类别	应用指南
编制	洪星
日期	2018/8/9

HOPERF®

CMT2300 可变长包数据收发应用分析

1 应用描述

使用 CMT2300 包模式进行数据收发，可以极大简化用户射频数据收发操作。包模式分固定包长模式和可变包长模式两种。固定包长模式采用预设固定包长，数据帧中不带包长字段，包长范围 1~2048Byte。可变包长模式中数据帧带一个字节的包长字段，因此包长范围 1~256byte。虽然固定包长模式下，包长范围更大，但由于数据帧没有包长字段，收发两端只能通过预设包长进行统一，包长灵活性不如可变包长模式。因此，在长包数据收发中一般推荐使用可变包长模式。

CMT2300 提供两个独立 32Byte 的 FIFO，合并使用后 FIFO 最大为 64Byte。当收发数据帧长度超过 64Byte，需要多次读写 FIFO 进行数据帧收发。

使用 CMT2300 可变包长模式进行数据收发时，需要注意 FIFO 空间的读写操作，以及 FIFO 读写中断的使用。

在本文中主要介绍使用 CMT2300 可变包长模式进行数据收发的几种方法和需注意的问题。

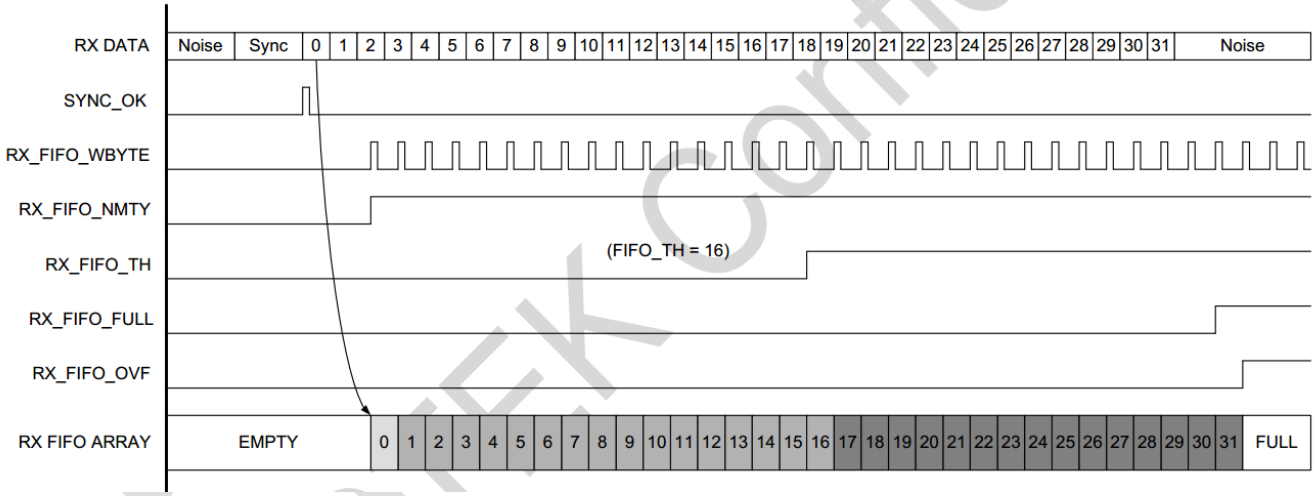
2 超过 64Byte 的 FIFO 读写操作

CMT2300 收发数据帧长度超过 64Byte 时，无法一次完成读写，就需要进行多次读写操作。

CMT2300 接收数据时，当检测 SyncWord 成功后就会接收 payload 数据，没接收 1Byte 数据就会往 RxFIFO 中写 1Byte 数据。用户需要在 RxFIFO 被写满之前从中读出数据，否则未及时读出的数据会被覆盖，造成接收数据缺失。

CMT2300 发送数据长度超过 64Byte 时，就需要分批多次写入 TxFIFO。CMT2300 每发送 1Byte 数据，TxFIFO 中就会空出 1Byte 空间，以便用户写入后面的数据。而写 TxFIFO 的过程都是在 Tx 状态中进行的，用户写 TxFIFO 数据不能过快也不能过慢，必须保持发送数据的连续性。

