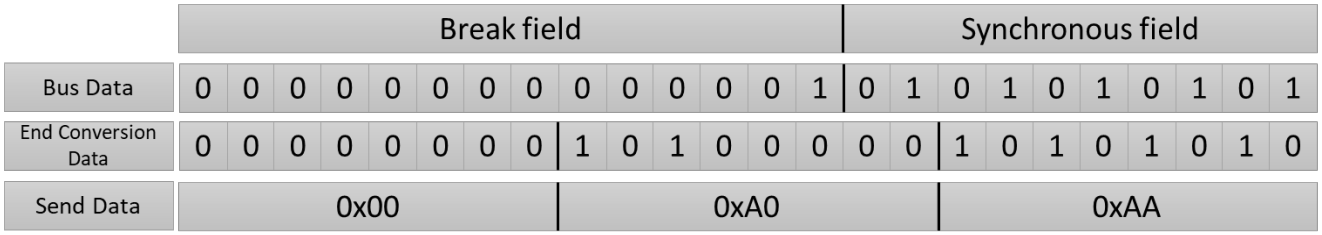

图 6. 间隔场及同步场数据结构

由于每帧标准的 10 为串行数据会包含 1 位的高点平停止位，所以在发送上述的常量数组时，需要改变定时器及移位器的相关配置失能起始位及停止位。

当主机需要发送帧头时，用户可修改 Timer 0 及 Shifter 0 的配置为：

- Timer Stop Bit: Disable
- Timer Start Bit: Disable
- Shifter Stop Bit: Disable
- Shifter Stop Bit: Disable

并且设置发送长度为 3 字节, 发送数组 {0x00, 0xA0, 0xAA}。

对于从机任务来说，至少要检测到连续 11 位的显性电平（低电平）才认为收到了间隔场，去除 1 位的起始位，接收端的 Timer 1 需要修改字长的配置以接收保存这一段场域：

- Timer Compare[15:8]: 0x13

3.3.2 主机节点状态机

LIN 总线数据收发时存在多个运行状态，需要进行不同的处理。将其分为应用层及驱动层两个层次的状态机模型。图 7 即为主机节点的状态机模型，蓝色框代表应用层状态，绿色框代表驱动层状态。

