32-bit microcontrollers

HC 32 F 460 Series General Purpose Synchronous Asynchronous Transceivers
USART

Applicable objects

F Series	HC 32F460

Table of Contents Table of Contents

1 Abstract

This application note introduces the Universal Synchronous/Asynchronous Receiver/Transmitter (USART) module of HC32F460 series chips, and briefly explains how to use USART for communication through UART mode.

2 USART Introduction

USART is a universal synchronous/asynchronous serial transceiver module, the interface is a highly flexible serial communication device.

USART Key Features:

- Supports full duplex communication

- Transmit and receive baud rate programmable

- Data word length programmable

- Stop bit configurable

- LSB/MSB optional

- Parity check control

- Transmission event detection: receive buffer full, send buffer empty, end of transmission flag

- Support error detection: checksum errors, frame errors, overflow errors

- Transmitter clock output for synchronous transmission

3 USART of HC32F460 Series

3.1 Introduction

HC32F460 series MCUs are equipped with 4 units of Universal Serial Transceiver Module (USART), Universal Serial Transceiver Module

(USART) supports Universal Asynchronous Serial Communication Interface (UART), clock synchronous communication interface, smart card interface (ISO/IEC7816-3), and flexible full duplex data exchange with external devices. Supports modem operation (CTS/RTS operation), multi-processor operation, and UART receive TIMEOUT function in conjunction with the TIMER0 module.

3.2 Description

3.2.1 UART operating mode

UART (Universal Asynchronous Receiver/Transmitter)is a universal asynchronous transceiver/transmitter that uses serial for data exchange to achieve full duplex communication.

1) UART Data Format

A frame of data in UART mode is composed of a start bit, a data bit, a check bit and a stop bit.

Starting position	Data bits	Parity check bits	Stop bit
1 position	8,9 bits	1 bit (optional)	1,2 bits

2) UART Basic Parameters

Baud rate, start bit, data bit, stop bit, check bit and signal level.

3) Main features of HC32F460 UART operation mode:

- Programmable data length: 8-bit/9-bit

- Check function can be configured: odd check/even check/no check

- Stop bit configurable: 1 bit / 2 bits

- Clock source selectable: internal clock source (clock generated by internal baud rate generator)/external clock source (clock input from CKn pin)

- Receiving errors: checksum errors, frame errors, overflow errors

- Modem operation (CTS/RTS)

- Communication between multiple processors

- Built-in digital filter eliminates noise on the receive data line

- Support receive data TIMEOUT function

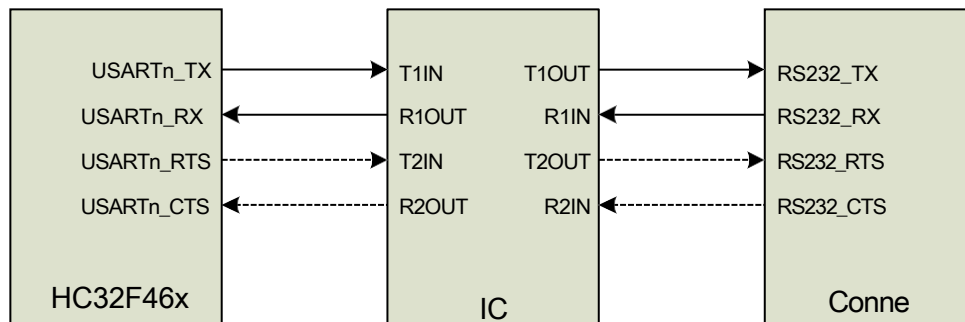- Unit 1 supports stop mode wake-up function

4) UART mode interface connection diagram

- TTL level interface
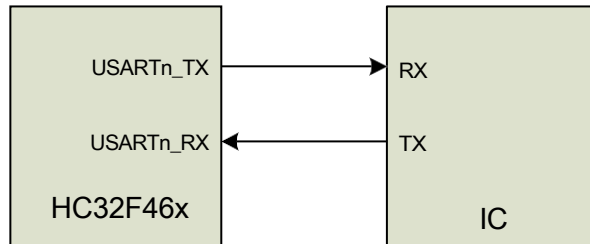
- RS232 interface

Hardware flow control is an optional configuration option and is connected according to the actual application.