为帮助用户方便可靠读写 FIFO，CMT2300 提供多种中断标志机制以及读写方式。



图、CMT2300 RX FIFO 中断时序示意图

超过 64Byte 的 RxFIFO 读操作方法：

方法 1：使用 RX_FIFO_TH 中断

在开启 CMT2300 进入接收状态后，检测 RX_FIFO_TH 中断，一旦中断有效表示 FIFO 已经被填入预设的数据长度，就可以读取。根据读出的数据的长度字段，可判断剩余待接收字节长度，并配合 PKT_OK 中断读取最后的数据字节。（最后一帧数据长度可能小于 RX_FIFO_TH 阈值，而无法触发 RX_FIFO_TH 中断）

方法 2：使用 RX_FIFO_NMTY 中断

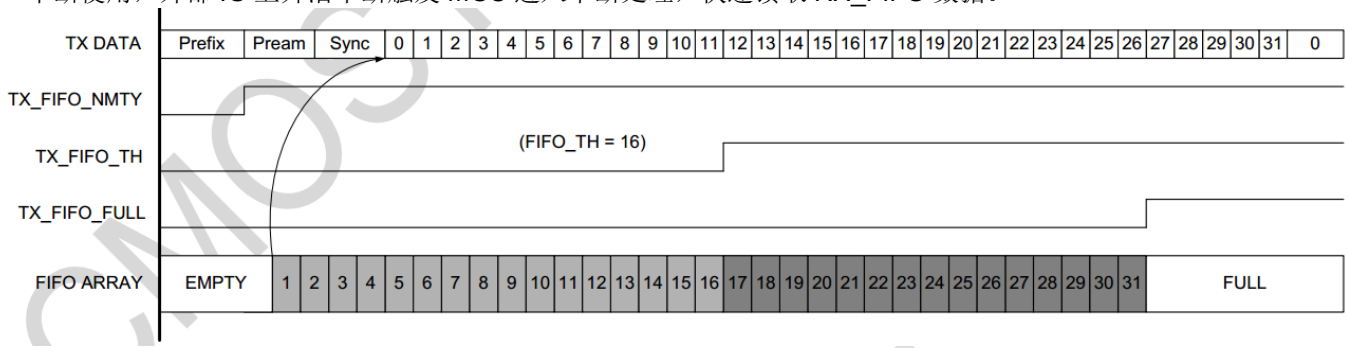
检测 RX_FIFO_NMTY 中断，一旦有效立即进行读取 1Byte，直到该中断无效。再次检测该中断有效时再读取 1Byte，如此反复可以实现边接收边读取。但需注意的是每次只读 1Byte，并判断中断是否有效。

方法 3：使用 RX_FIFO_WBYTE 中断

检测 RX_FIFO_WBYTE 中断，一旦中断有效便立即读取 1Byte。如此，芯片接收写入 RX_FIFO 1Byte，便可以读出 1Byte。

实现这样操作，要求 SPI 的读写速度要比芯片接收数据的速度要快 2 倍。一般这种方式需配合 MCU 的外部 IO

中断使用，外部 IO 上升沿中断触发 MCU 进入中断处理，快速读取 RX_FIFO 数据。



图、CMT2300 TX FIFO 中断示意图

超过 64Byte 的 TxFIFO 写操作方法：

方法 1：使用 TX_FIFO_TH 中断方式

首先一次写满 TX_FIFO，启动 CMT2300 发送后，持续检测 TX_FIFO_TH 中断，当中断无效后，写满 TX_FIFO 的剩余空间，如此循环写入所有需要发送的字节。

方法 2：检测 TX_FIFO_FULL 中断方式

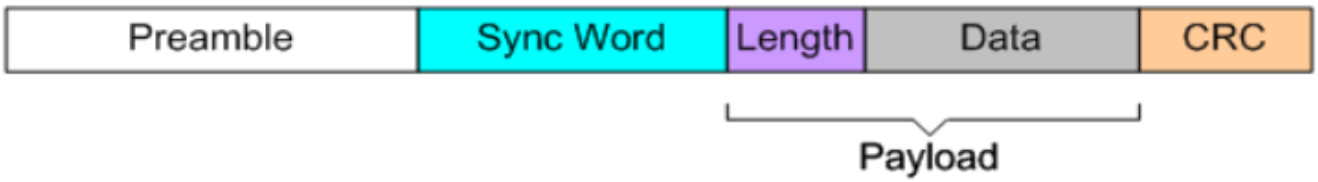
首先一次写满 TX_FIFO，启动 CMT2300 发送后，持续检测 TX_FIFO_FULL 中断是否有效，当中断无效时，写入 1Byte，如此循环写入所有需要发送的字节。