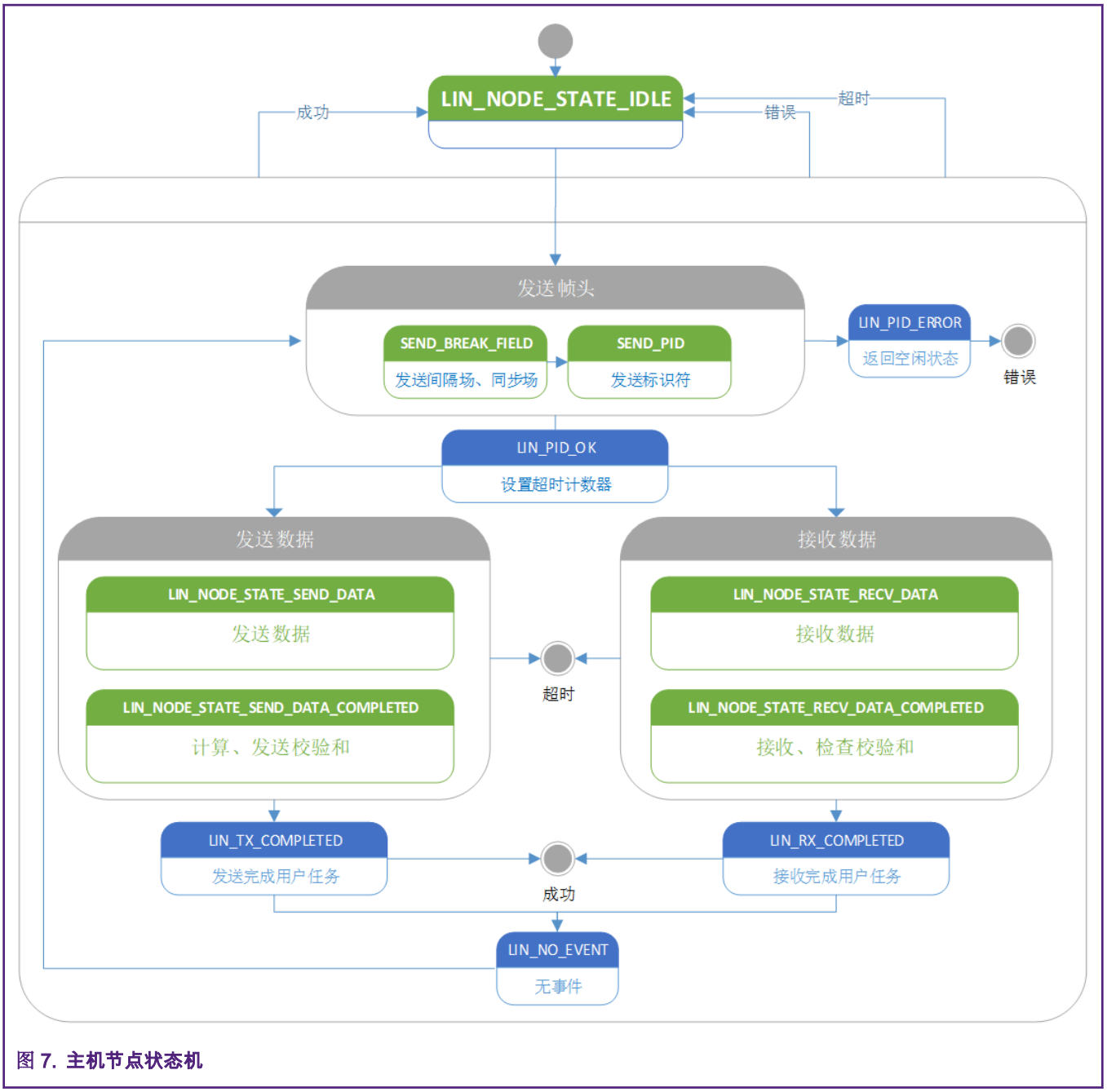

图 7. 主机节点状态机

3.3.3 从机节点状态机

与主机节点相同，从机节点的状态机同样分为应用层及驱动层两层，图 8 为从机节点的状态机转换图。

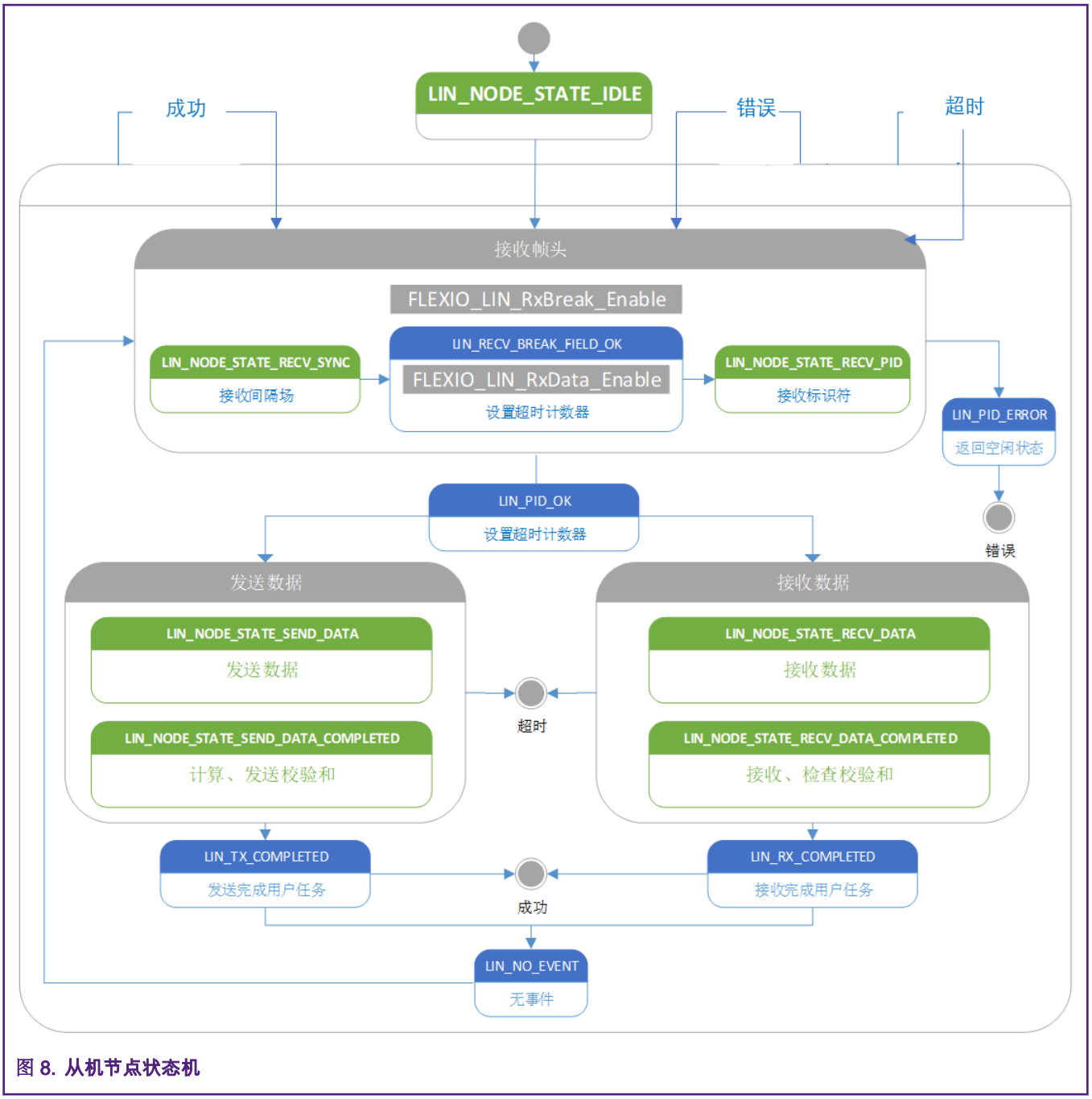

图 8. 从机节点状态机

在本例程中，从机节点一直在监测 LIN 总线上是否有间隔场发出，所以设置了两个使能函数来转换 FlexIO 的配置。使用 *FlEXIO_LIN_RxBreak_Enable* 来使能接收间隔场，使用 *FlEXIO_LIN_RxData_Enable* 来使能接收其他场域的数据。

4 示例的运行

4.1 示例平台

本应用笔记介绍了基于 I.MXRT1010-EVK 开发板的 FlexIO 模拟 LIN 总线的例程。I.MXRT1010-EVK 板如 图 9 所示。用户也可使用 I.MXRT 系列 MCU 的其他 EVK 板经过少许更改后实现这个例程。

