AN14458

如何在MCX A系列上达到24Mbit/s的LPUART最大波特率

第1.0版—2024年10月17日 应用笔记

文档信息

信息	内容
关键词	AN14458、MCX A系列、LPUART、24 Mbit/s、Trim Clock
摘要	本应用笔记介绍了如何在MCX A系列上达到24 Mbit/s的LPUART最大波特率，并在FRDM-MCXA153开发板上进行测试。

1 介绍

基于Arm Cortex-M33核的MCX A系列微控制器是一款通用MCU，旨在通过可扩展的设备选项、低功耗和智能外设，来满足广泛的应用需求。它们提供低功耗通用异步收发器（LPUART）。该系列MCU提供了具有主设备（commander）和从设备（responder）操作的异步串行总线。MCXA152/3支持3个LPUART模块，MCXA154/5/6支持5个LPUART模块。通常，微控制器通过UART与主设备进行通讯并打印调试日志。在这种用例中，最常见的波特率为9600bit/s和115200bit/s，这对微控制器来说是一项简单的任务。然而，在某些用例中，需要高波特率的LPUART，对于MCX A系列，支持最高24Mbit/s的波特率。本应用笔记介绍了MCX A系列中的LPUART模块，并详细介绍了如何达到24Mbit/s的LPUART最大波特率：

- 如何设置LPUART以获得24Mbit/s的波特率
- 如何实现LPUART的发送、接收和检查
- 如何计算传输误码率
- 如何输出FRO_HF时钟
- 如何手动调整FRO_HF时钟
- 如何使用外部晶振自动调整FRO_HF时钟

2 LPUART概述

LPUART提供具有主设备和从设备操作的异步串行总线。最大传输速率可达24Mbit/s。该数据是基于性能的，但并不涵盖在生产的测试极限中。

2.1 特性

- 全双工，标准NRZ格式
- 可编程波特率（13位模数分频器），具备可配置的（OSR）4至32倍的过采样率，最高可达24Mbit/s。
- 有关总线时钟的发送和接收波特率的异步操作：
  - 波特率可以独立于总线时钟频率进行配置。
  - 支持在低功耗模式下运行。
- 支持中断、DMA或轮询操作：
  - 发送数据为空，传输完成
  - 接收数据已满
  - 接收溢出错误、奇偶校验错误、帧错误和噪声错误
  - 空闲接收器检测
  - 接收脚上的有效沿
  - 支持LIN的中断检测
  - 接收数据匹配
- 硬件奇偶校验位的生成和检查
- 可编程的7位、8位、9位或10位数据位
- 可编程的1位或2位停止位
- 支持三种接收器唤醒方法：
  - 空闲线路唤醒
  - 地址标记唤醒

  - 接收数据匹配
- 自动地址匹配，以减少ISR开销
  - 地址标记匹配
  - 空闲线路地址匹配
  - 地址匹配开始，地址匹配结束
- 可选13位和11位中断字符生成
- 可配置的空闲字符长度检测，支持1、2、4、8、16、32、64或128个空闲字符
- 可选择的发送器输出和接收器输入极性
- 硬件流控制支持“请求发送”（RTS）和“清除发送”（CTS）信号
- 可选IrDA 1.4反相归零编码（RZI）格式，具有可编程的脉宽
- 用于发送和接收功能的独立FIFO结构：
  - 用于接收和发送请求的单独可配置的水印标记
  - 如果接收FIFO不为空，接收器可以选择在经过一段可配置数量的空闲字符后发出请求。

