SPC1169 UART 使用指南

版本 A/0 – 2023 年 4 月

概述

通用异步收发器（UART）能够灵活地与外部设备进行全双工数据交换，常用于短距离、低速的串行通信中，UART 通过可编程的波特率产生器提供了多种波特率。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
1	2023 年 4 月 28 日	XuQing He	Released	首次发布。

术语或缩写

术语或缩写	描述

1 UART 特性

SPC1169 内建两个 UART 单元功能并且兼容 16550A 和 16750 工业标准，SPC1169 的 UART 单元有以下特点：

- 最高支持 6.25Mbps；
- 异步收发器无需时钟；
- 支持奇、偶、无奇偶校验；
- 内建 64 Byte 接收缓存和 64 Byte 发送缓存；
- 可编程的 FIFO 接收中断阈值；
- 支持自动波特率检测；
- 支持不归零编码（NRZ）；
- 兼容 LIN1.3、2.0、2.1 和 2.2A 协议；

2 UART 波特率设置

由于 UART 是异步收发器无需时钟信号，只需双方约定好通信的波特率，而 Spintrol 提供了手动设置波特率和自动波特率两种方法。

2.1 手动波特率

手动波特率是用户通过配置 UARTBDCNT 寄存器来指定分频比，计算出 UART 的波特率。UART 接收器内部需要在每位数据内做 16 倍过采样，所以 UARTBDCNT 必须至少是 16。如果用户写入一个小于 16 的数，则会被硬件改写为 16。

$$Baud\ rate = \frac{f_{uart_{clk}}}{UARTBDCNT}$$

f_{uart_clk}为 UART 模块的时钟频率，可以通过 CLOCK_GetModuleClock()函数获取，$Baud\ rate$为 UART 的波特率。

2.2 自动波特率

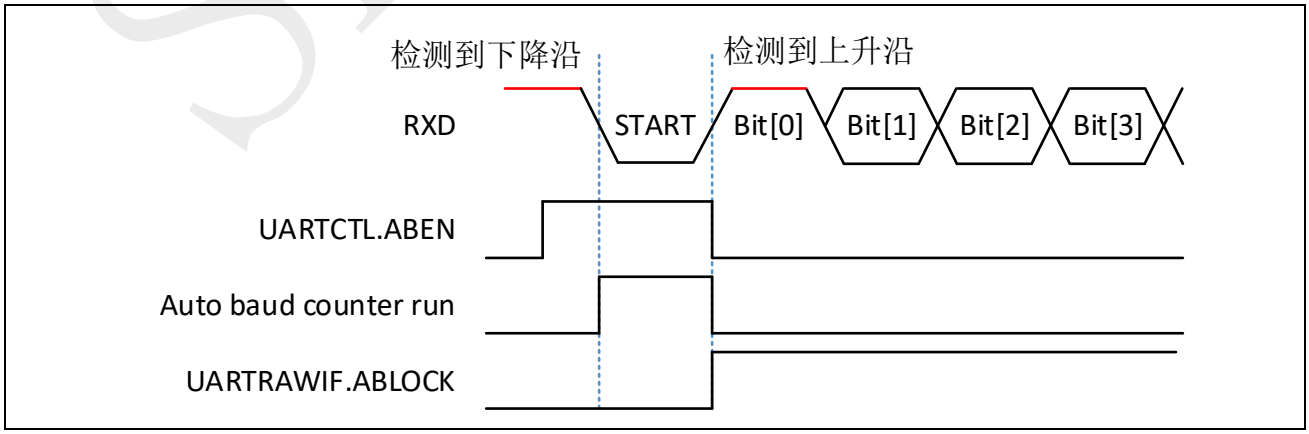
自动波特率是 UART 检测 RX 引脚的起始信号低电平的时间，并将此时间自动更新到 UARTBDCNT 寄存器。通过对 UARTCTL 寄存器的 ABEN 位进行置位可以使能自动波特率功能，自动波特率结束后 ABEN 位会被自动清零。

实现自动波特率功能对 RX 信号需要满足以下两个条件：

- 开始自动波特率功能前 RX 信号的默认电平必须为高电平；
- UART 接收的第一个比特的数据必须为高电平；

当上述条件满足并使能了自动波特率时，RX 接收到下降沿信号时，UART 的自动波特率的计数器开始计数，直到 RX 信号出现上升沿停止计数，UARTRAWIF 寄存器的 ABLOCK 位会被置位，UARTBDCNT 寄存器的值会更新。

图 2-1：UART 自动波特率



3 UART 实例

3.1 Poll 传输模式

3.1.1 先发送数据后接收数据

本示例中演示 UART 使用 Poll 传输模式，先进行发送数据，然后再接收数据，最后对发送的数据和接收的数据进行比较。示例可以通过是否配置 WORD_TEST 宏进行按字节还是字来传输，其示例如下：

