

32 位 微控制器

HC32F460 系列的通用同步异步收发器 USART

适用对象

系列	产品型号
HC32F460	HC32F460JEUA HC32F460JETA HC32F460KEUA HC32F460KETA HC32F460PETB

目 录

1	摘要	3
2	USART 简介.....	3
3	HC32F460 系列的 USART	4
	3.1 简介.....	4
	3.2 说明.....	4
	3.2.1 UART 工作模式.....	4
	3.2.2 时钟同步模式	6
	3.2.3 智能卡模式.....	7
	3.2.4 波特率计算.....	8
	3.2.5 使用注意事项	10
	3.2.6 寄存器介绍.....	11
	3.2.7 工作流程介绍	12
4	样例代码	13
	4.1 代码介绍	13
	4.2 代码运行	15
5	总结	16
6	版本信息 & 联系方式	17

1 摘要

本篇应用笔记主要介绍 HC32F460 系列芯片的通用同步异步收发器（Universal Synchronous/Asynchronous Receiver/Transmitter ,USART）模块，并通过 UART 模式简要说明如何使用 USART 进行通信。

2 USART 简介

USART 是一个通用同步/异步串行收发模块，该接口是一个高度灵活的串行通信设备。

USART 主要特性：

- 支持全双工通信
- 收发波特率可编程
- 数据字长度可编程
- 停止位可配置
- LSB/MSB 可选
- 奇偶校验控制
- 传输事件检测：接收缓冲区已满、发送缓冲区为空、传输结束标志
- 支持错误检测：校验错误，帧错误，上溢错误
- 同步发送的发送器时钟输出

3 HC32F460 系列的 USART

3.1 简介

HC32F460 系列的 MCU 搭载通用串行收发器模块（USART）4 个单元,通用串行收发器模块（USART）支持通用异步串行通信接口（UART）,时钟同步通信接口，智能卡接口（ISO/IEC7816-3），能够灵活地与外部设备进行全双工数据交换。支持调制解调器操作（CTS/RTS 操作）,多处理器操作。和 TIMER0 模块配合支持 UART 接收 TIMEOUT 功能。

3.2 说明

3.2.1 UART 工作模式

UART（Universal Asynchronous Receiver/Transmitter）是一种通用异步收发传输器，其使用串行的方式进行数据交换，实现全双工通信。

1) 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。

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

2) UART 基本参数

波特率、起始位、数据位、停止位、检验位和信号电平。

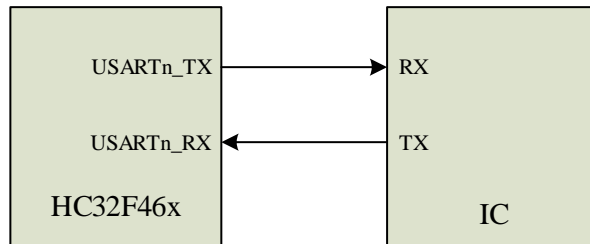
3) HC32F460 UART 工作模式主要特性：

- 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(CKn 管脚输入的时钟)
- 收信错误：校验错误，帧错误，上溢错误
- 调制解调器操作 (CTS/RTS)
- 多个处理器间通信