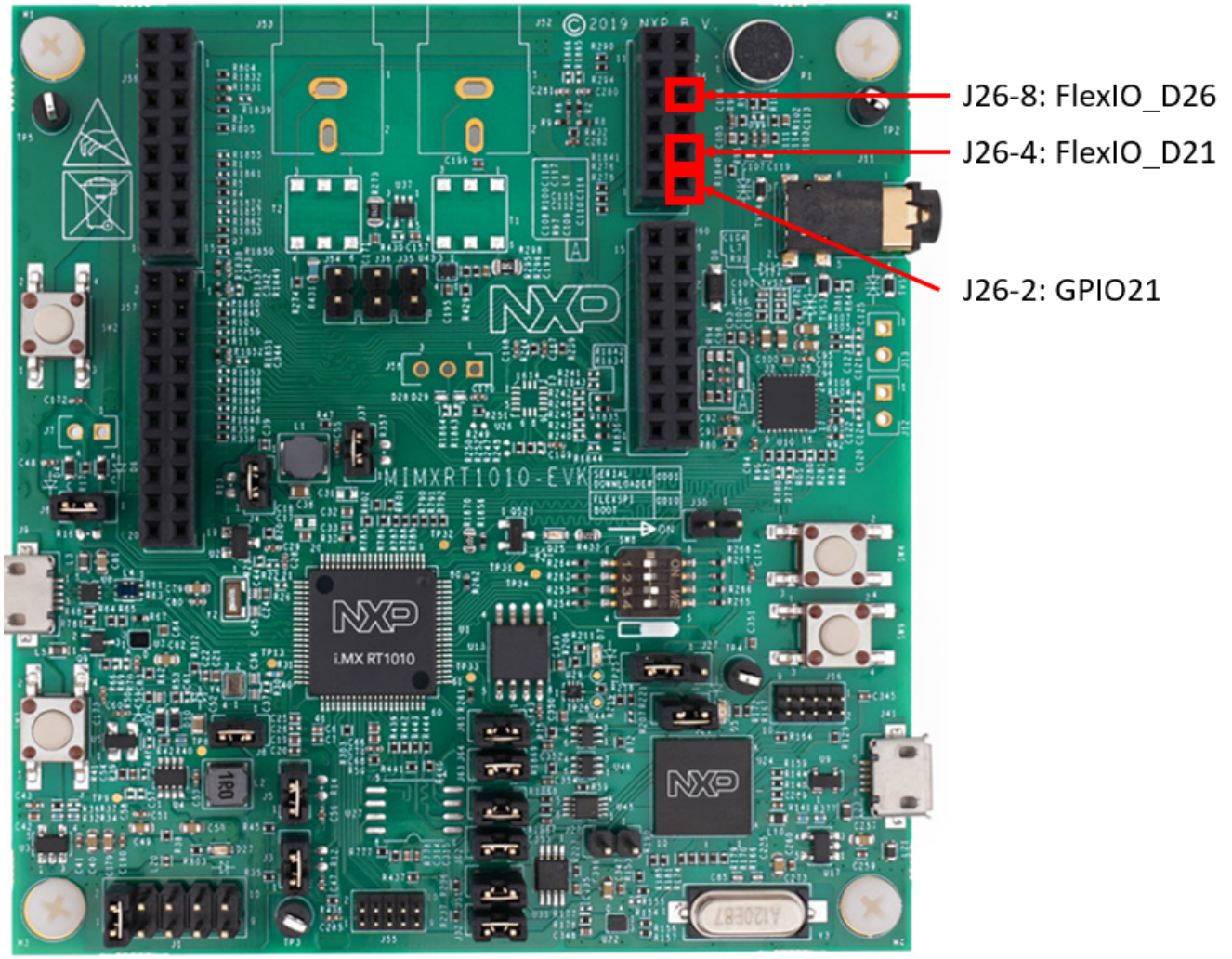

图 9. I.MXRT1010-EVK 板

在本例程中，使用 FlexIO_D21 引脚作为发送引脚，FlexIO_D26 作为接收引脚。使用两块板子分别作为 LIN 主机节点与从机节点，为了实现 LIN 协议的单线传输，还需要再外接两个 LIN 收发模块，本例程中使用的是 TJA1020 作为 LIN 的收发芯片。使用 GPIO21 脚作为 LIN 收发器的使能引脚。

具体的硬件引脚连接如下：

主机节点		主机收发器			丛集收发器			从机节点
J26-4(LIN_TX)	<->	TX	LIN	<->	LIN	RX	<->	J26-8(LIN_RX)
J26-8(LIN_RX)	<->	RX	INH		INH	TX	<->	J26-4(LIN_TX)
J26-2(Trans_EN)	<->	SLP	12V		12V	SLP	<->	J26-2(Trans_EN)
J60-14(GND)	<->	GND	GND	<->	GND	GND	<->	J60-14(GND)

图 10. Hardware pin connection

4.2 运行示例

用户可以从 NXP 官网下载相关示例程序，工程名为 *flexio_LIN*，这个工程中包含了主机与从机的代码，使用宏定义分隔。在 *flexio_LIN_driver.h* 文件中，将宏 FLEXIO_LIN_MODE 设置为 FLEXIO_LIN_MASTER_NODE 或 FLEXIO_LIN_SLAVE_MODE 可以在主机与从机模式中切换，使用两块 EVK 板，分别下载主机与从机模式的代码并按上述的连接方式连接两板，即可成功运行示例。图 11 所示是由示波器抓取的帧头（Header）部分的信号。

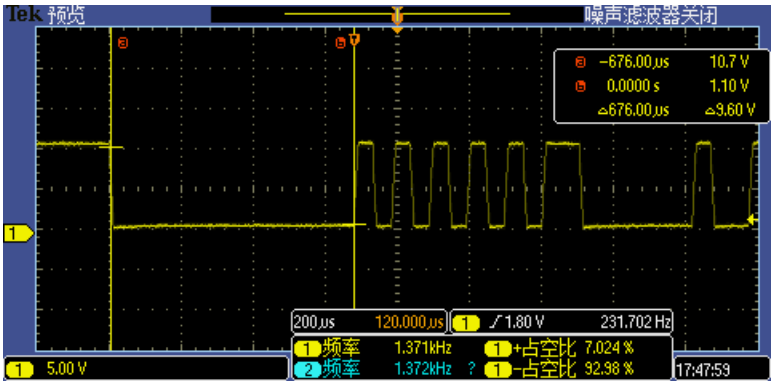

图 11. LIN 帧头部分波形

图 12 所示为调试串口所打印出的控制信息。本例程中实现了一个小的控制回路，在主机端的调试串口中按提示输入‘L’，‘M’或‘H’这三个字符可以控制从机端板子上的 LED 的亮度，此 LED 的亮度也是由 FlexIO 模拟的 PWM 波来实现调节的。