Poll 传输(先发送数据后接收数据)

```
#include <stdio.h>

#if defined(SPD1179)
    #include "spd1179.h"
#else
    #include "spc1169.h"
#endif
#include <stdlib.h>


#define               DatumCount                4
#define               WORD_TEST


uint16_t              i;
uint32_t              u32RXData;                                  /* Receive word
*/
uint32_t              u32TXData;                                  /* Send word */
uint8_t               u8RXData;                                  /* Receive byte
*/
uint8_t               u8TXData;                                  /* Send byte */


uint32_t UART_RX_Word(void)
{
    volatile uint32_t i = 0;
    uint32_t u32Data = 0;
    uint32_t u32Bit;

    for (i = 0; i < 4; i++)
    {
        /* Wait until data is available in RBR or the FIFO */
        while (!UART_GetStatus(UART1, UART_STS_RX_NOT_EMPTY))
        {
        }

        /* In receive process, the MSB come first and the LSB comes the last */
        u32Bit = ((3 - i) * 8);
        u32Data |= UART_ReadByte(UART1) << u32Bit;
    }

    return u32Data;
}


void UART_TX_Word(uint32_t u32Data)
{
    volatile uint32_t i = 0;
```

```
    uint32_t u32Mask;
    uint32_t u32Bit;

    for (i = 0; i < 4; i++)
    {
        /*
        * Write the datum to UART Transmit Holding Register, which will be sent
out,
        * the MSB sent out first.
        */
        u32Mask = 0xFF000000 >> (i * 8);
        u32Bit = (3 - i) * 8;
        UART_WriteByte(UART1, (u32Data & (u32Mask)) >> (u32Bit));

        /* Wait for the TX FIFO empty */
        while (!UART_GetStatus(UART1, UART_STS_TX_EMPTY));
    }
}


uint8_t UART_RX_Byte(void)
{
    uint8_t u8Data = 0;

    /* Wait until data is available in RBR or the FIFO */
    while (!UART_GetStatus(UART1, UART_STS_RX_NOT_EMPTY));

    /* In receive process, the MSB come first and the LSB comes the last */
    u8Data = UART_ReadByte(UART1);

    return u8Data;
}


void UART_TX_Byte(uint8_t u8Data)
{
    /* Write the datum to UART Transmit Holding Register */
    UART_WriteByte(UART1, u8Data);

    /* Wait for the TX FIFO empty */
    while (!UART_GetStatus(UART1, UART_STS_TX_EMPTY));
}


/*****************************************************************************
*****************************************
 *
 * @brief      In this case, UART using polling model to transmit datum, and the
other UART end communicating with this
 *             UART end will sent these datum back, The TX UART end in this demo
will receive the back datum and do a
 *             double check.
 *
 *
 *           This demo is a couple with another named 'UART_RX_and_SentBack'.
 *

*****************************************************************************
*****************************************/


int main(void)
{
```

```
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    /*
    * Init the UART1
    *
    * 1.Set the GPIO25/26 as UART FUNC
    *
    * 2.Enable the UART CLK
    *
    * 3.Set the rest para
    */
    PIN_SetChannel(PIN_GPIO6, PIN_GPIO6_UART1_TXD);
    PIN_SetChannel(PIN_GPIO7, PIN_GPIO7_UART1_RXD);
    UART_Init(UART1, 38400);

    printf("Enter the test\n");

#ifdef WORD_TEST
    /* Generate the world datum to be sent and sent them out */
    for (i = 0; i < DatumCount; i++)
    {
        u32TXData = rand() & 0xFFFFFFFF;
        UART_TX_Word(u32TXData);
        u32RXData = UART_RX_Word();
        printf("The u32TXData is %x\n", u32TXData);
        if (u32TXData != u32RXData)
        {
            printf("The test is Error\n");
            return 0;
        }
    }
    printf("The test is pass\n");
#else
    /* Generate the byte datum to be sent and sent them out */
    for (i = 0; i < DatumCount; i++)
    {
        u8TXData = rand() & 0xFF;
        UART_TX_Byte(u8TXData);
        u8RXData = UART_RX_Byte();

        printf("The u8TXData is %x\n", u8TXData);

        if (u8TXData != u8RXData)
        {
            printf("The test is Error\n");
            return 0;
        }
    }
    printf("The test is pass\n");
#endif

    while (1)
    {
```

```
    }
}
```

