SPC1068 UART 单元使用指南

概述

SPC1068 内建一个 UART 单元，有以下特点：

- 最高支持 3.6Mbps
- 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）

注意： 本文档主要以 SPC1068 为例进行介绍。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
C/0	2024-02-27	周佳莉	Released	首次发布。

术语或缩写

术语或缩写	描述
UART	Universal Asynchronous Receiver/Transmitter，通用异步接收器/发送器

1 UART 概述

在 UART 的驱动库中，已经有下列驱动函数可供调动，可大大方便用户使用和理解。

表 1-1：UART API 列表

API Name	Description
UART_Enable(UARTx)	使能 UARTx
UART_Disable(UARTx)	禁用 UARTx
UART_EnableInt(UARTx)	使能 UARTx 中断
UART_DisableInt(UARTx)	禁用 UARTx 中断
UART_EnableRxDataAvailableInt(UARTx)	使能 UARTx 的接收数据中断： 普通模式：UARTx 中是否有接收数据 FIFO 模式：UARTx RX FIFO 中的数据是否超过阈值
UART_DisableRxDataAvailableInt(UARTx)	禁止 UARTx 的接收数据中断
UART_EnableTxDataRequestInt(UARTx)	使能 UARTx 的 FIFO 发送数据请求中断
UART_DisableTxDataRequestInt(UARTx)	禁止 UARTx 的 FIFO 发送数据请求中断
UART_EnableRxErrorInt(UARTx)	使能 UARTx 的接收错误中断
UART_DisableRxErrorInt(UARTx)	禁止 UARTx 的接收错误中断
UART_EnableRxTimeoutInt(UARTx)	使能 UARTx 的接收超时中断
UART_DisableRxTimeoutInt(UARTx)	禁止 UARTx 的接收超时中断
UART_GetGlobalIntStatus(UARTx)	获取 UARTx 的中断状态
UART_GetTxDataRequestIntStatus(UARTx)	获取 UARTx 的 FIFO 发送数据请求状态
UART_GetRxDataAvailableIntStatus(UARTx)	普通模式：UARTx 中是否有接收数据 FIFO 模式：UARTx RX FIFO 中的数据是否超过阈值
UART_GetRxErrorIntStatus(UARTx)	获取 UARTx 的接收错误中断状态
UART_GetRxTimeoutIntStatus(UARTx)	获取 UARTx 的接收超时中断状态
UART_DisableFifo(UARTx)	禁止 UARTx FIFO 功能
UART_ResetRxFifo(UARTx)	清空 UART RX FIFO
UART_ResetTxFifo(UARTx)	清空 UART TX FIFO
UART_GetRxFifoLevel(UARTx)	获取 UARTx RX FIFO 中的数据量
UART_EnableSetBreak(UARTx)	在 UARTx 的发送管脚强制发送低电平
UART_DisableSetBreak(UARTx)	解除 UARTx 发送管脚强制低电平
UART_EnableStickParity(UARTx)	强制 Parity 位和 UARTLCR .EPS 相反
UART_DisableStickParity(UARTx)	解除 Parity 位和 UARTLCR .EPS 相反，Parity 根据数据变化
UART_EnableLoopBackMode(UARTx)	使能 UARTx 的 LoopBack 模式
UART_DisableLoopBackMode(UARTx)	禁用 UARTx 的 LoopBack 模式
UART_IsFifoError(UARTx)	UARTx 是否出现 FIFO 错误
UART_IsTxFifoEmpty(UARTx)	UARTx TX FIFO 是否为空
UART_IsTxDataRequest(UARTx)	UARTx 是否出现发送请求

UART_IsBreakReceived(UARTx)	UARTx 是否收到 Break 信号
UART_IsFrameError(UARTx)	UARTx 是否出现帧错误
UART_IsParityError(UARTx)	UARTx 是否发现 Parity 错误
UART_IsRxDataLost(UARTx)	UARTx 是否出现数据丢失错误
UART_IsRxDataReady(UARTx)	普通模式：UARTx 是否接受到数据 FIFO 模式：UARTx RX FIFO 中的数据是否超过阈值
void UART_Init(UART_REGS* UARTx, uint32_t u32Baudrate)	根据波特率 u32Baudrate 初始化 UART
void UART_SetFifoTriggerLevel(UART_REGS* UARTx, UART_RxTriggerLevelEnum eRxTrigLvl, UART_TxTriggerLevelEnum eTxTrigLvl)	配置 UARTx 的 FIFO 触发阈值
UART_WriteByte(UARTx, u8Data)	UARTx 发送一个字节数据
void UART_Write(UART_REGS* UARTx, uint8_t *pu8Buffer, uint32_t u32Offset, uint32_t u32Count)	通过 UARTx，把 pu8Buffer 中偏移 u32Offset 的 u32Count 个字节的数据发出
UART_ReadByte(UARTx)	通过 UARTx 读取一个字节数据
void UART_Read(UART_REGS* UARTx, uint8_t *pu8Buffer, uint32_t u32Offset, uint32_t u32Count)	通过 UARTx 读取 u32Count 个数据，存储至从 u32Offset 开始的 pu8Buffer 中