2.2 功能

图1所示为LPUART的发送器部分，图2所示为LPUART的接收器部分。

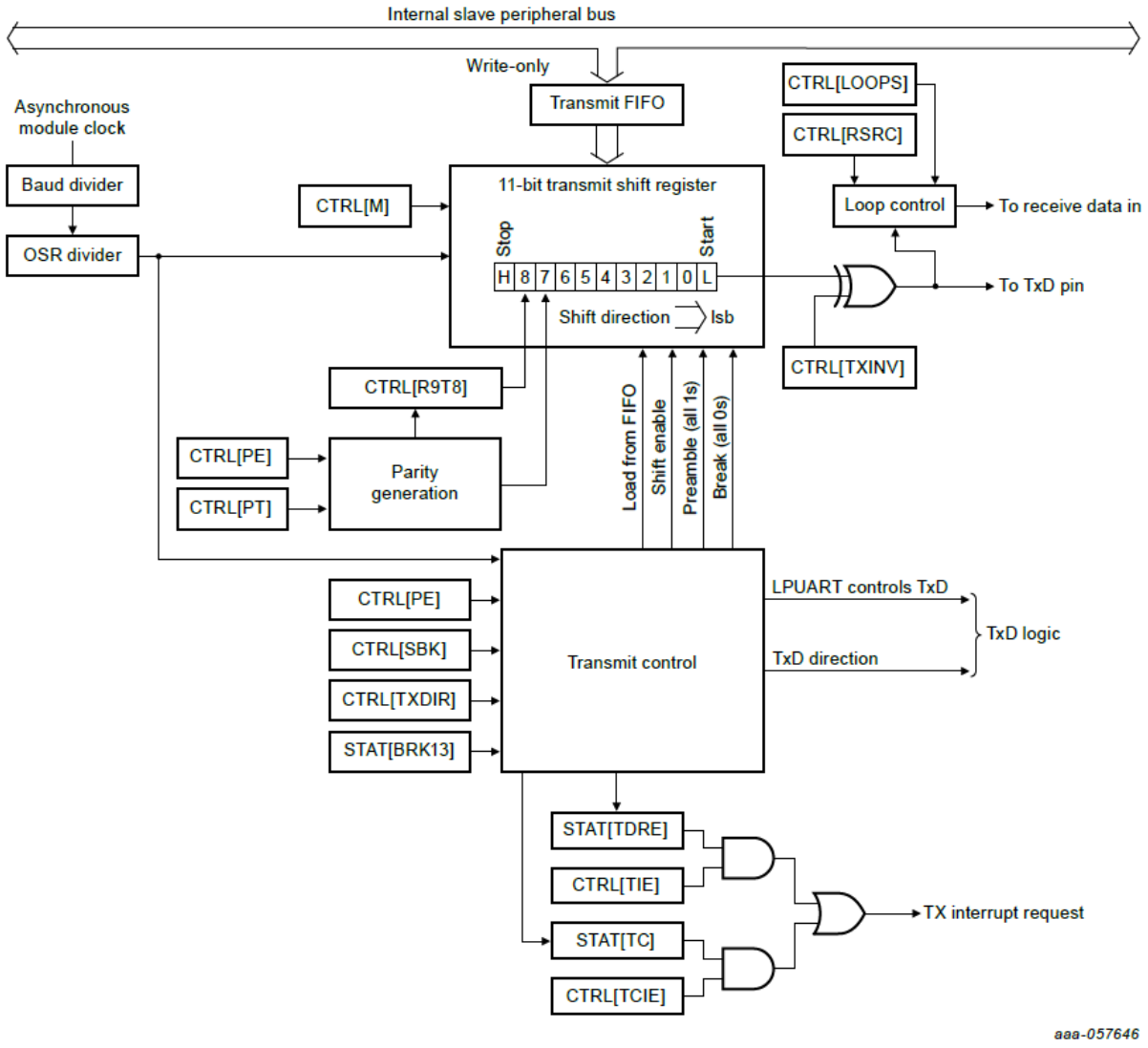

图1. 发送器结构框图

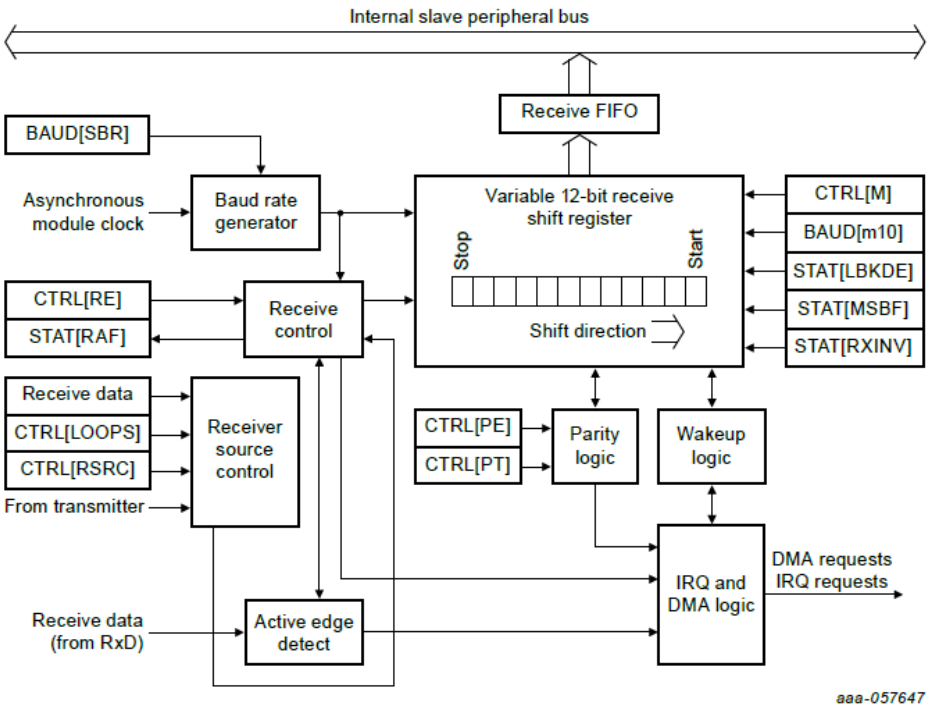

图2. 接收器结构框图

2.2.1 波特率的生成

波特率发生器中的13位模数计数器可得出接收器和发送器的波特率。写入BAUD[SBR]的范围在1到8191之间的数值决定了异步LPUART波特率时钟的波特时钟除数。波特率时钟驱动接收器，而由波特率时钟除以OSR进行分频生成的位时钟则驱动发送器。根据OSR，接收器的采集率为每位4到32个样本。