- RS485 interface
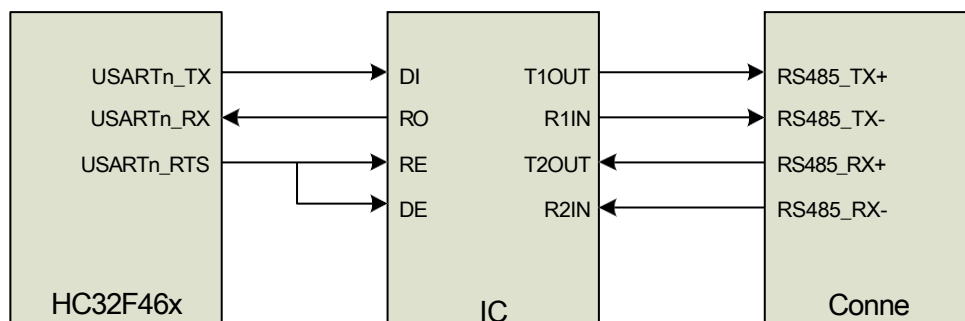
3.2.2 Clock synchronization mode

1) Clock synchronization mode main features:

- Data length: 8 bits

- Receiving error: overflow error

- Modem operation (CTS/RTS)

- Clock source: Internal clock source (clock generated by internal baud rate generator)/External clock source (clock input from CKn pin)

2) Clock synchronization mode data format

Clock synchronization mode A frame of data is composed of 8 data bits without start, check and stop bits.

Starting position	Data bits	Parity check bits	Stop bit
None	8 bits	None	None

3) Clock synchronization mode interface connection diagram

Hardware flow control is an optional configuration option and is connected according to the actual application.
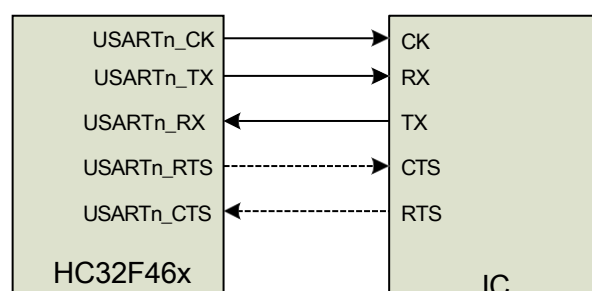
3.2.3 Smart Card Mode

1) Smart card mode main features:

- Data length: 8 bits

- Automatically sends out error signals when calibration errors are detected

- Support data retransmission

2) Smart card mode interface connection schematic

3.2.4 Baud rate calculation

The baud rate generator provides fractional baud rate mode and integer baud rate mode.
When the error of integer baud rate mode is large, fractional baud rate mode can be used to
reduce the baud rate calculation error.

1) The fractional baud rate is invalid and the baud rate calculation formula is as follows

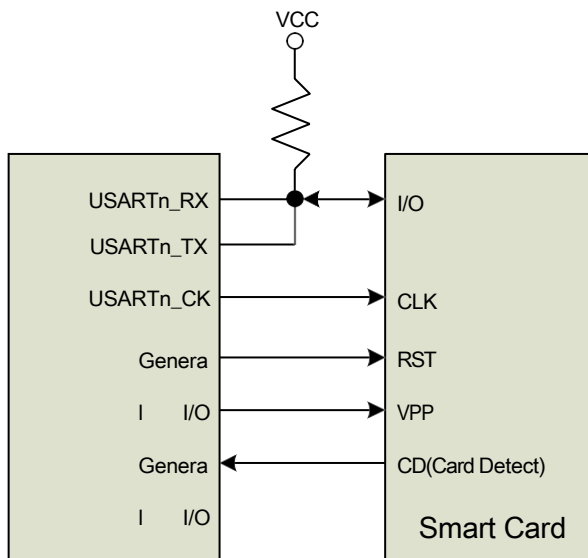| Mode	Baud rate calculation formula	Error E(%) Calculation formula
UART mode Multi-processor mode	$B = \dfrac{C}{8 \times (2 - OVER8) \times (DIV\_Integer + 1)}$	$E(\%) = \left\{ \dfrac{C}{8 \times (2 - OVER8) \times (DIV\_Integer + 1) \times B} - 1 \right\} \times 100$
Clock synchronization mode	$B = \dfrac{C}{4 \times (DIV\_Integer + 1)}$	-
Smart Card Mode	$B = \dfrac{C}{2 \times S \times (DIV\_Integer + 1)}$	$E(\%) = \left\{ \dfrac{C}{2 \times S \times (DIV\_Integer + 1) \times B} - 1 \right\} \times 100$