2 UART 实例

2.1 UART 收发实例

在这个示例中，会展示 UART 最基本的收发功能，以及最基本的接收中断的使用方法。

1. 首先定义一个 LED 以及要用到的全局变量

Example Code

```
#define LED GPIO_19
uint8_t u8TempData = 0x55;
uint8_t u8TempData2 = 0x0;
```

2. 在 main()中，进行 UART 的基本配置

Example Code

```
/* Configure LED driver pin */
GPIO_SetPinAsGPIO(LED);
GPIO_SetPinDir(LED, GPIO_OUTPUT);

/* Configure GPIO pin as UART function */
GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
/* Enable UART Clock */
CLOCK_EnableModule(UART_MODULE);
/* UART Init */
UART_Init(UART, 38400);

/* Enable Internal loopback for ease of test */
UART_EnableLoopBackMode(UART);

/* Enable RX data interrupt */
UART_EnableRxDataAvailableInt(UART);
/* Enable UART interrupt (main switch of UART interrupt) */
UART_EnableInt(UART);
/* Enable UART interrupt in CPU side */
NVIC_EnableIRQ(UART_IRQn);

/* Send a byte to trigger the loop */
UART_WriteByte(UART, u8TempData);

while(1)
{}
```

3. 在中断函数中，进行两次接收数据比较，如果符合条件，则 LED 闪烁

Example Code

```
void UART_IRQHandler(void)
{
  u8TempData2 = u8TempData;

  /* Read UART reception buffer */
  u8TempData = UART_ReadByte(UART);

  /* If data correct */
  if((u8TempData2+1) == u8TempData)
  {
    /* Toggle LED  */
    GPIO_TogglePin(LED);
  }

  Delay_ms(100);

  /* Send data again */
  UART_WriteByte(UART, u8TempData +1);
}
```

观察现象，如果看到 LED 闪烁，则可以证明 UART 的发送和接收正常运行。

2.2 UART FIFO 使用实例

在速度较高、数据量较大的 UART 通信中，接受/发送 FIFO 是一个非常有用的特性，在 SPC1068 的 UART 中，带有 64Byte 的发送/接受 FIFO。经过一些系统实测，这个特性可以让 SPC1068 的 UART 工作在 3.6Mbps 持续发送接收的状态。

同样，在 SCP1068 提供的 SDK 中，已经包含了很多针对 UART FIFO 功能优化过的 API 函数。用户如果需要使用 UART 的 FIFO 功能，只需要简单的调用这些 API 即可，无需详细地理解 UART FIFO 的实现细节。

这里要特别注意的是 FIFO 寄存器的设置，由于这个寄存器是一个 Write-Only 寄存器，即对它进行读取操作不会有返回值，所以一般使用的位与、位或操作是无法作用到这个寄存器上的。这里限定使用 SDK 中的函数进行设置：

```
void UART_SetFifoTriggerLevel(UART_REGS* UARTx,
                              UART_RxTriggerLevelEnum eRxTrigLvl,
                              UART_TxTriggerLevelEnum eTxTrigLvl)
```