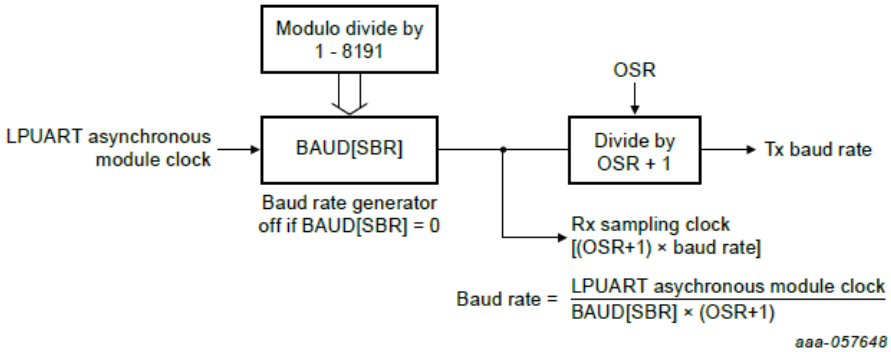

图3. 波特率的生成

为了达到24Mbit/s的最大波特率，将FRO_HF 96MHz设置为LPUART异步模块时钟，将BAUD[SBR]设置为1，OSR + 1设置为4，则波特率为96MHz / 4 = 24Mbit/s。

2.2.2 波特率的容差

发送设备可以以一个低于或高于接收器的波特率运行。

累积的位时间错位可能会导致3个停止位数据样本之一落在实际停止位之外。如果3个样本不都是相同的逻辑值，则会发生噪声错误。如果接收器时钟未对齐，使得3个停止位样本中的大多数为逻辑零，则会发生帧错误。

当接收器对传入帧进行采样时，它可以在帧内的任何有效下降沿上重新同步过采样时钟。帧内的重新同步可以纠正发送器位时间和接收器位时间之间的偏差。

一般来说，增加每位的样本数会增加波特率容差，而减少每位的样本数则会降低波特率容差。

注：*由于LPUART在连续接收数据样本上实施了三重表决，因此增加每位的样本数会使这些样本更接近，从而降低三重表决逻辑可过滤的噪声宽度。*

2.2.3 计算波特率容差

使用以下定义：

- SAM是每位的采样点数（有效范围为8到32；等于（OSR+1）×（BOTHEDGE + 1））。
- BIT是字符中的位数，包括起始位、数据位和停止位（有效范围为9到13）。

理想的波特率容差可按如下方式计算：

- 慢速数据速率容差 =（（SAM÷2）-1）÷（（SAM×BIT）-（SAM÷2）+2）
- 快速数据速率容差 =（（SAM÷2）-2）÷（SAM×BIT）

在此示例中，OSR+1 = 4，BOTHEDGE = 1，SAM = 4*2 = 8，BIT = 10（1个起始位，8个数据位，1个停止位）：

- 慢速数据速率容差 =（4-1）÷（80 - 4 + 2）= 3 ÷ 78 = 3.85 %
- 快速数据速率容差 =（4-2）÷80=2÷80 = 2.5%

3 软件设计

该软件基于FRDM-MCXA153 SDK 2.16.0 `driver_examples/lpuart/lpuart_edma_transfer`，这是一个使用eDMA的LPUART传输示例。用户需熟悉此示例和相关的硬件。

该软件使用一个FRDM-MCXA153作为发送端（Sender），使用另一个FRDM-MCXA153作为接收端（Receiver），发送端使用LPUART@24Mbps波特率将数据发送给接收端，然后接收来自接收端回复的数据并计算误码率。

3.1 时钟生成

要将LPUART波特率设置为24Mbit/s，请使用FRO_HF 96MHz作为LPUART异步模块时钟，并将其分频为24MHz，以达到24Mbit/s的LPUART波特率，有关更多详细信息，请参阅第2.2.1节。

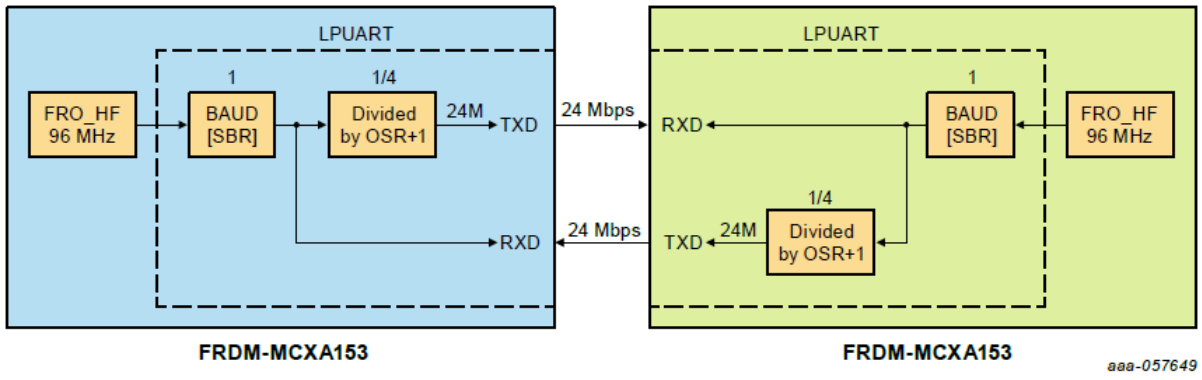

图4. 时钟生成

时钟生成的核心代码如下：

```
CLOCK_SetupFROHFClocking(96000000U);   /* !< Enable FRO HF(96MHz) output */
/*!< Set up dividers */
CLOCK_SetClockDiv(kCLOCK_DivFRO_HF_DIV, 1U); /* !< Set FROHFDIV divider to value 1 */

/* attach 96 MHz clock to FLEXCOMM2, use LPUART2 to test the 24Mbps baud rate */
CLOCK_SetClockDiv(kCLOCK_DivLPUART2, 1u);
CLOCK_AttachClk(kFRO_HF_DIV_to_LPUART2);
RESET_PeripheralReset(kLPUART2_RST_SHIFT_RSTn);
```

