RT1021 LPUART 使能接收 FIFO

Calvin Ji 2020.12

North China CAS Team

在一些对串口波特率要求较高且一帧连续数据包很长的场合，标准的串口接收机制（每接收一个字节产生一次中断）会有较大的缺陷，其一中断太频繁且间隔很短严重占据 CPU 资源，其二存在接收太快 CPU 处理不及导致的数据被覆盖进而丢包的风险。通常我们处理此种情况的应对措施一般是通 UART 接收+DMA 的方式，而本文档不提这种传统方式，而是利用 i.MXRT LPUART IP 的接收 FIFO 和 RingBuffer 结合的方法来解决此类问题。

RT1020/RT1050/RT1060 的 LPUART 串口 FIFO 深度为 4，可以通过配置 Watermark 功能（不能大于 4）来决定串口接收到几个字节之后才会产生一次中断（大于 Watermark 寄存器中 RXWATER 值才会置位 RDRF 即接收满中断）如图 1，然后在接收满中断里通过查询 RXCOUNT 值来判断接收到多少字节并逐一读取接收 FIFO 直到 RXCOUNT 为 0 结束（每次读取接收 FIFO，RXCOUNT 自动减 1）。不过淡出这种方式存在一个潜在问题，即当一帧数据的最后几个字节少于或者等于 watermark 值时，并不能产生 RDRF 接收满中断，从而导致最后这几个字节不能被正常读取直到下一帧数据过来时才能取出来上一帧的剩余数据，这当然不是我们想要的。幸运的是，RT 的 LPUART 提供了 Rx FIFO IDLE 功能如图 2，即当一帧数据结束，接收器空闲时（空闲的条件可配置）且 Rx FIFO 里不为空（即 Rx FIFO 仍然有低于 watermark 阈值的残余数据）也会触发一次 RDRF 接收满中断，这样我们就可以在最后一次中断里把剩余数据读出来送到 ringbuffer 里。这样 LPUART Rx FIFO+Ringbuffer 的方式就可以较好的改善前面提到的应用场合。

43.4.1.13.4 Fields

Field	Function
31-27 —	Reserved
26-24 RXCOUNT	Receive Counter The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer.
23-18 —	Reserved
17-16 RXWATER	Receive Watermark When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE] and must be greater than 0.
15-11 —	Reserved

LPUART_WATER寄存器

Table continues on the next page...

12-10 RXIDEN	Receiver Idle Empty Enable When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty. 000b - Disable RDRF assertion due to partially filled FIFO when receiver is idle. 001b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character. 010b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters. 011b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters. 100b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters. 101b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters. 110b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters. 111b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters.
9 TXOFE	Transmit FIFO Overflow Interrupt Enable When this field is set, the TXOF flag generates an interrupt to the host. 0b - TXOF flag does not generate an interrupt to the host. 1b - TXOF flag generates an interrupt to the host.
8	Receive FIFO Underflow Interrupt Enable

LPUART_FIFO寄存器

Table continues on the next page...

测试环境：

SDK 版本：SDK_v2.6.1
IDE 工具：IAR_v8.40.2
开发板：MIMXRT1021-EVK

软件配置：

1. 首先在串口配置出使能 FIFO watermark 功能且配置其位最大 FIFO 深度减 1，因为只有接收字节数大于该 watermark 值时才会触发 RDRF 接收满中断；

```
    /*
     * config.baudRate_Bps = 115200U;
     * config.parityMode = kLPUART_ParityDisabled;
     * config.stopBitCount = kLPUART_OneStopBit;
     * config.txFifoWatermark = 0;
     * config.rxFifoWatermark = 0;
     * config.enableTx = false;
     * config.enableRx = false;
     */
    LPUART_GetDefaultConfig(&config);
    config.baudRate_Bps = BOARD_DEBUG_UART_BAUDRATE;
    config.enableTx     = true;
    config.enableRx     = true;
    /* Enable Rx FIFO with max. supported fifo size-1, max. 4 depth in RT1021 */
    config.rxFifoWatermark = FSL_FEATURE_LPUART_FIFO_SIZEn(DEMO_LPUART)-1;
```