需要注意，无论是 TX_FIFO_TH 中断还是 TX_FIFO_FULL 中断，均需要在检测中断无效后再进行写 TX_FIFO，这与正常操作逻辑相反！

另外不推荐使用 TX_FIFO_NMTY，因为射频发送需要保持数据的连续性，即 TX_FIFO 准备数据的及时性。如果使用 TX_FIFO_NMTY 中断，在检测 TX_FIFO 为空时再去 TX_FIFO 数据可能来不及，而造成发送数据错误。

3 可变长包接收操作

使用 CMT2300 进行可变长包接收，用户需要判断数据接收的完整性，以及读 RX_FIFO 的及时性。



图、可变长包数据帧格式（NodeID 不存在）

在可变长数据包中 Payload 的第一个 Byte 为 Data 长度（不包括长度字节本身）。用户从 RX_FIFO 中读出的第一个字节即为后续待接收数据的长度。用户需要根据这个字节判断接收数据的完整性，并配置 RX_FIFO 的各种中断标志及时读取 RX_FIFO 数据。

在无法判断最后一次读 RX_FIFO 长度的情况下，还可以结合 PKT_OK 中断判断芯片接收是否完成。

配置 CMT2300 进行接收（可变长包）示例代码：

```
void Radio_GoRx_VariableLen(void)
{
    CMT2300A_GoStby();
    CMT2300A_SetFifoThreshold(16);  // FIFO_TH
    // Config GPIOs
    CMT2300A_ConfigGpio(CMT2300A_GPIO1_SEL_INT1|
                        CMT2300A_GPIO2_SEL_INT2|
                        CMT2300A_GPIO3_SEL_DOUT);
    // Config interrupt
    CMT2300A_ConfigInterrupt(
        CMT2300A_INT_SEL_PKT_OK,         // PKT_OK
        CMT2300A_INT_SEL_RX_FIFO_TH      // TX_FIFO_TH
        //CMT2300A_INT_SEL_RX_FIFO_NMTY // TX_FIFO_NMTY
    );

    CMT2300A_EnableReadFifo();
    CMT2300A_ClearFifo();
    CMT2300A_GoRx();
}
```

查询并接收（可变长包）示例代码：

```
u16 Radio_Recv_VariableLen(u8 pBuf[])
{
    u16 rxLen = 0;
    do
    {
        if(CMT2300A_ReadGpio2())  // RX_FIFO_TH/RX_FIFO_NMTY
        {
            CMT2300A_ReadFifo(&pBuf[rxLen], 1);
            rxLen += 1;
        }
        if(CMT2300A_ReadGpio1())  // PKT_OK
        {
            u16 pktLen;
            u16 remainLen;

            CMT2300A_GoStby();
            if(rxLen < 1)
            {
                CMT2300A_ReadFifo(&pBuf[0], 1);
                rxLen += 1;
            }
            pktLen = pBuf[0];
            remainLen = (pktLen-rxLen)<32 ? (pktLen-rxLen) : 32;
            if(remainLen > 0)
            {
                CMT2300A_ReadFifo(&pBuf[rxLen], remainLen);
                rxLen += remainLen;
            }

            CMT2300A_ClearFifo();
            CMT2300A_ClearInterruptFlagsA();
            CMT2300A_GoSleep();
            return rxLen;
        }
    }while(rxLen);
    return rxLen;
}
```

4 可变长包发送操作

使用 CMT2300 可变长包模式进行数据发送，需要注意：

1、 CMT2300 发送的 payload 数据中长度字节是芯片自动添加，不属于用户写 TX_FIFO 数据。在 Go_Tx 前用户需要通过写寄存器的方式配置 payload Length，否则会造成发送数据异常。