3.2 传输流程

为了验证传输的可靠性，采用以下传输流程进行测试，其中一个MCXA153作为发送端（Sender），另一个MCXA153作为接收端（Receiver）。

1. 发送端发送启动信号。接收端接收启动信号并记录。
2. 接收端将启动信号回复给发送端，发送端检查其是否正确，如果正确，则连接成功。
3. 发送端开始发送测试数据，接收端接收这些数据并记录。
4. 接收端将测试数据回复给发送端。发送端检查这些数据并返回到步骤3。

注：

1. *启动信号：0x12、0x21；测试数据：从0x0000到0x07FF，共4096字节。*
2. *发送端应在接收端之后复位。*
3. *此流程还支持如MCXA156等其它MCX A系列，并没有差别。仅以MCXA153为例。*

用户通过下图能够更清晰地了解整个过程：

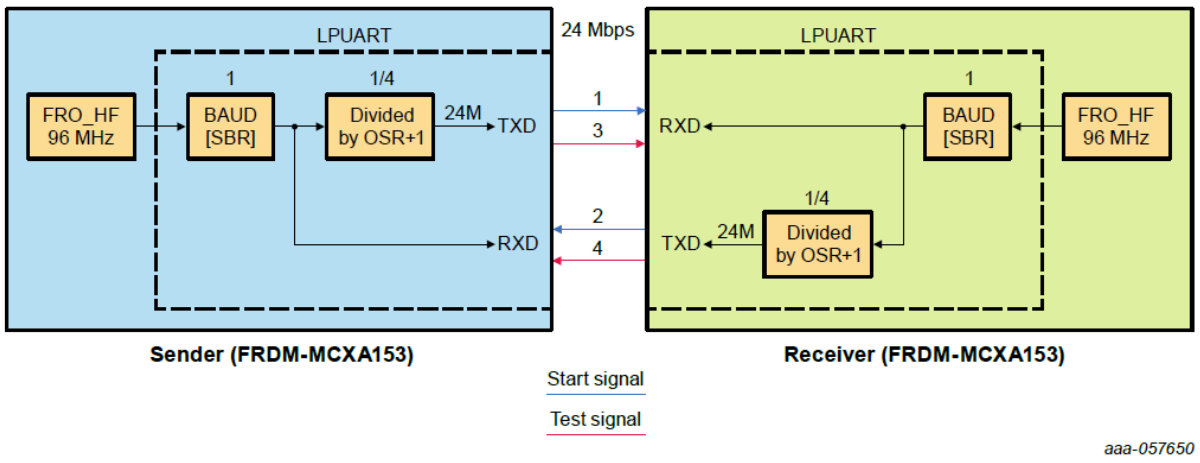

图5. 传输流程

要通过LPUART发送/接收数据，请参考以下代码。发送端发送、接收和检查启动信号的核心代码如下：

```
uint8_t g_startBuffer[2] = {0x12, 0x21};   // Start signal
uint8_t g_rxBuffer[2]    = {0};            // RX buffer

/* LPUART eDMA handle define */
lpuart_edma_handle_t g_lpuartEdmaHandle;

/* LPUART transfer and receive structure define */
lpuart_transfer_t xfer;
lpuart_transfer_t sendXfer;
lpuart_transfer_t receiveXfer;

/* Send communication start signal */
xfer.data     = g_startBuffer;
xfer.dataSize = 2;
txOnGoing     = true;     // See SDK example for details
LPUART_SendEDMA(LPUART2, &g_lpuartEdmaHandle, &xfer);
PRINTF("FRDM-MCXA153 LPUART 24Mbps Demo, this is the sender\r\n");
PRINTF("Sending the communication start signal...\r\n");

/* Wait send finished */
while (txOnGoing)    // See SDK example for details
{
}
PRINTF("Send Done\r\n");

PRINTF("Receiving the communication start signal...\r\n");
/* Receive communication start signal */
receiveXfer.data = g_rxBuffer;
receiveXfer.dataSize = 2;
rxOnGoing = true;
LPUART_ReceiveEDMA(LPUART2, &g_lpuartEdmaHandle, &receiveXfer);
while(rxOnGoing)    // See SDK example for details
{
}
PRINTF("Receive Done\r\n");

PRINTF("Received the connect success signal\r\n");
if((receiveXfer.data[0] == g_startBuffer[0]) && (receiveXfer.data[1] ==
 g_startBuffer[1]))
{
  connectSuccess = true;
  PRINTF("Connect Success\r\n");
}
```

```
else
{
  PRINTF("Connect Fail\r\n");
}
```