- 内置数字滤波器可以消除接收数据线上的噪音
- 支持接收数据 TIMEOUT 功能
- 单元 1 支持停止模式唤醒功能

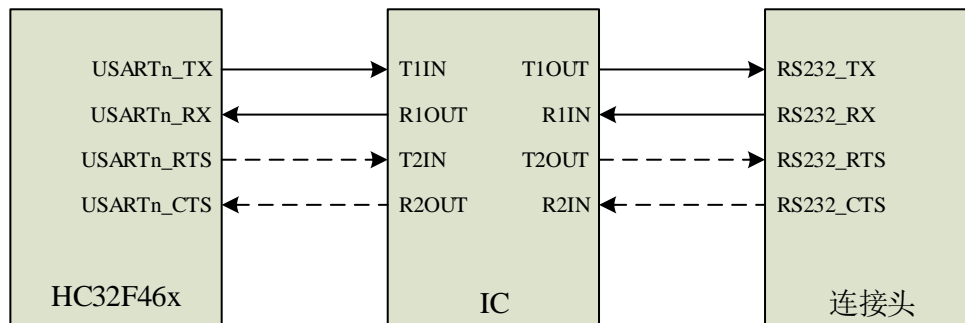
4) UART 模式接口连接示意图

- TTL 电平接口

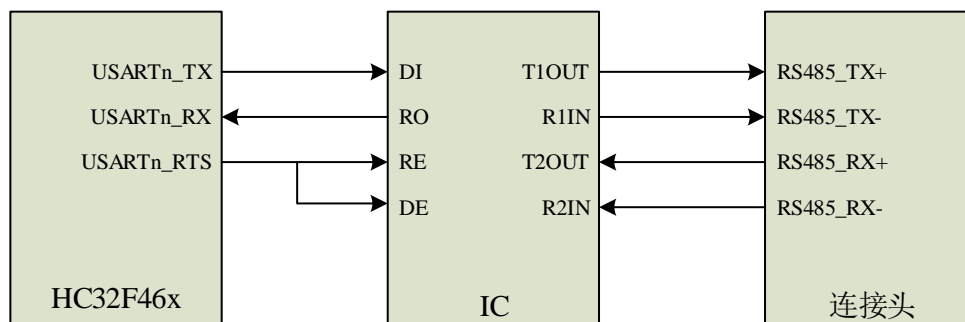


- RS232 接口

硬件流控为可选配置选项，根据实际应用连接。



- RS485 接口



3.2.2 时钟同步模式

1) 时钟同步模式主要特性:

- 数据长度: 8 位
- 接收错误: 上溢错误
- 调制解调器操作(CTS/RTS)
- 时钟源: 内部时钟源(内部波特率生成器生成的时钟)/外部时钟源(CKn 管脚输入的时钟)

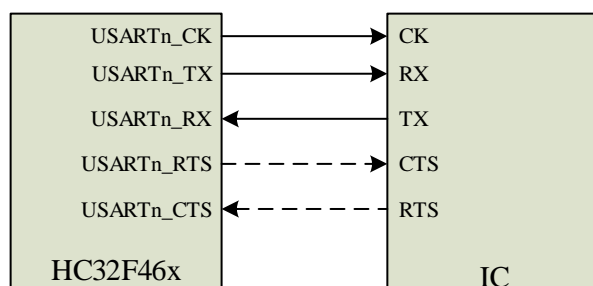
2) 时钟同步模式数据格式

时钟同步模式一帧数据是由 8 位数据位组成, 无起始位、校验位和停止位。

起始位	数据位	奇偶校验位	停止位
无	8 位	无	无

3) 时钟同步模式接口连接示意图

硬件流控为可选配置选项, 根据实际应用连接。

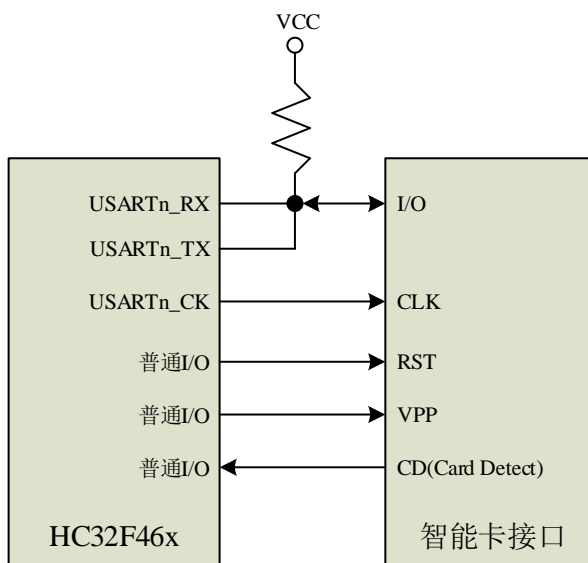


3.2.3 智能卡模式

1) 智能卡模式主要特性：

- 数据长度：8 位
- 检测到校验错误时能自动送出错误信号
- 支持数据重发

2) 智能卡模式接口连接示意图



3.2.4 波特率计算

波特率生成器提供小数波特率模式和整数波特率模式。整数波特率模式误差较大时，可采用小数波特率模式来降低波特率计算误差。

1) 小数波特率无效，波特率计算公式如下

模式	波特率计算公式	误差E(%)计算公式
UART模式 多处理器模式	$B = \frac{C}{8 \times (2 - OVER8) \times (DIV_Integer + 1)}$	$E(\%) = \left\{ \frac{C}{8 \times (2 - OVER8) \times (DIV_Integer + 1) \times B} - 1 \right\} \times 100$
时钟同步模式	$B = \frac{C}{4 \times (DIV_Integer + 1)}$	-
智能卡模式	$B = \frac{C}{2 \times S \times (DIV_Integer + 1)}$	$E(\%) = \left\{ \frac{C}{2 \times S \times (DIV_Integer + 1) \times B} - 1 \right\} \times 100$