用户需要设定 eRxTrigLvl（接收 FIFO 触发阈值）和 eTxTrigLvl（发送 FIFO 触发阈值），同时，该函数会默认打开 FIFO 功能。

1. 例 1：UART 接收 FIFO 使用例程

在 mian() 中，加入 UART 带 FIFO 配置的初始化

Example Code

```
{
  /* Configure LED driver pin */
  GPIO_SetPinAsGPIO(LED);
  GPIO_SetPinDir(LED, GPIO_OUTPUT);

  /* Configure GPIO pin as UART function */
  GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
  /* Enable UART Clock */
  CLOCK_EnableModule(UART_MODULE);
    /* UART Init */
  UART_Init(UART, 38400);

  /* RX FIFO trigger level: 16 */
  /* TX FIFO trigger level:  0 */
  UART_SetFifoTriggerLevel(UART, UART_RXFIFO_16BYTE_OR_MORE,
                                 UART_TXFIFO_EMPTY);

  /* Enable Loopback for ease of test */
  UART_EnableLoopBackMode(UART);

  /* Enable normal RX interrupt */
  UART_EnableRxDataAvailableInt(UART);

  /* Enable UART interrupt (main switch of UART interrupt) */
  UART_EnableInt(UART);
```

```
  /* Enable UART ISR in CPU side */
  NVIC_EnableIRQ(UART_IRQn);


  while(1)
  {
    Delay_ms(50);

    UART_WriteByte(UART, 0x55);
  }
}



void UART_IRQHandler(void)
{
  /* RX data more than 16 */
  if(UART_GetRxDataAvailableIntStatus(UART))
  {
    /* Check RX FIFO Level */
    if(UART_GetRxFifoLevel(UART) >= 16)
    {
      /* Read data from FIFO  */
      UART_Read(UART, UARTRead, 0, 16);

      /* Toggle LED */
      GPIO_TogglePin(LED);
    }
  }
}
```

2. 例 2：UART 发送 FIFO 使用例程

在下面的例子中，展示了如何使用 UART 的发送 FIFO，这个例子是基于之前的接收中断的例子上增加了发送中断的部分。在发送 FIFO 的数据被发送完之后，可以向 CPU 发送中断，进入中断后重新进行数据的装载。

Example Code

```
{
  /* Configure LED driver pin */
  GPIO_SetPinAsGPIO(LED);
  GPIO_SetPinDir(LED, GPIO_OUTPUT);

  /* Configure GPIO pin as UART function */
  GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
  /* Enable UART Clock */
  CLOCK_EnableModule(UART_MODULE);
  /* UART Init */
  UART_Init(UART, 38400);
```

```
  /* RX FIFO trigger level: 16 */
  /* TX FIFO trigger level:  0 */
  UART_SetFifoTriggerLevel(UART, UART_RXFIFO_16BYTE_OR_MORE,
                               UART_TXFIFO_EMPTY);

  /* Enable Loopback for ease of test */
  UART_EnableLoopBackMode(UART);

  /* Enable TX FIFO interrupt */
  UART_EnableTxDataRequestInt(UART);
  /* Enable RX interrupt */
  UART_EnableRxDataAvailableInt(UART);
  /* Enable UART interrupt (main switch of UART interrupt) */
  UART_EnableInt(UART);

  /* Enable UART interrupt in CPU side */
  NVIC_EnableIRQ(UART_IRQn);

  while(1)
  {
  }
}


void UART_IRQHandler(void)
{
  uint32_t i;

  switch(UART->UARTIIR_UARTFCR.UARTIIR.bit.IID)
  {
    case UARTIIR_BIT_IID_TX_FIFO_REQUEST_DATA_INT:
      /* Fill Tx FIFO */
      for(i = 0; i <= 5; i++)
      {
        UART_WriteByte(UART, 0x55);
      }

      Delay_ms(20);

      break;

    case UARTIIR_BIT_IID_RX_DATA_READY_INT:
      /* Check Rx FIFO level */
      if(UART_GetRxFifoLevel(UART) >= 16)
      {
        /* Read data from RX FIFO */
        UART_Read(UART, UARTRead, 0, 16);

        /* Toggle LED */
        GPIO_TogglePin(LED);
```

```
        Delay_ms(100);
      }

      break;

    default:
      break;
  }
}
```

2.3 UART 自动波特率检测实例

SPC1068 的 UART 具有自动波特率计算的能力，它内建一个计数器，会计算在 RX 上接收到的从第一个下降沿到第一个上升沿之间时钟差距。根据 UART 的协议，如果接收到的第一位是“1”的话，这个差距应该是 1 个 UART 时钟。 根据这个假设，如果对端发送 0bxxxxxx1 的数据，则可以计算出正确的波特率。