发送端发送、接收和检查测试数据的核心代码如下：

```
uint16_t g_txBuffer16[2048] = {0};    // Send test data
uint16_t g_rxBuffer16[2048] = {0};    // Receive test data

/* Set values for the test data */
for(uint16_t i = 0; i < 2048U; i++)
{
  g_txBuffer16[i] = i;
}

PRINTF("Start to do test...\r\n");
sendXfer.txData16    = g_txBuffer16;
sendXfer.dataSize    = 2048U * 2;
receiveXfer.rxData16 = g_rxBuffer16;
receiveXfer.dataSize = 2048U * 2;

while (1)
{
  if(connectSuccess == true && uartCheckDone == false)
  {
     txOnGoing     = true;    // See SDK example for details
     LPUART_SendEDMA(LPUART2, &g_lpuartEdmaHandle, &sendXfer);
     /* Wait send finished */
     while (txOnGoing)     // See SDK example for details
     {
     }
     PRINTF("Send Done\r\n");

     rxBufferEmpty  = true;     // See SDK example for details
     rxOnGoing = true;          // See SDK example for details
     LPUART_ReceiveEDMA(LPUART2, &g_lpuartEdmaHandle, &receiveXfer);
     while(rxOnGoing)           // See SDK example for details
     {
     }
     PRINTF("Receive Done\r\n");

     UART_Check(); // Count Transfer number and Error data number, see section 3.3
  }
}
```

接收端接收、检查和回复启动信号的核心代码如下：

```
uint8_t g_startBuffer[2] = {0x12, 0x21};  // Start signal
uint8_t g_rxBuffer[2]     = {0};          // RX buffer

/* LPUART eDMA handle define */
lpuart_edma_handle_t g_lpuartEdmaHandle;

/* LPUART transfer and receive structure define */
lpuart_transfer_t xfer;
lpuart_transfer_t receiveXfer;

PRINTF("FRDM-MCXA153 LPUART 24Mbps Demo, this is the Receiver\r\n");
PRINTF("Receiving the communication start signal...\r\n");
/* Receive communication start signal */
receiveXfer.data = g_rxBuffer;
receiveXfer.dataSize = 2;
rxOnGoing = true;    // See SDK example for details
LPUART_ReceiveEDMA(LPUART2, &g_lpuartEdmaHandle, &receiveXfer);
```

```
while(rxOnGoing)    // Wait for start signal
{
}
PRINTF("Receive Done\r\n");

if((receiveXfer.data[0] == g_startBuffer[0]) && (receiveXfer.data[1] ==
 g_startBuffer[1]))
{
  PRINTF("Connect Success\r\n");
  connectSuccess = true;    // Global variable to indicate connect success
  rxBufferEmpty    = true;  // See SDK example for details
  /* Reply communication start signal */
  xfer.data        = g_startBuffer;
  xfer.dataSize = 2;
  txOnGoing     = true;    // See SDK example for details
  LPUART_SendEDMA(LPUART2, &g_lpuartEdmaHandle, &xfer);
  PRINTF("Sending the connect success signal...\r\n");
  /* Wait send finished */
  while (txOnGoing)    // See SDK example for details
  {
  }
  PRINTF("Send done\r\n");
}
else
{
  PRINTF("Connect Fail\r\n");
}
```

接收端接收和回复测试数据的核心代码如下：

```
uint16_t g_txBuffer16[2048] = {0};  // Send test data
uint16_t g_rxBuffer16[2048] = {0};  // Receive test data

PRINTF("Start to do test...\r\n");
sendXfer.txData16    = g_txBuffer16;
sendXfer.dataSize    = 2048U * 2;
receiveXfer.rxData16 = g_rxBuffer16;
receiveXfer.dataSize = 2048U * 2;

while (1)
{
 if(connectSuccess == true) // Sender and Receiver connect success
 {
    rxBufferEmpty  = true;  // See SDK example for details
    rxOnGoing = true;       // See SDK example for details
    LPUART_ReceiveEDMA(LPUART2, &g_lpuartEdmaHandle, &receiveXfer);
    while(rxOnGoing)        // See SDK example for details
    {
    }
    PRINTF("Receive Done\r\n");

    memcpy(g_txBuffer16, g_rxBuffer16, BUFFER_LENGTH * 2);
    memset(g_rxBuffer16, 0, BUFFER_LENGTH * 2);

    txOnGoing     = true;   // See SDK example for details
    LPUART_SendEDMA(LPUART2, &g_lpuartEdmaHandle, &sendXfer);
    /* Wait send finished */
    while (txOnGoing)       // See SDK example for details
    {
    }
    PRINTF("Reply Done\r\n");
 }
}
```