3.1.2 先接收数据后发送数据

本示例中演示 UART 使用 Poll 传输模式，把接收到的数据再进行发送。示例可以通过是否配置 WORD_TEST 宏进行按字节还是字来传输，其示例如下：

Poll 模式(从机发送与接收)

```
#include <stdio.h>

#if defined(SPD1179)
    #include "spd1179.h"
#else
    #include "spc1169.h"
#endif


#define             WORD_TEST


uint32_t            u32RXData;                                  /* Receive word */
uint8_t             u8RXData;                                   /* Receive byte */


uint32_t UART_RX_Word(void)
{
    volatile uint32_t i = 0;
    uint32_t u32Data = 0;
    uint32_t u32Bit;

    for (i = 0; i < 4; i++)
    {
        /* Wait until data is available in RBR or the FIFO */
        while (!UART_GetStatus(UART1, UART_STS_RX_NOT_EMPTY))
        {
        }

        /* In receive process, the MSB come first and the LSB comes the last */
        u32Bit = ((3 - i) * 8);
        u32Data |= UART_ReadByte(UART1) << u32Bit;
    }

    return u32Data;
}


void UART_TX_Word(uint32_t u32Data)
{
    volatile uint32_t i = 0;
    uint32_t u32Mask;
    uint32_t u32Bit;

    for (i = 0; i < 4; i++)
    {
        /*
        * Write the datum to UART Transmit Holding Register, which will be sent
out,
```

```
        * the MSB sent out first.
        */
        u32Mask = 0xFF000000 >> (i * 8);
        u32Bit = (3 - i) * 8;
        UART_WriteByte(UART1, (u32Data & (u32Mask)) >> (u32Bit));

        /* Wait for the TX FIFO empty */
        while (!UART_GetStatus(UART1, UART_STS_TX_EMPTY));
    }
}


uint8_t UART_RX_Byte(void)
{
    uint8_t u8Data = 0;

    /* Wait until data is available in RBR or the FIFO */
    while (!UART_GetStatus(UART1, UART_STS_RX_NOT_EMPTY));

    /* In receive process, the MSB come first and the LSB comes the last */
    u8Data = UART_ReadByte(UART1);

    return u8Data;
}


void UART_TX_Byte(uint8_t u8Data)
{
    /* Write the datum to UART Transmit Holding Register */
    UART_WriteByte(UART1, u8Data);

    /* Wait for the TX FIFO empty */
    while (!UART_GetStatus(UART1, UART_STS_TX_EMPTY));
}


/*****************************************************************************
*****************************************
 *
 * @brief      In this case, UART using polling model receive the datum come
from transmitting end, and sent these datum
 *            back.
 *
 *

*****************************************************************************
*****************************************/


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART0
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    /*
```

```
     * Init the UART1
     */
    PIN_SetChannel(PIN_GPIO6, PIN_GPIO6_UART1_TXD);
    PIN_SetChannel(PIN_GPIO7, PIN_GPIO7_UART1_RXD);
    UART_Init(UART1, 38400);

    printf("Enter the test\n");

#ifdef WORD_TEST
    /* Receive the byte data comes from the transmitting end, and sent it back
*/
    while (1)
    {
        u32RXData = UART_RX_Word();
        UART_TX_Word(u32RXData);
        printf("u32RXData is %x\n", u32RXData);
    }
#else
    /* Receive the byte data comes from the transmitting end, and sent it back
*/
    while (1)
    {
        u8RXData = UART_RX_Byte();
        UART_TX_Byte(u8RXData);
        printf("u8RXData is %x\n", u8RXData);
    }
#endif
}
```

3.2 中断传输模式

本示例中演示使用 Spintrol 的 ISP 串口工具进行发送数据，SPC1169 进行接收数据，当 SPC1169 接收到的数据大于所设的接收 FIFO 阈值时，会触发中断，然后在中断中对接收到的数据进行打印，其示例如下：

中断传输模式

```
#include <stdio.h>

#if defined(SPD1179)
    #include "spd1179.h"
#else
    #include "spc1169.h"
#endif


uint8_t           u8Data;                       /* Receive data */


/*****************************************************************************
*****************************************
 *
 * @brief      In this case, UART read data using INT model, when there is one
byte in the receive FIFO, the RX INT will be
 *            triggered, then, the data will be receive in the INT function.
 *

*****************************************************************************
*****************************************/


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* Set the receive fifo INT triger level at least 2 byte */
    UART_SetRxFIFOThreshold(UART0, 1);

    /* Enable data reaching INT */
    UART_EnableInt(UART0, UART_INT_RX_REQ);

    /* Enable the Gloable INT of UART */
    NVIC_EnableIRQ(UART0_IRQn);

    while (1)
    {

    }
}
```

```
void UART0_IRQHandler(void)
{
    printf("Enter the int\n");
    /* Read data until the receive FIFO is empty */
    while (UART_GetRxFIFOLevel(UART0))
    {
        u8Data = UART_ReadByte(UART0) - '0';
        printf("Rece data(0~10): %d\n", u8Data);
    }

    UART_ClearInt(UART0, UART_INT_ALL);
}
```