Use UART for RS485 Communication

Example Code Introduction for 8-bit NuMicro® Family

Information

Application	This code is to use the UART of the MCU for RS485 communication.
BSP Version	N76E003 BSP Keil C51 v1.0.6
Hardware	NuTiny-SDK-N76E003 v2.1

1 Function Description

1.1 Introduction

Typical serial communication standards are RS232 and RS485, RS232 is full-duplex point-to-point communication, and RS485 is half-duplex communication (2-wire system), which can be networked from point to point, and RS485 is the voltage across the cable. The difference is used to indicate the transfer signal, which is quite different from the RS232 electrical characteristics. RS485 only specifies the electrical characteristics of the receiver and transmitter, and does not specify or recommend any data protocol.

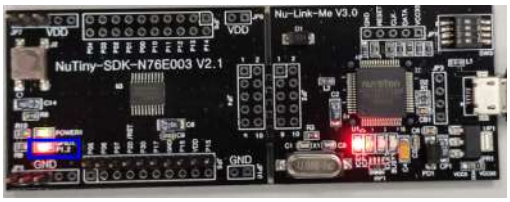
1.2 Principle

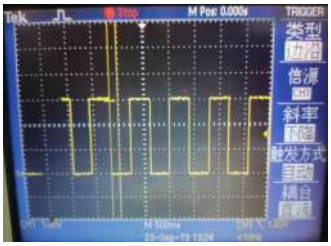
RS232 can use the UART of the microcontroller to communicate, of course, RS485 can also use the UART of the microcontroller to communicate. On the hardware circuit, only the RS232 driver chip is replaced by the RS485 driver chip.

1.3 Demo Result

This demo demonstrates the need for two sets of boards, one for the master and one for the slave. The slave board is powered on first, and then the master board is powered on. If there is no problem with the circuit, you can see the LED on the motherboard flashing (or using the oscilloscope to observe the level is constantly flipping). Note that the MCU and 485 chip connections must be connected to the transmit and receive enable pins in addition to the normal TX and RX pins.

As shown in the figure below, after the normal communication is successful, the red LED on the board will flash, use the oscilloscope to observe the P12 pin, and the pin will have a waveform output.





2 Code Description

Explain important code workflows:

```
    /* Master send and receive data */
           Send_Data(CMD_ACTIVE, 0, NULL);// Check if the slave is ready
           if (Recv_Data(&u8type, g_u8StatusBuf) && u8type == CMD_READY)
           {
               Send_Data(CMD_GETDATA, 0, NULL);//Request data from the slave
               if (Recv_Data(&u8type, g_u8StatusBuf) && u8type == CMD_SENDDATA)
               {
                   // Successfully receive slave data
                   Clr_StatusBuf();
                   P12 = ~P12;
               }
           }
    /* Slave send and receive data */

           if (Recv_Data(&u8type, NULL) == 0)                // Receive frame error
               continue;
           switch (u8type)
           {
               case CMD_ACTIVE:  // The master asks the slave whether it is ready
                   Send_Data(CMD_READY, 0, NULL);            // Send READY command
                   break;

               case CMD_GETDATA:                             // master read slave request
                   Clr_StatusBuf();
                   Get_Stat();                               // Data acquisition function
                   Send_Data(CMD_SENDDATA, 10, g_u8StatusBuf);
                   break;

               default:
                   break;
           }
```

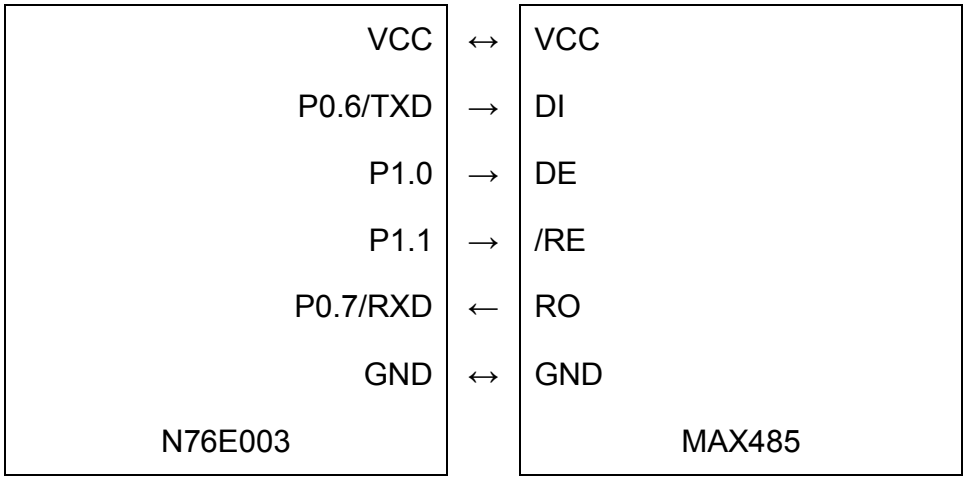
3 Software and Hardware Environment

Software Environment

- BSP version
  - N76E003 BSP Keil C51 v1.0.6
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuTiny-SDK-N76E003
  - MAX485
- Diagram



4 Directory Information

- EC_N76E003_UART_RS485_V1.00
  - Library — Sample code header and source files
    - Device — compliant device header file
    - Startup — A51 startup file and executeable file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click N76E003_UART_RS485_Master.uvproj. and N76E003_UART_RS485_Slave.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Sep. 20, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.