3.3 误码率的计算

要检查误码率，请按照以下步骤操作：

1. 统计传输次数。
2. 比较发送端发送的测试数据和接收端回复的测试数据。
3. 找出不匹配的数据并计算错误数据的数量。
4. 使用传输次数和错误数据的数量计算误码率。计算误码率的核心代码如下：

```
/**
 * @brief    UART_Check    UART print the transfer number and error data number
 * @param    NULL
 * @return   NULL
*/
void UART_Check()
{
  static uint32_t transferNum = 0;
  static uint32_t errorDataNum = 0;
  float  errorDataRate = 0;

  transferNum ++;              // transfer number +1 every time

  if (transferNum >= 1000)
  {
    uartCheckDone   = true;   // Stop the transfer after transfer number reach 1000
  }
  for(uint16_t i = 0; i < BUFFER_LENGTH; i++)   // Check whether the data is correct
  {
   if(g_rxBuffer16[i] != g_txBuffer16[i])
   {
    errorDataNum++;         // If not match, error data number +1
   }
 }
 errorDataRate = 1.0f * errorDataNum / transferNum; // Calculate the error data rate
 PRINTF("Transfer Num:%u Error Data Num:%u Error Data Rate:%f\r\n", transferNum,
 errorDataNum, errorDataRate);
}
```

3.4 时钟输出

用户可以使用时钟输出引脚来输出时钟信号并观察之。在本应用笔记中，P3_8用于输出时钟信号。使用示波器或其它设备捕捉信号并计算时钟误差。

***注：**首先，将P3_8配置为时钟输出功能。*

时钟输出的核心代码如下，在此设置后会捕获到约96MHz的信号：

```
/**
 * @brief    app_ClockOut    Clock out setting, to observe clock signal
 * @param    NULL
 * @return   NULL
*/
void app_ClockOut()
{
// Allow clock update
   SYSCON -> CLKUNLOCK             = SYSCON_CLKUNLOCK_UNLOCK(0);
// Choose FRO_HF_DIV as the source
   MRCC0  ->  MRCC_CLKOUT_CLKSEL   = MRCC_MRCC_CLKOUT_CLKSEL_MUX(1);
// Direct output
   MRCC0  ->  MRCC_CLKOUT_CLKDIV   = MRCC_MRCC_CLKOUT_CLKDIV_DIV(0);
// Freezes clock update
```

```
    SYSCON -> CLKUNLOCK             |= SYSCON_CLKUNLOCK_UNLOCK(1);
}
```

3.5 手动调整FRO_HF

在MCX A系列中，FRO_HF（FIRC）支持调整功能。这意味着用户可以手动调整时钟，以使其更接近所需的频率。通过这种方式，用户可以模拟温度和其它因素对时钟的影响。调整寄存器包括粗调（coarse trim）和细调（fine trim）功能。对于粗调，中心附近的频率变化率约为每位2.5%。对于细调，中心附近的频率变化率约为每位0.06%。手动调整FRO_HF的核心代码如下：

```
/**
 * @brief    app_FircTrim     Manual Trim FRO_HF clock
 * @param    type             TRIM_COARSE 2.5%, TRIM_FINE 0.06%
 *           value            FIRCTRIM + value * 2.5%(TRIM_COARSE) or 0.06%(TRIM_FINE)
 * @return   NULL
*/
void app_FircTrim(trim_type type,char value)
{
   uint32_t trim_value  = SCG0 -> FIRCTRIM;   // Record the initial trim value
   uint8_t  trim_coarse = (uint8_t)(trim_value >> 8);
   uint8_t  trim_fine   = (uint8_t)trim_value;
   PRINTF("%x\r\n", trim_value);
   /* UNLOCK the trim */
   SCG0 -> TRIM_LOCK = (SCG_TRIM_LOCK_TRIM_LOCK_KEY(0x5A5A)) |
 (SCG_TRIM_LOCK_TRIM_UNLOCK(1));
   if (type == TRIM_COARSE)  // Coarse trim mode, use coarse trim register
   {
    trim_coarse = trim_coarse + value;
    /* Change the trim register */
    SCG0 -> FIRCTRIM = (trim_value & 0xFFFF0000) | (trim_fine & 0xff) | ((trim_coarse &
 0x3f) << 8);
   }
   else if(type == TRIM_FINE) // Fine trim mode, use fine trim register
   {
    trim_fine = trim_fine + value;
    /* Change the trim register */
    SCG0 -> FIRCTRIM = (trim_value & 0xFFFF0000) | (trim_fine & 0xff)   | ((trim_coarse &
 0x3f) << 8);
   }
   trim_value  = SCG0 -> FIRCTRIM;
   PRINTF("%x\r\n", trim_value);  // Print the trim value
 }
```

3.6 使用外部晶振进行自动调整

如果需要更精确的时钟，则支持使用外部晶振进行自动调整。经过自动调整后，FRO_HF的精度可达±0.25%（-40~125℃），而未进行自动调整的精度为±3%（-40~125℃）。