2) The fractional baud rate is valid and the baud rate calculation formula is as follows

Mode	Baud rate calculation formula	Error E(%) Calculation formula
UART mode Multi-processor mode	$B = \dfrac{C \times (128 + DIV\_Fraction)}{8 \times (2 - OVER8) \times (DIV\_Integer + 1) \times 256}$	$E(\%) = \left\{ \dfrac{C \times (128 + DIV\_Fraction)}{8 \times (2 - OVER8) \times (DIV\_Integer + 1) \times 256 \times B} - 1 \right\} \times 100$
Clock synchronization mode	$B = \dfrac{C \times (128 + DIV\_Fraction)}{4 \times (DIV\_Integer + 1) \times 256}$	
Smart Card Mode	$B = \dfrac{C \times (128 + DIV\_Fraction)}{2 \times S \times (DIV\_Integer + 1) \times 256}$	$E(\%) = \left\{ \dfrac{C \times (128 + DIV\_Fraction)}{2 \times S \times (DIV\_Integer + 1) \times 256 \times B} - 1 \right\} \times 100$

B: Baud rate Unit: Mbps

C: The clock unit set by register USARTn_PR.PSC[1:0] bits:

MHz S: The base clock number of one bit data transmission set by

register USARTn_CR3.BCN OVER8: The value set by

register USARTn_CR1.OVER8

DIV_Integer: Register USARTn_BRR.DIV_Integer set value

DIV_Fraction: Register USARTn_BRR.DIV_Fraction set value

- Taking UART mode as an

  example, the following scenario is

  assumed: the clock source is the

  internal clock source

  PCLK = 8000000Hz

  USARTn_PR.PSC[1:0] value is 0, then C=8000000Hz

USARTn_CR1.OVER8 value is 1, then OVER8=1

Set the baud rate to 115200, then B=115200

- First, bring the parameters OVER8, C, B into the UART integer baud rate mode formula as follows to calculate
  DIV_Integer = 7

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

- Bring the parameters OVER8, C, DIV_Integer into the UART integer baud rate mode formula to calculate the actual baud rate as
  125000, with an error of 8.5069%

- Bringing the parameters OVER8, C, B, DIV_Integer, into the UART fractional baud rate mode formula as follows, the calculation yields DIV_Fraction = 126

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

- Bring the parameters OVER8, C, DIV_Integer, DIV_Fraction into the UART fractional baud rate mode formula and calculate the actual baud rate as 114746 with an error of 0.394%

- Comparing the above calculation results, we can see that the UART decimal baud rate mode correction can significantly reduce the baud rate error

3.2.5 Precautions for use

1) UART Mode Notes:

- When the UART mode transmitter transmit action is disabled (USARTn_CR1.TE=0), the TX pin can be used as a normal IO and the output value and direction can be set. If 0 is output, it causes the receiver to generate a frame error from the receiver, which interrupts data transmission. If 1 is output, it causes the receiver to not detect the start bit and thus cannot start data transmission.

- When the UART mode receiver operates, a frame error can be generated by checking whether the RX line is always 0 to confirm whether the transmission is interrupted by the sender. If the receive data start bit detection method is low level detection, the receive error may occur again if the error flag is cleared and the data continues to be received at all low levels. This problem can be avoided if the receive data start bit detection method is falling edge detection.

2) Clock synchronization mode notes:

- When sending data using the external input clock, wait more than 6 PCLK after the TDR update before inputting the clock. The total input clock period must be greater than or equal to 6PCLK, and the high and low periods must be greater than or equal to 2PCLK.