2) 小数波特率有效，波特率计算公式如下

模式	波特率计算公式	误差E(%)计算公式
UART模式 多处理器模式	$B = \frac{C \times (128 + DIV_Fraction)}{8 \times (2 - OVER8) \times (DIV_Integer + 1) \times 256}$	$E(\%) = \left\{ \frac{C \times (128 + DIV_Fraction)}{8 \times (2 - OVER8) \times (DIV_Integer + 1) \times 256 \times B} - 1 \right\} \times 100$
时钟同步模式	$B = \frac{C \times (128 + DIV_Fraction)}{4 \times (DIV_Integer + 1) \times 256}$	
智能卡模式	$B = \frac{C \times (128 + DIV_Fraction)}{2 \times S \times (DIV_Integer + 1) \times 256}$	$E(\%) = \left\{ \frac{C \times (128 + DIV_Fraction)}{2 \times S \times (DIV_Integer + 1) \times 256 \times B} - 1 \right\} \times 100$

B: 波特率 单位: Mbps

C: 寄存器 USARTn_PR.PSC[1:0]位设定的时钟 单位: MHz

S: 寄存器 USARTn_CR3.BCN 设定的一位数据传输的基本时钟数

OVER8: 寄存器 USARTn_CR1.OVER8 设定值

DIV_Integer: 寄存器 USARTn_BRR.DIV_Integer 设定值

DIV_Fraction: 寄存器 USARTn_BRR.DIV_Fraction 设定值

- 以 UART 模式为例，假设情况如下：

时钟源为内部时钟源

PCLK = 8000000Hz

USARTn_PR.PSC[1:0]值为 0，则 C=8000000Hz

USARTn_CR1.OVER8 值为 1，则 OVER8=1

设置波特率为 115200，则 B=115200

- 首先将参数 OVER8, C, B 带入 UART 整数波特率模式公式，如下所示，计算得到
 $\text{DIV_Integer} = 7$

$$115200 = \frac{8000000}{8 \times (2 - 1) \times (\text{DIV_Integer} + 1)}$$

- 将参数 OVER8, C, DIV_Integer 带入 UART 整数波特率模式公式，计算实际波特率为 125000，误差为 8.5069%
- 将参数 OVER8, C, B, DIV_Integer，带入 UART 小数波特率模式公式，如下所示，计算得到 $\text{DIV_Fraction} = 126$

$$115200 = \frac{8000000 \times (128 + \text{DIV_Fraction})}{8 \times (2 - 1) \times (7 + 1) \times 256}$$

- 将参数 OVER8, C, DIV_Integer, DIV_Fraction 带入 UART 小数波特率模式公式，计算实际波特率为 114746，误差为 0.394%
- 比较以上计算结果可知，通过 UART 小数点波特率模式修正后，可显著降低波特率误差

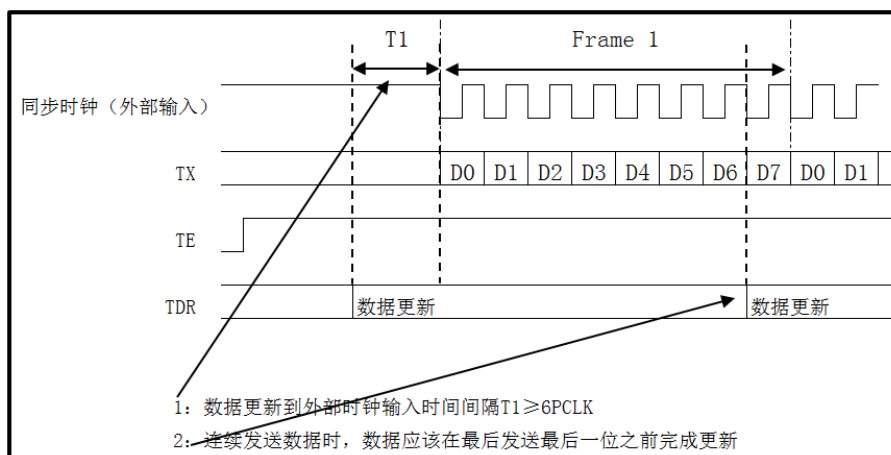
3.2.5 使用注意事项

1) UART 模式注意事项:

- UART 模式发送器发送动作禁止时(USARTn_CR1.TE=0), 则 TX 引脚可以当作普通 IO 使用, 可以设定输出值和方向。如果输出 0, 则会使接收方产生接收方产生帧错误, 从而中断数据传输。如果输出 1, 则使接收方检测不到起始位从而无法开始数据传输。
- UART 模式接收器动作时, 产生帧错误时可以通过检测 RX 线是否为始终为 0 来确认是否为发送方中断传输。发生帧错误的同时校验错误也有可能发生, 此时如果接收数据开始位检测方式为低电平检测, 则在清除错误标志后继续接收全为低电平数据, 接收错误可能再次发生。此时如果接收数据开始位检测方式为下降沿检测, 则可避免此问题。

2) 时钟同步模式注意事项:

- 使用外部输入时钟发送数据时, 在 TDR 更新后等待 6 个 PCLK 以上再输入时钟。输入时钟总周期要大于等于 6PCLK, 高电平和低电平期间必须大于等于 2PCLK。
- 连续发送数据时, 下一帧数据需要在当前帧最后一位发送前完成更新。



3) 其他注意事项:

- 写 USARTn_TDR 应该通过 TI 中断去写, 一次中断只能进行一次写操作。
- 通信过程中, 不应该修改波特率寄存器的值。
- 为了防止发送禁止时 TX 通信线 Hi-Z 状态, 可以采用以下方法:
 - 上拉
 - 发送数据结束时, USARTn_CR1.TE=0 之前, 将 TX 引脚设为普通 IO 输出

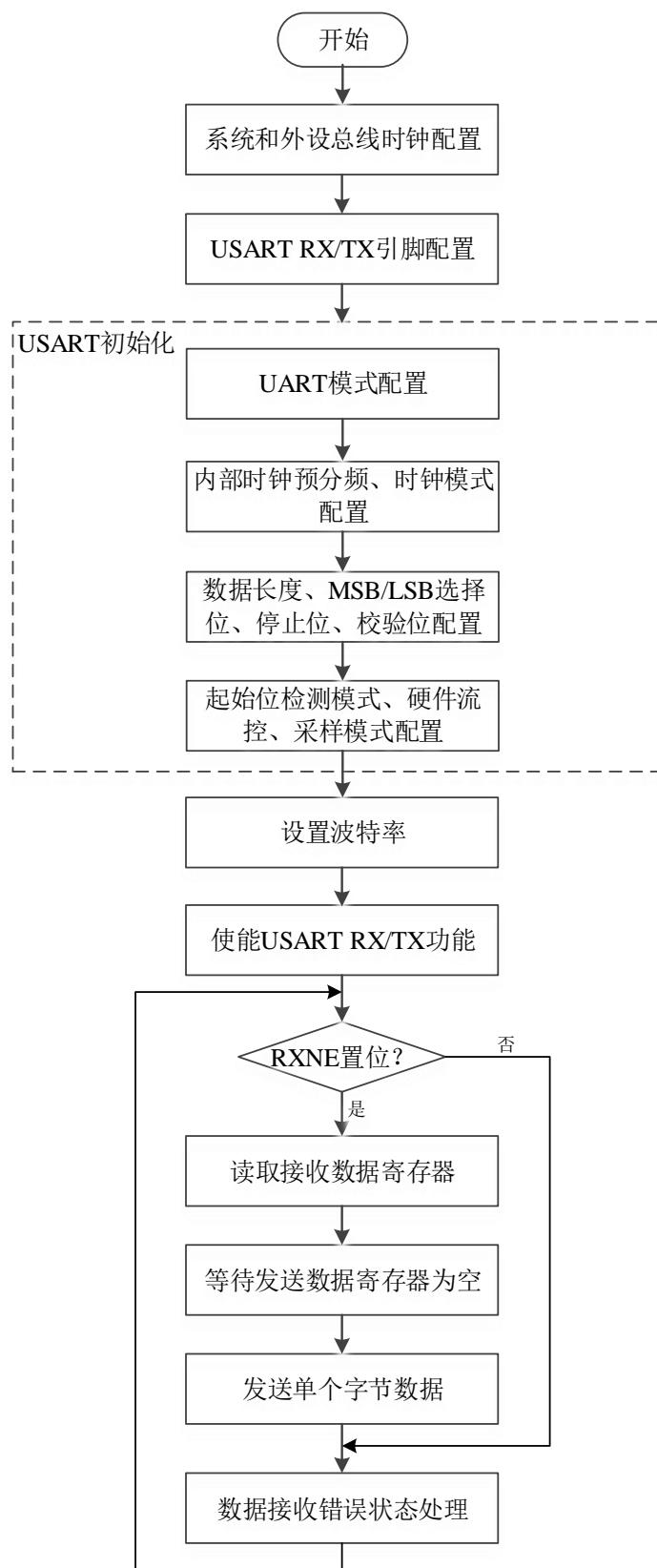
-
- 发送数据开始前，USARTn_CR1.TE=1 之后，将 IO 设为 TX 功能