***注：**此处需要外部晶振。在本示例中，一个8MHz的外部晶振连接到MCU。*

自动调整的核心代码如下：

```
/**
 * @brief    app_AutoTrim   Run Auto-trimming to trim the clock using external crystal
 * @param    NULL
 * @return   NULL
*/
void app AutoTrim()
{
  CLOCK_SetupExtClocking(8000000U);            // Enable the 8 MHz external crystal
```

```
  SCG0 -> FIRCTCFG |= SCG_FIRCTCFG_TRIMSRC(2); // Select the external crystal(SOSC) as
 the source
  SCG0 -> FIRCTCFG |= SCG_FIRCTCFG_TRIMDIV(7); // Divide the SOSC to 1 MHz
  SCG0 -> FIRCCSR  &= ~SCG_FIRCCSR_LK(1);      // Unlock the FIRCCSR register
  SCG0 -> FIRCCSR  |= SCG_FIRCCSR_FIRCTREN(1); // Enable auto trim
  SCG0 -> FIRCCSR  |= SCG_FIRCCSR_FIRCTRUP(1); // Enable update
  /* Until it returns 1, indicating FIRC is valid */
  while(!(SCG0 -> FIRCCSR & SCG_FIRCCSR_FIRCVLD_MASK));
  /* Until it returns 0, indicating no error */
  while(SCG0 -> FIRCCSR & SCG_FIRCCSR_FIRCERR_MASK);
  /* Until it returns 1, indicating FIRC auto trim locked to target frequency range */
  while(!(SCG0 -> FIRCCSR & SCG_FIRCCSR_TRIM_LOCK_MASK));
  SCG0 -> FIRCCSR  |= SCG_FIRCCSR_LK(1);       // Lock the FIRCCSR register
}
```

4 板上测试

要测试LPUART的功能，使用FRDM-MCXA153开发板，如图6所示。

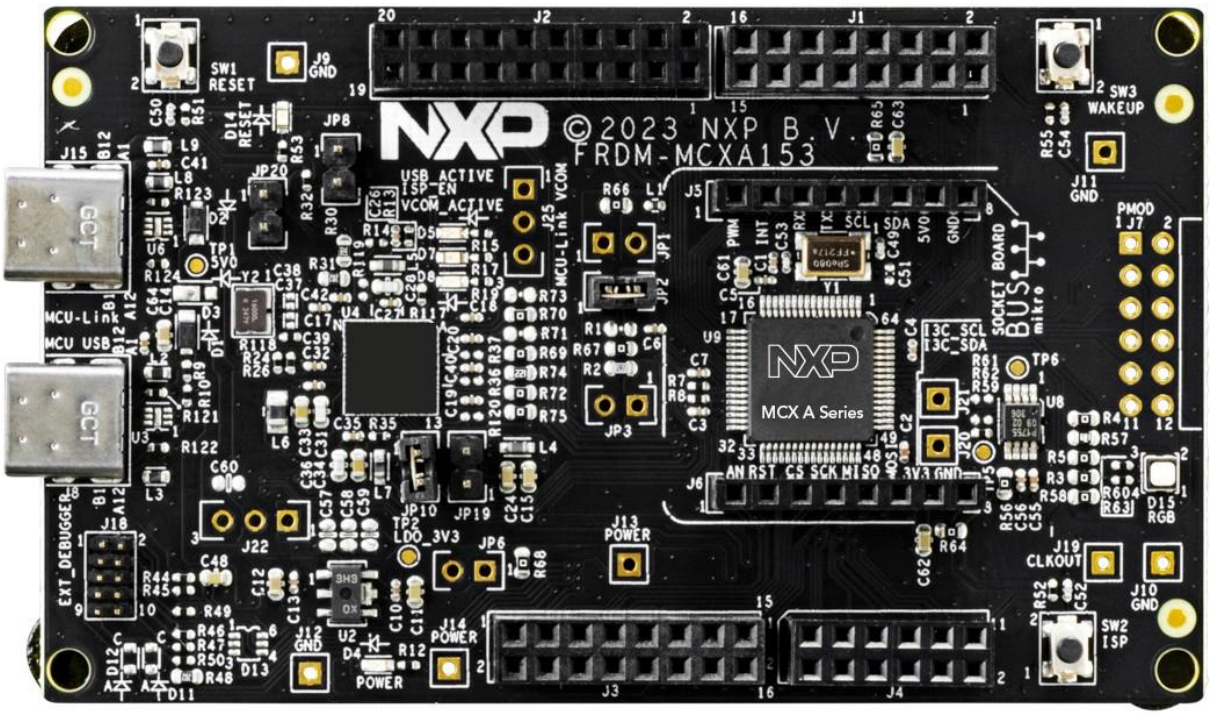

图6. FRDM-MCXA153开发板

4.1 开发板的连接

FDRM-MCXA153开发板随附一个板载调试器。使用USB-C线通过J15将开发板连接到计算机，以进行下载和调试。在此测试中，需要两块FRDM-MCXA153开发板。其连接如下所示。开发板的连接如图7所示。

- LPUART2_RXD (J1-2)——LPUART2_TXD (J1-4)
- LPUART2_TXD (J1-4)——LPUART2_RXD (J1-2)
- 5 V (J5-7)——5 V (J5-7)
- GND (J5-8)——GND (J5-8)