2、 射频发送需要保持数据的连续性。在用户所有数据发送完成前，需保持 TX_FIFO 非空，且一定要避免 TX_FIFO 溢出。

3、 在用户数据全部写入 TX_FIFO 后，需判断芯片发送完成才能配置芯片退出。用户可以通过 TX_DONE 中断判断芯片是否发送完成。

4、 为提高程序的可靠性，建议在 CMT2300 启动发送后，进行超时判断。如果芯片在充足时间段内没有发送完成，则清除退出或置位后重新发送。

CMT2300 发送的 payload

可变长包发送示例代码：

```
u8 Radio_Send_VariableLen(const u8 pBuf[], u8 len)
{
    u8 wrLen = 0;
    u32 delay;

    CMT2300A_GoStby();
    CMT2300A_ConfigGpio(CMT2300A_GPIO1_SEL_INT1 |
                        CMT2300A_GPIO2_SEL_INT2 |
                        CMT2300A_GPIO3_SEL_DOUT);
    CMT2300A_ConfigInterrupt(
        CMT2300A_INT_SEL_TX_DONE,       // TX_DONE 发送完成中断
        CMT2300A_INT_SEL_TX_FIFO_TH);  // TX_FIFO_TH 超过阈值中断 \ TX_FIFO_FULL 满中断
    CMT2300A_ClearInterruptFlagsA();
    CMT2300A_SetPayloadLength(len);  // 设置 payload 长度（必须!!）
    CMT2300A_EnableWriteFifo();
    CMT2300A_ClearFifo();

    wrLen = (len < 32) ? len : 32;
    CMT2300A_WriteFifo(pBuf, wrLen);  // 写 TX_FIFO
    CMT2300A_GoTx();  // 启动发送
    delay = 5000;
    while(delay--)
    {
        if(CMT2300A_ReadGpio1())  // TX_DONE
        {
            CMT2300A_ClearInterruptFlagsA();
            CMT2300A_GoSleep();
            return 1;  // ·¢ËÍÍê³É
        }
        if(CMT2300A_ReadGpio2() == 0)  // TX_FIFO_TH/TX_FIFO_FULL
        {
            if(wrLen < len)
            {
                CMT2300A_WriteFifo(&pBuf[wrLen], 1);  // 写 TX_FIFO
                wrLen++;
            }
        }
        system_delay_100us(1);
    }
    CMT2300A_ClearInterruptFlagsA();
    CMT2300A_GoSleep();
    return 0;  // 发送超时
}
```

5 注意事项

为使用 CMT2300 可靠进行数据收发，在程序中需注意几个问题：

1、 TX_FIFO 相关中断的反逻辑判断。相比 RX_FIFO 相关中断的逻辑判断，在使用 TX_FIFO 中断进行写数据时需利用其中断逻辑反向性，即需要判断 TX_FIFO 中断无效后再进行操作。

2、 控制 CMT2300 在进行数据发送时，需进行发送超时保护。即在充足时间内芯片没有检测到芯片发送完成，则需要进行超时退出，或者置位芯片重新进行发送，避免程序卡死。

3、 在 CMT2300 接收数据时，需要进行接收超时保护和接收是否完整判断。当检测到 RX_FIFO 开始有数据时，及时进行数据接收缓存。同时在充足时间内没有检测到数据接收完成，则需要重置芯片进行接收，避免程序卡死，无法进行接收。

4、 如果 MCU 中还有使用其他中断，建议在读写 FIFO 操作前先屏蔽全局中断，完成操作后在开启中断。

5、 使用 CMT2300 可变长包模式进行数据收发，发送数据时，payload 长度字节不属于用户数据内容，由芯片自动添加。而接收数据时，芯片会判断 SycnWord 后的首字节（RX_FIFO 第一个字节）为长度，并以此判断接收数据的接收。收发两端必须统一数据帧长度字节规则。

6、 CMT2300 可变长包格式中，payload 长度字节不包含 Length 字段本身。如果开启 CMT2300 中 CRC 校验功能，则 payload 长度字节包含 CRC 校验字段长度（2 字节）。这个规则有别于一些标准协议的定义。

7、 建议将 CMT2300 的中断映射到 GPIO 上，通过查询 IO 电平判断中断是否有效。避免 SPI 读写寄存器过慢或过于频繁，导致芯片操作异常。