下面的例子实现了 UART 的自动波特率计算，不过在具体流程上作了一定简化，即对端第一个数据必须发送 0x1。

UART 的初始化代码如下：

Example Code

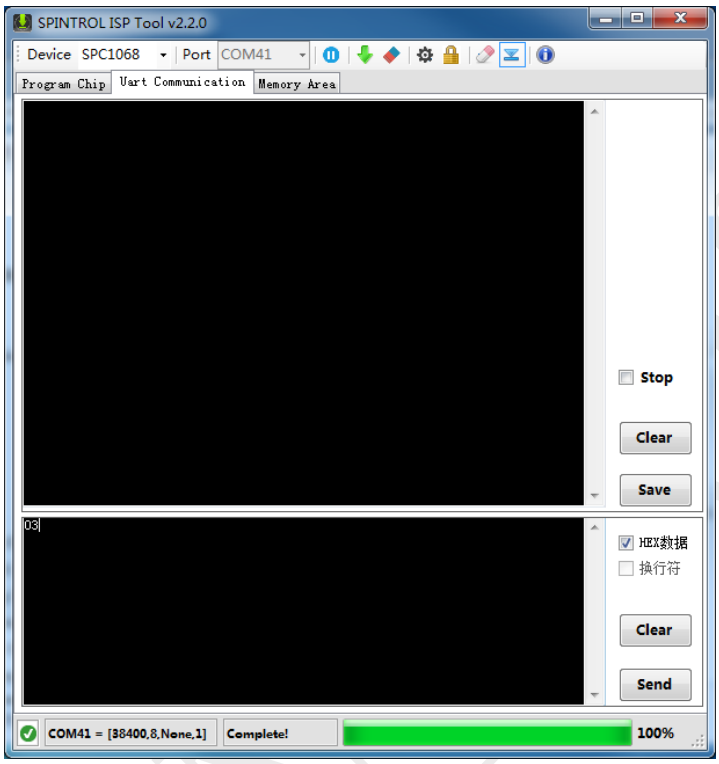
```
{
  /* Configure LED driver pin */
  GPIO_SetPinAsGPIO(LED);
  GPIO_SetPinDir(LED, GPIO_OUTPUT);

  /* Configure GPIO pin as UART function */
  GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
  /* Enable UART Clock */
  CLOCK_EnableModule(UART_MODULE);

Autobaud:
  UART->UARTABR.bit.ABE    = UARTABR_BIT_ABE_ENABLE;              /*
Enable Autobaud detect */
  UART->UARTABR.bit.ABUP   = UARTABR_BIT_ABUP_UART_PROG_DIVISOR;  /*
UART unit programs Divisor Latch registers */
  UART->UARTABR.bit.ABLIE  = UARTABR_BIT_ABLIE_ENABLE;            /*
Enable Autobuad lock interrupt */
  UART->UARTABR.bit.ABT    = UARTABR_BIT_ABT_FORMULA;             /*
Formula used to calculate baud rates */

  UART->UARTLCR.bit.WLS    = UARTLCR_BIT_WLS_8_DATA_BIT;          /*
8 databits */
  UART->UARTLCR.bit.STB    = UARTLCR_BIT_STB_1_STOP_BIT;          /*
1 stop bit */
  UART->UARTLCR.bit.EPS    = UARTLCR_BIT_PEN_NO_PARITY;           /*
No parity  */

  /* Enable UART unit */
  UART_Enable(UART);
  /* Enable UART interrupt */
  UART_EnableInt(UART);
  /* Enable UART interrupt in CPU side */
  NVIC_EnableIRQ(UART_IRQn);

  /* Wait auto-baud lock */
  while(autoBaud == 0);

  /* Disable auto-baud detect */
  UART->UARTABR.bit.ABE = 0;
```

```
  /* Disable auto-baud lock interrupt */
  UART->UARTABR.bit.ABLIE = 0;

  /* Wait data received */
  while(!UART_IsRxDataReady(UART));

  /* Read data and validate it */
  while(UART_ReadByte(UART) != 0x01)
  {
    UART_Disable(UART);
    goto Autobaud;
  }

  printf("Auto-baud success!\n");

  /* Enable receive data available interrupt */
  UART_EnableRxDataAvailableInt(UART);


  while(1)
  {
    Delay_ms(1000);
    GPIO_TogglePin(LED);
  }
}

void UART_IRQHandler(void)
{
  /* Auto-baud locked */
  if(UART->UARTIIR_UARTFCR.UARTIIR.bit.ABL == 1)
  {
    autoBaud = 1;
  }

  /* Receive data available */
  if(UART_GetRxDataAvailableIntStatus(UART))
  {
    /* Write data back */
    UART_WriteByte(UART, UART_ReadByte(UART));
  }
}
```

具体的观察方法：启动 ISP 工具后，在任意波特率下，向 SPC1068 的 UART 发送一个字节数据：

1. 如果数据不等于 0x1，则没有任何反应

图 2-1：自动波特率检测失败



2. 如果数据等于 0x1，则可以收到 SPC1068 返回的正确数据，可以判定 Autobaud 工作正常。

图 2-2：自动波特率检测成功