2. 由于原 SDK 的 LPUART 底层驱动文件里并没有提供上述提到的 RXIDEN 即 Rx FIFO 空闲配置，而且为了保持 SDK 兼容性，我们并没有去修改底层驱动文件，而是在 LPUART_Init 之后手动使能 RXIDEN 功能，注意参考手册说明，RXIDEN 在使能前需要先关掉 LPUART 的收发器且 FIFO 里不能有数据，所以需要添加如下图代码；

```
    LPUART_Init(DEMO_LPUART, &config, DEMO_LPUART_CLK_FREQ);
    /* Enable the assertion of RDRF when the receiver is idle with FIFO Enabled */
    LPUART_EnableTx(DEMO_LPUART, false);LPUART_EnableRx(DEMO_LPUART, false);
    DEMO_LPUART->FIFO |= (LPUART_FIFO_TXFLUSH_MASK | LPUART_FIFO_RXFLUSH_MASK);//Clear the FIFO firstly
    DEMO_LPUART->FIFO |= LPUART_FIFO_RXIDEN(1);
    LPUART_EnableTx(DEMO_LPUART, true);LPUART_EnableRx(DEMO_LPUART, true);
```

3. 使能串口接收满中断，并且同时把串口的一些常见错误中断也使能尤其时 RxOverrun 这种错误，LPUART 在出现 RxOverrun 错误时如果该标志不及时清理会阻塞后续接收，所以该中断必须使能；

```
    /* Enable RX interrupt. */
    LPUART_EnableInterrupts(DEMO_LPUART, kLPUART_RxDataRegFullInterruptEnable
                                       | kLPUART_RxOverrunInterruptEnable
                                       | kLPUART_NoiseErrorInterruptEnable
                                       | kLPUART_FramingErrorInterruptEnable);
```

4. 中断服务函数里的处理如下图，即在接收满中断里读取 RXFIFO 的数据（RxFIFO 时通过 LPUART_D 寄存器来访问）知道 RXCOUNT 为 0 退出，而在错误中断处理这里，做了简单优化，即在 LPUART 的中断里由于绝大数时间不会产生错误而是以接收满为主，所以在判断错误标志时先统一判断下是否有任意一个错误中断，有的话才进去逐一判断是哪个错误，而不是直接逐一判断，这样相对节省些中断处理时间；

```
void DEMO_LPUART_IRQHandler(void)
{
    uint8_t data;
    /* If new data arrived. */
    if ((kLPUART_RxDataRegFullFlag)&LPUART_GetStatusFlags(DEMO_LPUART))
    {
      do{
        data = LPUART_ReadByte(DEMO_LPUART);

        /* If ring buffer is not full, add data to ring buffer. */
        if (((rxIndex + 1) % DEMO_RING_BUFFER_SIZE) != txIndex)
        {
            demoRingBuffer[rxIndex] = data;
            rxIndex++;
            rxIndex %= DEMO_RING_BUFFER_SIZE;
        }
      }while(DEMO_LPUART->WATER & LPUART_WATER_RXCOUNT_MASK);//Read out all data inside FIFO
    }
    /* Use multi flags OR operation to save time in lpuart interrupt */
    if((kLPUART_RxOverrunFlag|kLPUART_NoiseErrorFlag|kLPUART_FramingErrorFlag)&LPUART_GetStatusFlags(DEMO_LPUART))
    {
      if(kLPUART_RxOverrunFlag & LPUART_GetStatusFlags(DEMO_LPUART))
        LPUART_ClearStatusFlags(DEMO_LPUART, kLPUART_RxOverrunFlag);
      if(kLPUART_NoiseErrorFlag & LPUART_GetStatusFlags(DEMO_LPUART))
        LPUART_ClearStatusFlags(DEMO_LPUART, kLPUART_NoiseErrorFlag);
      if(kLPUART_FramingErrorFlag & LPUART_GetStatusFlags(DEMO_LPUART))
        LPUART_ClearStatusFlags(DEMO_LPUART, kLPUART_FramingErrorFlag);
```