32 位微控制器

HC32F4A0 系列的通用同步异步收发器的 UART 不定长数据接收

本产品支持芯片系列如下

F 系列	**HC32F4A0**

目　录

1 摘要

在通信中，用户经常会处理不定长的数据包。发送数据时，只需要根据指定长度发送数据包，处理相对简单，而接收数据时则需要检查数据包的结束。

本篇应用笔记主要介绍 HC32F4A0 系列通用同步异步收发器（Universal Synchronous Asynchronous Receiver Transmitter，USART）的 UART 模式，结合定时器 Timer0，实现接收不定长数据。

2 UART 简介

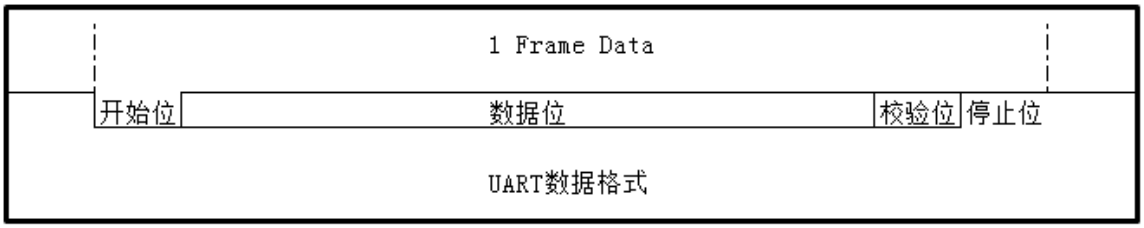
HC32F4A0 系列 MCU 搭载通用串行收发器模块（USART）10 个单元。全通道支持通用异步串行通信接口 （Universal Asynchronous Receiver Transmitter，UART）。UART（USART_1，USART_2，USART_6，USART_7）和 Timer0 模块配合支持 UART 接收 TIMEOUT 功能。

2.1 UART 主要特性

- 数据长度可编程：8 位/9 位
- 校验功能可配置：奇校验/偶校验/无校验
- 停止位可配置：1 位/2 位
- 时钟源可选：内部时钟源（内部波特率生成器生成的时钟）/外部时钟源（USARTn_CK 管脚输入的时钟）
- 接收错误：校验错误，帧错误，上溢错误
- 支持多个处理器间通信
- 内置数字滤波器
- 支持接收数据超时功能
- 单元 1 支持停止模式唤醒功能

2.2 UART 数据格式

UART 模式一帧数据是由起始位、数据位、校验位和停止位组成。



UART数据格式

起始位	数据位	奇偶校验位	停止位
1 位	8，9 位	1 位（可选）	1,2 位

3 UART 接收超时功能

UART 接收数据停止位被检测时 TIMEOUT 计数器启动，经过设定的 TIMEOUT 时间（设定单位为接收位数）后未检测到下一帧接收数据时，发生 TIMEOUT，如果此时 CR1.RE=1，则 TIMEOUT 状态位 USARTn_SR.RTOF 置位，如果此时 USARTn_CR1.RE=0，则等待 USARTn_CR1.RE=1 后 TIMEOUT 状态位 USARTn_SR.RTOF 置位。