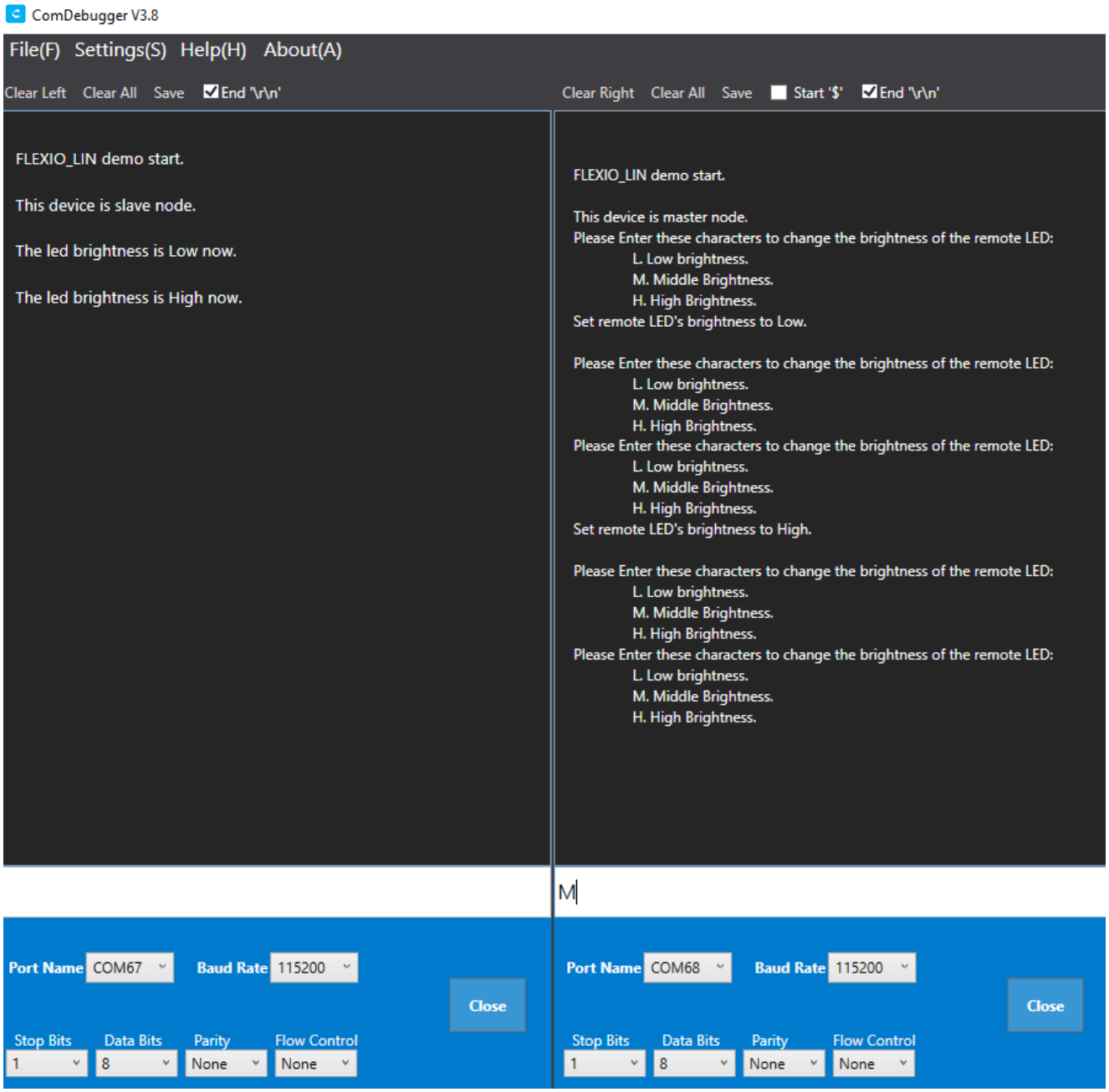

图 12. 调试串口打印的控制信息

5 参考资料

1. *I.MX RT1010 Processor Reference Manual* (document I.MXRT1010RM)
2. MCUXpresso SDK: Software Development Kit for NXP MCUs
   https://mcuxpresso.nxp.com/en/welcome

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: March 31 2020

Document identifier: AN12788

arm