注：*发送端在接收端进入测试后必须进行复位。*

如果两块开发板独立供电，则无需5V连接。

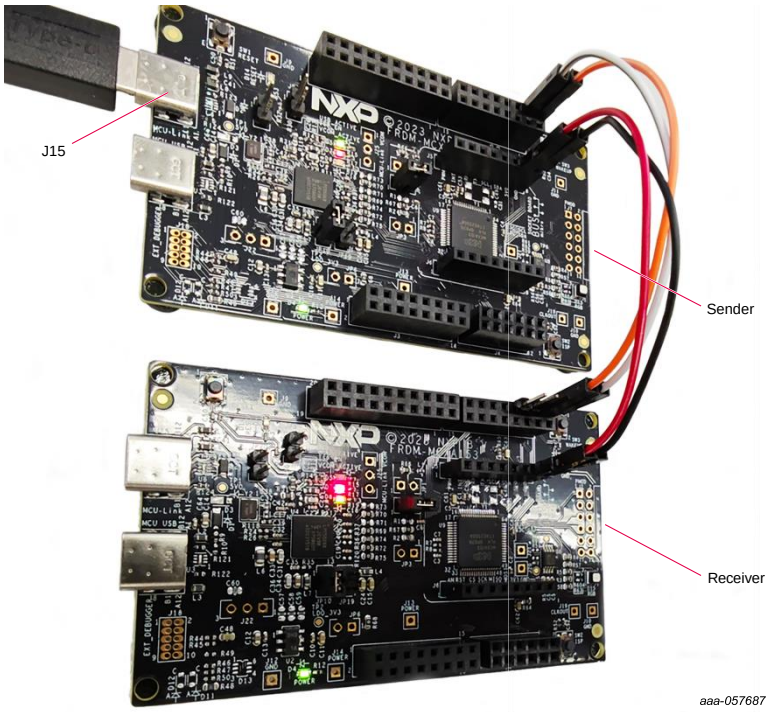

图7. FRDM-MCXA153开发板的连接

4.2 UART调试终端的设置

要观察来自开发板的调试消息，请打开UART调试终端。将终端程序设置为适当的COM端口，并使用“115200, 8, none, 1, none” （“115200，8位，无，1位，无”）设置，如图8所示。

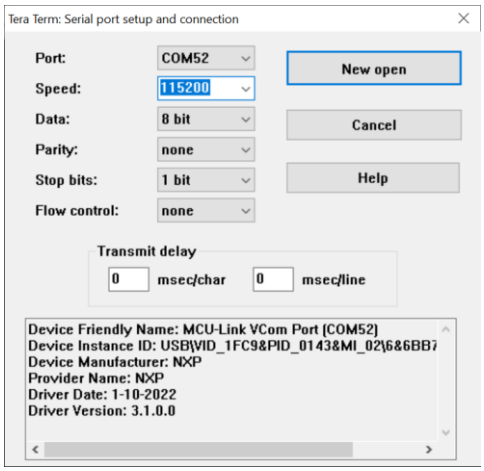

图8. UART调试终端的设置

4.3 测试结果

依次按下接收端和发送端上的RESET（复位）按钮，24Mbit/s的LPUART通讯将开始。测试结果会打印到UART调试终端。测试结果如图9所示。发送端和接收端完成了1000次测试数据传输，未发现错误数据。

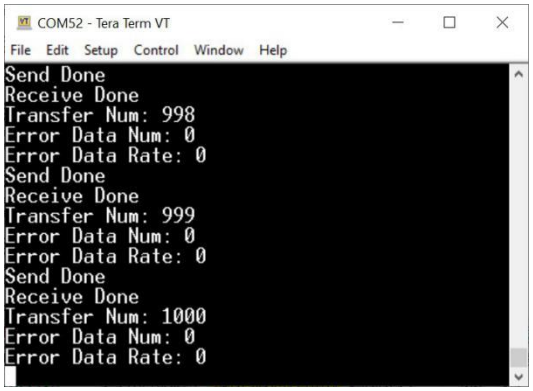

图9. UART调试终端的测试结果

5 结论

本应用笔记提供了相关信息和代码，帮助用户在MCX A系列上达到24 Mbps的LPUART最大波特率。结果表明，此种通讯是可靠的。

6 修订历史

表1. 修订历史

文档ID	发布日期	说明
AN14458 v.1.0	2024年10月17日	初始版本

7 关于本文中源代码的说明

本文中所示的示例代码具有以下版权和BSD-3-Clause许可：

2024年恩智浦版权所有；在满足以下条件的情况下，可以源代码和二进制文件的形式重新分发和使用本源代码（无论是否经过修改）：

1. 重新分发源代码必须保留上述版权声明、这些条件和以下免责声明。
2. 以二进制文件形式重新分发时，必须在文档和/或随分发提供的其他材料中复制上述版权声明、这些条件和以下免责声明。
3. 未经事先书面许可，不得使用版权所有者的姓名或参与者的姓名为本软件的衍生产品进行背书或推广。

本软件由版权所有者和参与者“按原样”提供，不承担任何明示或暗示的担保责任，包括但不限于对适销性和特定用途适用性的暗示保证。在任何情况下，无论因何种原因或根据何种法律条例，版权所有者或参与者均不对因使用本软件而导致的任何直接、间接、偶然、特殊、惩戒性或后果性损害（包括但不限于采购替代商品或服务；使用损失、数据损失或利润损失或业务中断）承担责任，无论是因合同、严格责任还是侵权行为（包括疏忽或其他原因）造成的，即使事先被告知有此类损害的可能性也不例外。

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com.cn/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

目录