- When sending data continuously, the next frame of data needs to be updated before the last bit of the current frame is sent.
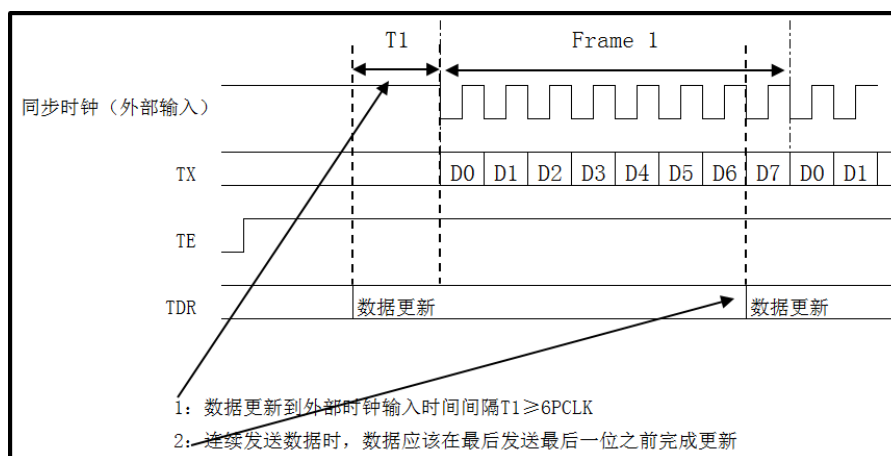
3) Other Notes:

- Writing USARTn_TDR should be done through TI interrupts, and only one write operation can be done in one interrupt.

- The baud rate register value should not be modified during communication.

- To prevent the Hi-Z state of the TX communication line when transmit is disabled, the following method can be used:

  - Pull up

  - Set the TX pin to normal IO output before USARTn_CR1.TE=0 at the end of sending data

3.2.6 Register Introduction

English description (abbreviation)	Chinese Description
USART Status Register (USART_SR)	USART Status Register
USART Data Register (USART_DR)	USART Data Register
USART Bit Rate Register (USART_BRR)	USART baud rate register
USART Control Register 1 (USART_CR1)	USART control register 1
USART Control Register 2 (USART_CR2)	USART control register 2
USART Control Register 3 (USART_CR3)	USART control register 3
USART Prescaler Register 1 (USART_PR)	USART Prescaler Register