TIMEOUT 计数器采用 Timer0 模块的计数器，具体对应关系如下：

USART1：Timer0 Unit1 A 通道

USART2：Timer0 Unit1 B 通道

USART5：Timer0 Unit2 A 通道

USART6：Timer0 Unit2 B 通道

TIMEOUT 功能 Timer0 比较计数器值设定：

Timer0 为 16 位计数器，计数时钟最大可以选择 1024 分频，TMR0_CMPA<B>R 值设定计算公式如下：

$$CMPA<B>R = \frac{RTB}{2^{CKDIVA<B>}} - 4\ （计数时钟不分频）$$

$$CMPA<B>R = \frac{RTB}{2^{CKDIVA<B>}} - 2\ （计数时钟 2 分频）$$

$$CMPA<B>R = \frac{RTB}{2^{CKDIVA<B>}} - 1\ （计数时钟 4 分频及以上）$$

CMPA<B>R: TMR0_CMPA<B>R 寄存器值

RTB: Receive Timeout Bits

CKDIRA<B>: TMR0.BCONR.CKDIVA<B>位寄存器值

4 UART 不定长数据接收

4.1 UART 接收不为空和超时标志设置时序

UART 接收数据停止位被检测时超时计数器启动，经过设定的超时接收位数后未检测到下一帧接收数据时，将置位接收超时标志 RTOF。