3.2.6 寄存器介绍

英文说明（缩写）	中文说明
USART Status Register (USART_SR)	USART 状态寄存器
USART Data Register (USART_DR)	USART 数据寄存器
USART Bit Rate Register (USART_BRR)	USART 波特率寄存器
USART Control Register 1 (USART_CR1)	USART 控制寄存器 1
USART Control Register 2 (USART_CR2)	USART 控制寄存器 2
USART Control Register 3 (USART_CR3)	USART 控制寄存器 3
USART Prescaler Register 1 (USART_PR)	USART 预分频寄存器

3.2.7 工作流程介绍

在本章节主要介绍本 AN 使用的样例 `uart_polling_rx_tx` 的工作流程。



4 样例代码

4.1 代码介绍

用户可根据上述的工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载到设备驱动库（Device Driver Library, DDL）的样例代码并使用其中的 USART 的样例进行验证。

以下部分简要介绍本 AN 基于 DDL 的 USART 模块样例 `uart_polling_rx_tx` 代码所涉及的各项配置。

1) 设置 USART 初始化结构体变量：

```
/* USART configure */
static const stc_usart_uart_init_t m_stcInitCfg = {
    UsartIntClkCkNoOutput,
    UsartClkDiv_1,
    UsartDataBits8,
    UsartDataLsbFirst,
    UsartOneStopBit,
    UsartParityNone,
    UsartSamleBit8,
    UsartStartBitFallEdge,
};
```

2) 初始化系统时钟：

```
/* Initialize Clock */
ClkInit();
```

3) 初始化 USART RX/TX 引脚：

```
/* Initialize USART IO */
PORT_SetFunc(USART_RX_PORT, USART_RX_PIN, USART_RX_FUNC, Disable);
PORT_SetFunc(USART_TX_PORT, USART_TX_PIN, USART_TX_FUNC, Disable);
```

4) 初始化 UART 功能：

```
USART_UART_Init(USART_CH, &m_stcInitCfg);
```

5) 设置波特率:

```
/* Set baudrate */  
USART_SetBaudrate(USART_CH, USART_BAUDRATE);
```

6) 使能 USART RX/TX 功能:

```
/*Enable RX && TX function*/  
USART_FuncCmd(USART_CH, UsartRx, Enable);  
USART_FuncCmd(USART_CH, UsartTx, Enable);
```

4.2 代码运行

用户可以通过华大半导体的网站下载到 HC32F460 的 DDL 的样例代码（uart_polling_rx_tx），并配合评估用板（EV-HC32F460-LQFP100-050-V1.1）运行相关代码学习使用 USART 模块。

以下部分主要介绍如何在评估板上运行 USART 样例代码并观察结果：

- 一 确认安装正确的 IAR EWARM v7.7 工具（请从 IAR 官方网站下载相应的安装包，并参考用户手册进行安装）。
- 一 从华大半导体网站下载 HC32F460 DDL 代码。
- 一 下载并运行 usart\uart_polling_rx_tx\中的工程文件：