3.2.7 Workflow Introduction

This section describes the workflow of the sample uart_polling_rx_tx used in this AN.

```
                        ┌──────────┐
                        │  Start   │
                        └────┬─────┘
                             │
                ┌────────────▼────────────┐
                │  System and peripheral  │
                └────────────┬────────────┘
                             │
                ┌────────────▼────────────┐
                │   USART RX/TX Pin        │
                └────────────┬────────────┘
          ┌ ─ ─ ─ ─ ─ ─ ─ ─ ┼ ─ ─ ─ ─ ─ ─ ─ ─ ┐
          │ USART           │
          │      ┌──────────▼──────────┐        │
          │      │     UART mode       │        │
          │      └──────────┬──────────┘        │
          │      ┌──────────▼──────────┐        │
          │      │ Internal clock      │        │
          │      │ prescaling,         │        │
          │      │ clock mode          │        │
          │      └──────────┬──────────┘        │
          │      ┌──────────▼──────────┐        │
          │      │ Data length,        │        │
          │      │ MSB/LSB selection   │        │
          │      │ bit, stop bit,      │        │
          │      └──────────┬──────────┘        │
          │      ┌──────────▼──────────┐        │
          │      │ Start bit detection │        │
          │      │ mode, hardware flow │        │
          │      └──────────┬──────────┘        │
          └ ─ ─ ─ ─ ─ ─ ─ ─ ┼ ─ ─ ─ ─ ─ ─ ─ ─ ┘
                ┌────────────▼────────────┐
                │        Set baud         │
                └────────────┬────────────┘
                ┌────────────▼────────────┐
                │   Enables USART RX/TX   │
                └────────────┬────────────┘
                             │
             ┌──────────────▶│
             │       ◇───────▼───────◇    N
             │       < RXNE set? >─────────┐
             │       ◇───────┬───────◇     │
             │               │ Yes         │
             │    ┌──────────▼──────────┐  │
             │    │  Read the receive   │  │
             │    └──────────┬──────────┘  │
             │    ┌──────────▼──────────┐  │
             │    │ Wait for the send   │  │
             │    │       data          │  │
             │    └──────────┬──────────┘  │
             │    ┌──────────▼──────────┐  │
             │    │ Sending a single    │  │
             │    │       byte          │  │
             │    └──────────┬──────────┘  │
             │    ┌──────────▼──────────┐◀─┘
             │    │ Data reception error│
             │    └──────────┬──────────┘
             └───────────────┘
```
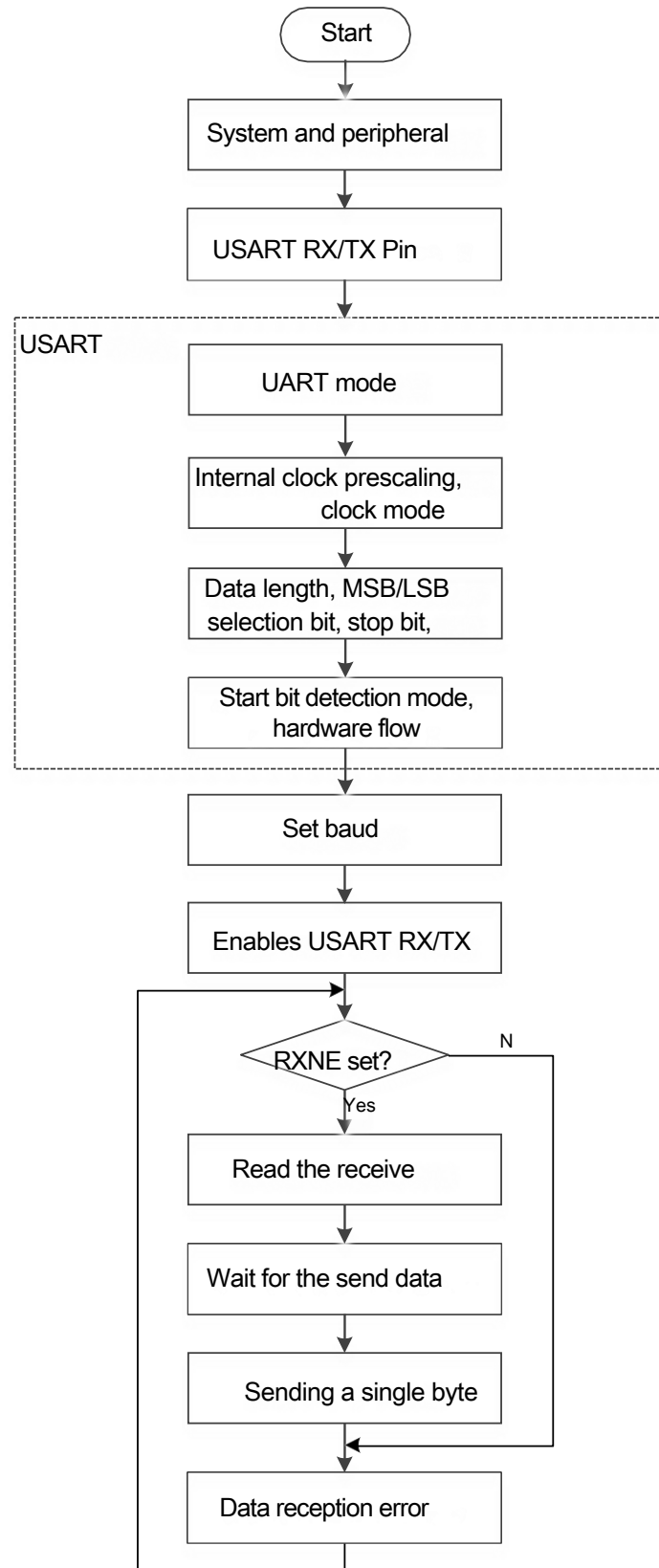
4   Sample Code

4.1   Code Introduction

Users can write their own code to learn and verify the module according to the above workflow, or download the sample code of Device Driver Library (DDL) directly from the website of UW Semiconductors and use the sample of USART to verify.

The following section briefly describes the configuration involved in the sample uart_polling_rx_tx code of the DDL-based USART module of this AN.

1) Set the USART initialization structure variable:

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

2) Initialize the system clock:

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

3) Initialize the USART RX/TX pins:

```
/* 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) Initialize UART Function:

```
USART_UART_Init(USART_CH, &m_stcInit Cfg).
```

5) Set baud rate:

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

6) Enables the USART RX/TX function:

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

Applicati

4.2 Code Run

Users can download the sample code of HC32F460 DDL (uart_polling_rx_tx) from UW
Semiconductors website and run the code with the evaluation board (EV-HC32F460-
LQFP100-050-V1.1) to learn how to use the USART module.

The following section describes how to run the USART sample code on the evaluation board
and observe the results:

— Verify that the correct IAR EWARM v7.7 tool is installed (please download the appropriate
   installation package from the official IAR website and refer to the user manual for
   installation).

— Download the HC32F460 DDL code from the UW Semiconductors website.

— Download and run the project file in usart\uart_polling_rx_tx\ at

  1) Connect the PC to the board J1 via the USB data cable.

  2) Open the Serial Assistant software and

      configure the port with the following

      parameters. Baud rate: 115200

      Data bits: 8

      Checksum bit:

      NoneStop bit:

      1

  3) Open the uart_polling_rx_tx\ project and open the 'main.c' view as follows:
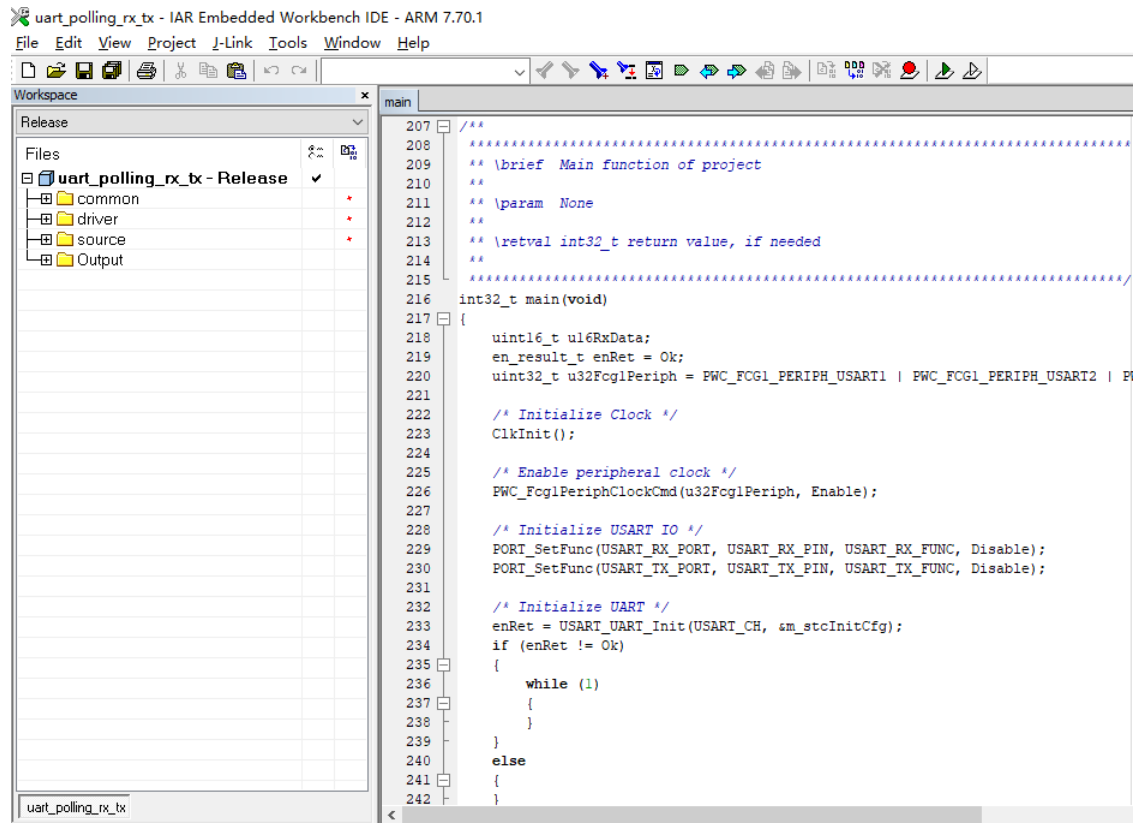
4) Click to recompile the entire project.

5) Click Download the code to the evaluation board and run it at full speed.

6) Enter letters or numbers in the Serial Assistant window, and the window will display back the input content.

5   Summary

The above section briefly introduces the USART of HC32F460 series, explains the registers and part of the operation flow of USART module, and demonstrates how to use the USART sample code, so that users can configure and u s e   t h e   U S A R T   module according to their needs in actual development.

6 Version Information & Contact

Date	Versions	Modify records
2019/3/15	Rev1.0	Initial Release
2020/8/26	Rev1.1	Update supported models

If you have any comments or suggestions in the process of purchase and use, please feel free to contact us.

Email: mcu@hdsc.com.cn

Website: http://www.hdsc.com.cn/mcu.htm

Address: 10/F, Block A, 1867 Zhongke Road, Pudong

New Area, Shanghai, 201203, P.R. China