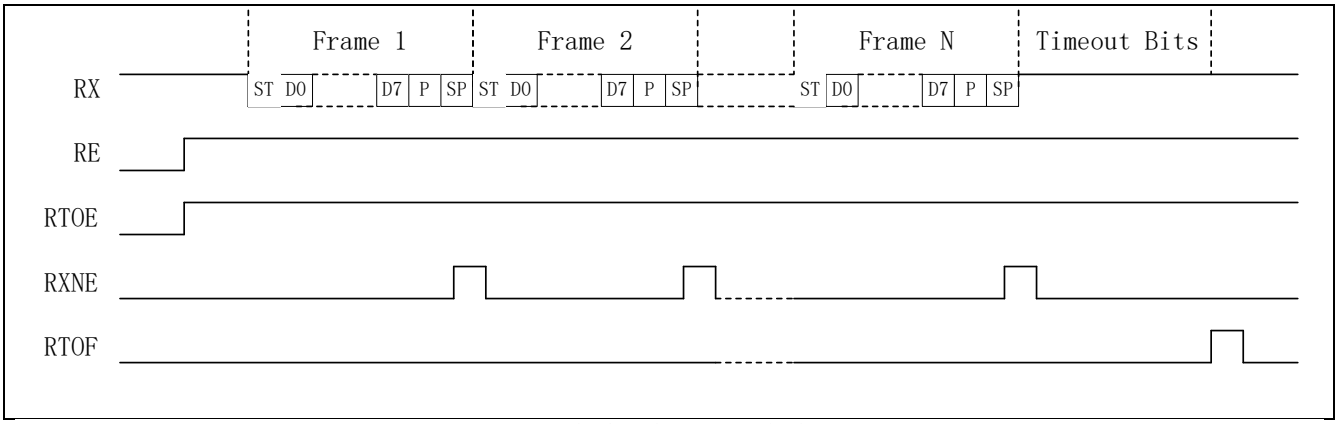


图 4-1 UART 接收不为空和超时标志设置时序

4.2 工作流程

4.2.1 定时器 Timer0 配置

样例代码中 Timer0 操作流程如下图所示：

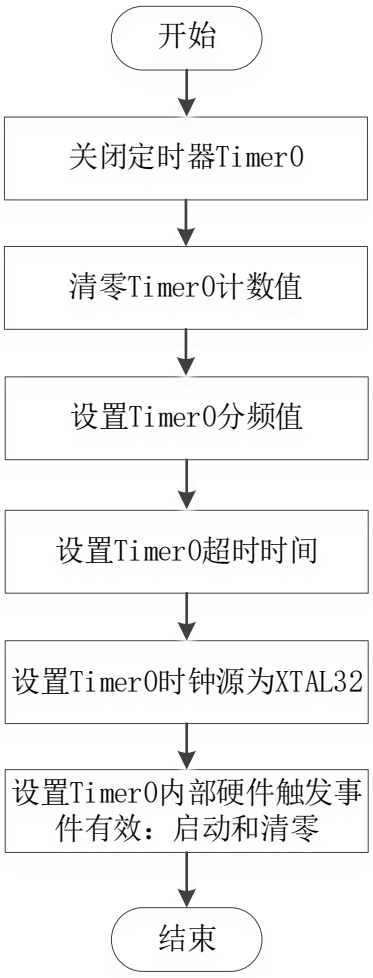


图 4-2 定时器 Timer0 配置流程

4.2.2 UART 接收超时中断处理

样例代码中接收超时中断处理流程如下图所示：

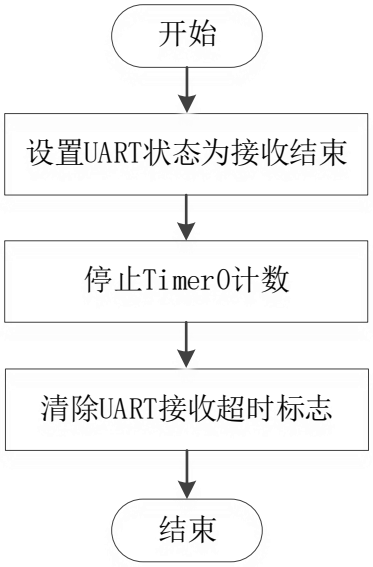


图 4-3 UART 接收超时中断处理流程

4.2.3 UART 接收中断处理

样例代码中接收中断处理流程如下图所示：

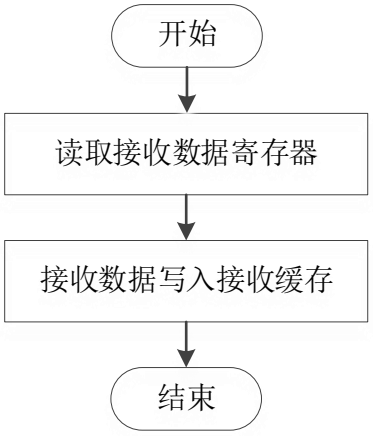


图 4-4 UART 接收中断处理流程

4.2.4 主程序处理

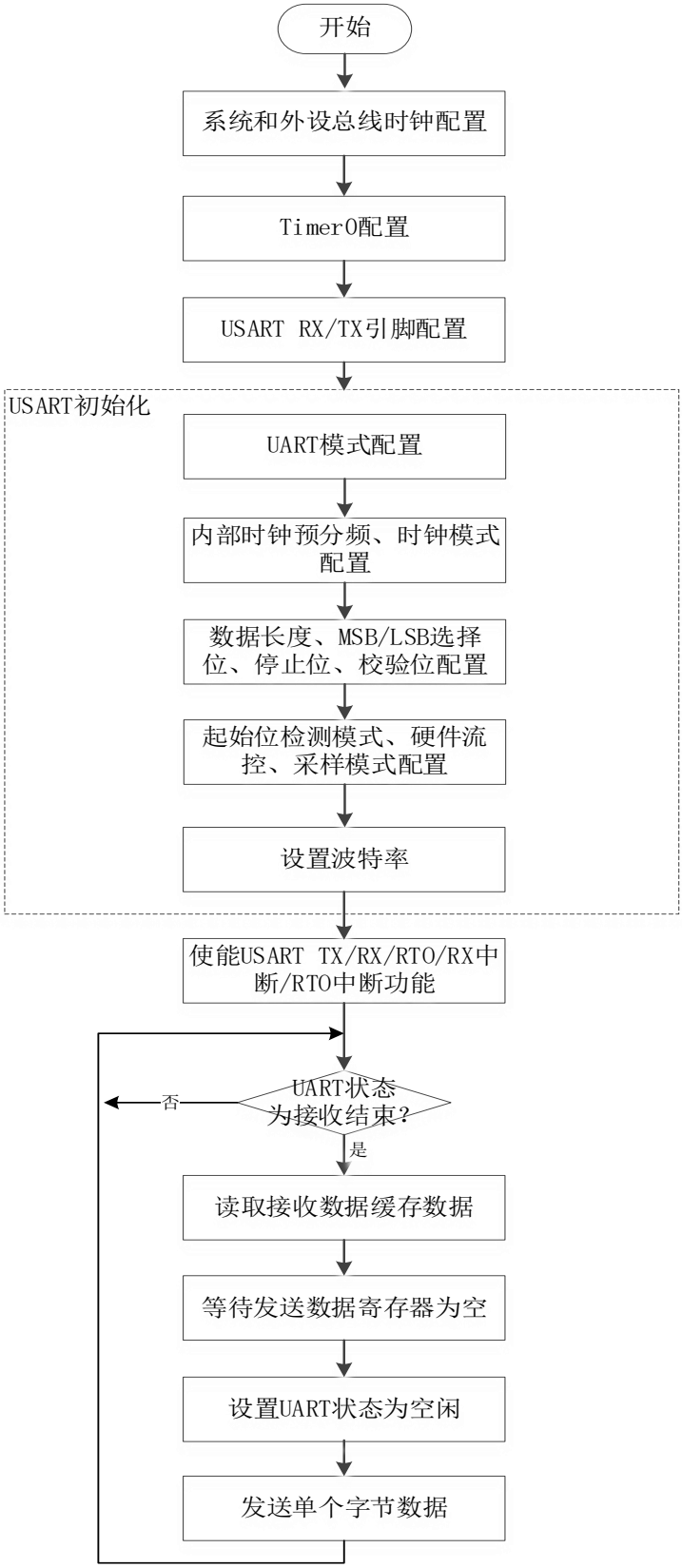


图 4-5 UART 接收中断处理流程

4.3 程序示例

UART 不定长度数据接收操作通过设备驱动库（Device Driver Library, DDL）的样例 uart_rx_none_fixed_length 展示。下面主要介绍比较关键的代码。