- 1) 通过 USB 数据线，连接 PC 和板子 J1。
- 2) 打开串口助手软件，配置端口如下参数。

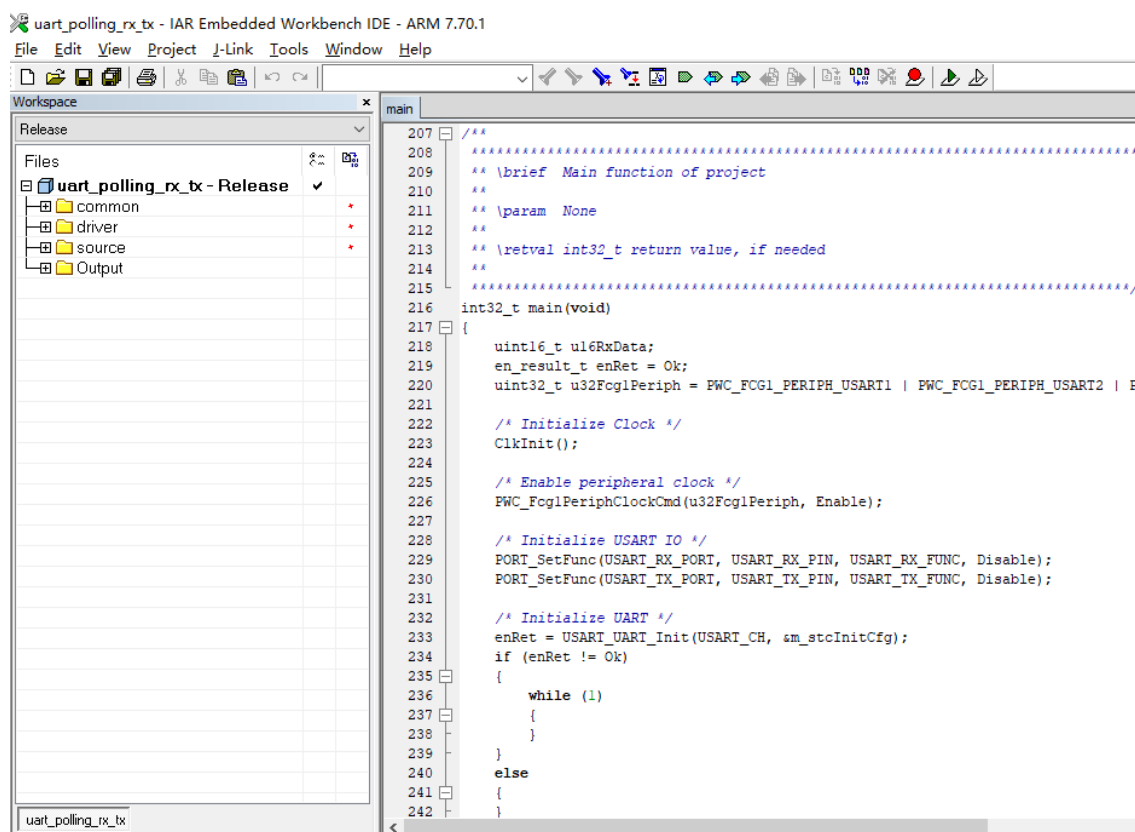
波特率：115200

数据位：8

校验位：None

停止位：1



- 3) 打开 uart_polling_rx_tx\工程，并打开 ‘main.c’ 如下视图：



```

207  /**
208  ****
209  ** \brief Main function of project
210  **
211  ** \param None
212  **
213  ** \retval int32_t return value, if needed
214  **
215  ****
216  int32_t main(void)
217  {
218      uint16_t ul6RxData;
219      en_result_t enRet = Ok;
220      uint32_t u32FcglPeriph = PWC_FCG1_PERIPH_USART1 | PWC_FCG1_PERIPH_USART2 | PWC_FCG1_PERIPH_USART3;
221
222      /* Initialize Clock */
223      ClkInit();
224
225      /* Enable peripheral clock */
226      PWC_FcglPeriphClockCmd(u32FcglPeriph, Enable);
227
228      /* Initialize USART IO */
229      PORT_SetFunc(USART_RX_PORT, USART_RX_PIN, USART_RX_FUNC, Disable);
230      PORT_SetFunc(USART_TX_PORT, USART_TX_PIN, USART_TX_FUNC, Disable);
231
232      /* Initialize USART */
233      enRet = USART_UART_Init(USART_CH, sm_stcInitCfg);
234      if (enRet != Ok)
235      {
236          while (1)
237          {
238              ;
239          }
240      }
241      else
242      {
243          ;
244      }
245  }

```

- 4) 点击  重新编译整个项目。
- 5) 点击  将代码下载到评估板上，全速运行。
- 6) 在串口助手窗口输入字母或数字，窗口回显输入内容。

5 总结

以上章节简要介绍了 HC32F460 系列的 USART，说明了 USART 模块的寄存器及部分操作流程，并且演示了如何使用 USART 样例代码，在实际开发中用户可以根据自己的需要配置和使用 USART 模块。

6 版本信息 & 联系方式

日期	版本	修改记录
2019/3/15	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: 上海市张江高科园区碧波路 572 弄 39 号

邮编: 201203