1) 定时器 Timer0 配置。

```
static void TMR0_Config(M4_TMR0_TypeDef *TMR0x,
                        uint8_t u8Channel,
                        uint16_t u16TimeoutBits)
{
    uint32_t u32CmpVal;
    stc_tmr0_init_t stcTmr0Init;
    uint32_t u32Fcg2Periph = (M4_TMR0_1 == TMR0x) ? PWC_FCG2_TMR0_1 :
PWC_FCG2_TMR0_2;

    PWC_Fcg2PeriphClockCmd(u32Fcg2Periph, Enable);

    /* Clear CNTAR register for channel A */
    TMR0_SetCntVal(TMR0x, u8Channel, 0U);

    /* Timer0 basetimer function initialize */
    (void)TMR0_StructInit(&stcTmr0Init);
    stcTmr0Init.u32ClockDivision = TMR0_CLK_DIV8;
    stcTmr0Init.u32ClockSource = TMR0_CLK_SRC_XTAL32;
    stcTmr0Init.u32HwTrigFunc = (TMR0_BT_HWTRG_FUNC_START |
TMR0_BT_HWTRG_FUNC_CLEAR);
    if (TMR0_CLK_DIV1 == stcTmr0Init.u32ClockDivision)
    {
        u32CmpVal = (u16TimeoutBits - 4UL);
    }
    else if (TMR0_CLK_DIV2 == stcTmr0Init.u32ClockDivision)
    {
        u32CmpVal = (u16TimeoutBits/2UL - 2UL);
    }
    else
    {
        u32CmpVal = (u16TimeoutBits / (1UL << (stcTmr0Init.u32ClockDivision >>
TMR0_BCONR_CKDIVA_POS)) - 1UL);
    }
    DDL_ASSERT(u32CmpVal <= 0xFFFFUL);
    stcTmr0Init.u16CmpValue =  (uint16_t)(u32CmpVal);
    (void)TMR0_Init(TMR0x, u8Channel, &stcTmr0Init);

    /* Clear compare flag */
    TMR0_ClearStatus(TMR0x, u8Channel);
}
```

2) UART 接收中断处理。

```
static void USART_Rx_IrqCallback(void)
{
    uint16_t u16Data = USART_RecData(USART_UNIT);

    (void)RingBufWrite(&m_stcRingBuf, (uint8_t)u16Data);
}
```

3) UART 接收超时中断处理。

```
static void USART_RxTimeout_IrqCallback(void)
{
    m_enUartState = UartRxEnd;

    TMR0_Cmd(TMR0_UNIT, TMR0_CH, Disable);
    USART_ClearStatus(USART_UNIT, USART_CLEAR_FLAG_RTOF);
}
```

4) UART 缓存写操作。

```
static en_result_t RingBufWrite(stc_ring_buffer_t *pstcBuffer, uint8_t u8Data)
{
    en_result_t enRet = Ok;

    if (pstcBuffer->u16UsedSize >= pstcBuffer->u16Capacity)
    {
        enRet = ErrorBufferFull;
    }
    else
    {
        pstcBuffer->au8Buf[pstcBuffer->u16InIdx++] = u8Data;
        pstcBuffer->u16InIdx %= pstcBuffer->u16Capacity;
        pstcBuffer->u16UsedSize++;
    }

    return enRet;
}
```

5) UART 缓存读操作。

```
static en_result_t RingBufRead(stc_ring_buffer_t *pstcBuffer, uint8_t *pu8Data)
{
    en_result_t enRet = Ok;

    if (pstcBuffer->u16UsedSize == 0U)
    {
        enRet = ErrorBufferEmpty;
    }
    else
    {
        *pu8Data = pstcBuffer->au8Buf[pstcBuffer->u16OutIdx++];
```

```
        pstcBuffer->u16OutIdx %= pstcBuffer->u16Capacity;
        pstcBuffer->u16UsedSize--;
    }

    return enRet;
}
```

6) 样例主程序。

```
int32_t main(void)
{
    uint16_t u16Data;
    stc_irq_signin_config_t stcIrqSigninCfg;
    const stc_usart_uart_init_t stcUartInit = {
        .u32Baudrate = USART_BAUDRATE,
        .u32BitDirection = USART_LSB,
        .u32StopBit = USART_STOPBIT_1BIT,
        .u32Parity = USART_PARITY_NONE,
        .u32DataWidth = USART_DATA_LENGTH_8BIT,
        .u32ClkMode = USART_INTERNCLK_OUTPUT,
        .u32PclkDiv = USART_PCLK_DIV64,
        .u32OversamplingBits = USART_OVERSAMPLING_8BIT,
        .u32NoiseFilterState = USART_NOISE_FILTER_DISABLE,
        .u32SbDetectPolarity = USART_SB_DETECT_FALLING,
    };

    /* MCU Peripheral registers write unprotected */
    Peripheral_WE();

    /* Initialize system clock. */
    BSP_CLK_Init();

    /* Initialize TMR0. */
    TMR0_Config(TMR0_UNIT, TMR0_CH, 5000);

    /* Configure USART RX/TX pin. */
    GPIO_SetFunc(USART_RX_PORT, USART_RX_PIN, USART_RX_GPIO_FUNC,
PIN_SUBFUNC_DISABLE);
    GPIO_SetFunc(USART_TX_PORT, USART_TX_PIN, USART_TX_GPIO_FUNC,
PIN_SUBFUNC_DISABLE);

    /* MCU Peripheral registers write protected */
    Peripheral_WP();

    /* Enable peripheral clock */
    PWC_Fcg3PeriphClockCmd(USART_FUNCTION_CLK_GATE, Enable);

    /* Initialize UART function. */
    if (Ok != USART_UartInit(USART_UNIT, &stcUartInit))
    {
        for (;;)
        {
        }
    }

    /* Register RX IRQ handler && configure NVIC. */
```

```
    stcIrqSigninCfg.enIRQn = USART_UNIT_RX_INT_IRQn;
    stcIrqSigninCfg.enIntSrc = USART_UNIT_RX_INT_SRC;
    stcIrqSigninCfg.pfnCallback = &USART_Rx_IrqCallback;
    (void)INTC_IrqSignIn(&stcIrqSigninCfg);
    NVIC_ClearPendingIRQ(stcIrqSigninCfg.enIRQn);
    NVIC_SetPriority(stcIrqSigninCfg.enIRQn, DDL_IRQ_PRIORITY_00);
    NVIC_EnableIRQ(stcIrqSigninCfg.enIRQn);

    /* Register RX error IRQ handler && configure NVIC. */
    stcIrqSigninCfg.enIRQn = USART_RXERR_INT_IRQn;
    stcIrqSigninCfg.enIntSrc = USART_RXERR_INT_SRC;
    stcIrqSigninCfg.pfnCallback = &USART_RxErr_IrqCallback;
    (void)INTC_IrqSignIn(&stcIrqSigninCfg);
    NVIC_ClearPendingIRQ(stcIrqSigninCfg.enIRQn);
    NVIC_SetPriority(stcIrqSigninCfg.enIRQn, DDL_IRQ_PRIORITY_01);
    NVIC_EnableIRQ(stcIrqSigninCfg.enIRQn);

    /* Register RX timeout IRQ handler && configure NVIC. */
    stcIrqSigninCfg.enIRQn = USART_RXTO_INT_IRQn;
    stcIrqSigninCfg.enIntSrc = USART_RXTO_INT_SRC;
    stcIrqSigninCfg.pfnCallback = &USART_RxTimeout_IrqCallback;
    (void)INTC_IrqSignIn(&stcIrqSigninCfg);
    NVIC_ClearPendingIRQ(stcIrqSigninCfg.enIRQn);
    NVIC_SetPriority(stcIrqSigninCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
    NVIC_EnableIRQ(stcIrqSigninCfg.enIRQn);

    /* Enable TX && RX && RX interrupt function */
    USART_FuncCmd(USART_UNIT, (USART_RX | USART_INT_RX | USART_TX | \
                        USART_RTO | USART_INT_RTO), Enable);

    for (;;)
    {
      if (m_enUartState == UartRxEnd)
      {
        while (m_stcRingBuf.u16UsedSize > 0U)
        {
          if (Ok == RingBufRead(&m_stcRingBuf, (uint8_t *)&u16Data))
          {
            while (Reset == USART_GetStatus(USART_UNIT, USART_FLAG_TXE)) /* Wait Tx data
register empty */
            {
            }

            USART_SendData(USART_UNIT, u16Data);
          }
        }

        m_enUartState = UartIdle;
      }
    }
  }
```

5 总结

以上章节简要介绍 HC32F4A0 系列的 UART 接收不定长度数据的使用方法。样例代码演示了如何根据结合 Timer0，使用 UART 超时功能。只有通道 USART1、USART2、USART5 和 USART6 支持接收超时功能，在开发中用户可以根据自己的实际需要选择 USART 通道使用该功能。

6 版本信息 & 联系方式

日期	版本	修改记录
2021/7/27	Rev1.0	初版发布

如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email：mcu@hdsc.com.cn

网址：http://www.hdsc.com.cn/mcu.htm

通信地址：上海市浦东新区中科路 1867 号 A 座 10 层

邮编：201203