Modbus protocol SDK quick start

Protocol Introduction

This routine introduces how to use the serial port 2 of the STM32Core platform to connect the Wittur smart serial port 485 protocol, and then directly print data through the serial port 1, receive sensor data and communicate with the sensor;
Please read the relevant sensor manual before viewing this routine to understand the protocol used by the sensor and the basic functions of the sensor

Device Wiring

First prepare the WitMotion 485 communication sensor, a 485 to TTL level module, STM32Core development board, and a serial port 3 in 1 adapter. Please check the below Wiring:

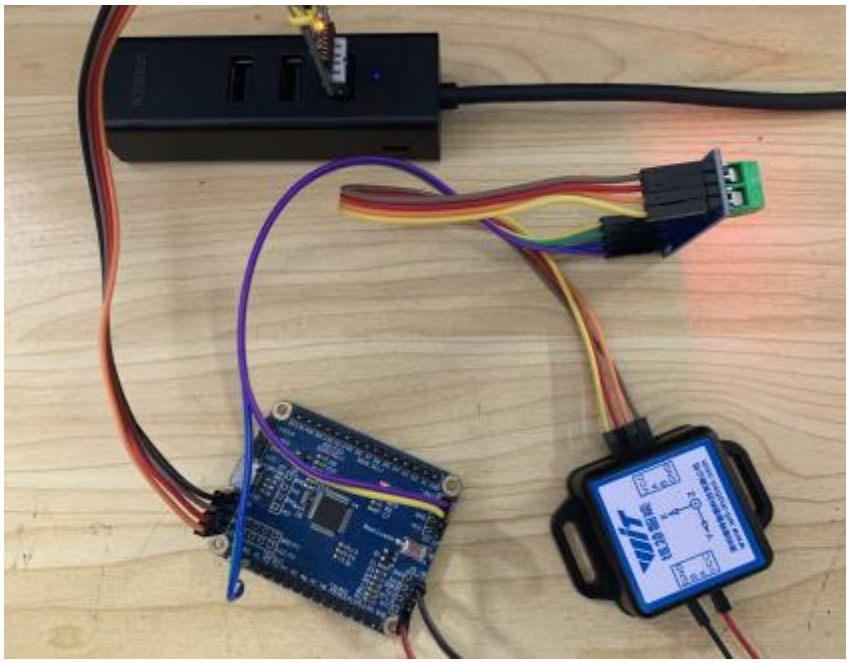
```
USB-TTL                    STM32Core                 485 adapter board          WT901C485
 VCC          -----          VCC            ----          VCC          ---         VCC
 TX           -----        RX1  (GPIOA_10)
 RX           -----        TX1  (GPIOA_9)


GND           -----          GND    ----                   GND          ---         GND

                           RX2  (GPIOA_3)  ----           RO      A      ---          A
                           TX2  (GPIOA_2)  ----           DI      B      ---          A
                           T/R  (GPIOA_1)  ----             RE和DE短接

-------------------------------------
*/
```

Below is the demo wiring diagram:



Download the program to the development board and open the serial port at the same time Debug assistant, power on again, and the following information will be displayed:

```
[17:22:03.551]IN←◆
********************** wit-motion modbus example  ************************

[17:22:03.901]IN←◆9600 baud find sensor


************************         WIT_SDK_DEMO   ************************
************************          HELP          ************************
UART SEND:a\r\n   Acceleration calibration.
UART SEND:m\r\n   Magnetic field calibration,After calibration send:   e\r\n   to indicate the end
UART SEND:U\r\n   Bandwidth increase.
UART SEND:u\r\n   Bandwidth reduction.
UART SEND:B\r\n   Baud rate increased to 115200.
UART SEND:b\r\n   Baud rate reduction to 9600.
UART SEND:h\r\n   help.
******************************************************************************

[17:22:04.463]IN←◆acc:0.007 -0.011 0.997
gyro:-0.977 -0.610 0.366
angle:-0.835 -0.549 179.231
mag:-30 -9278 -5110

[17:22:04.982]IN←◆acc:0.007 -0.011 0.997
gyro:-0.977 -0.610 0.305
angle:-1.203 -0.780 179.648
mag:-275 -9269 -5134
```

The module can be configured by sending corresponding instructions through the prompt information.

Initialization

Modbus is an interactive form of one question and one answer, so it is necessary to continuously query the module before data can be sent back.
For API function introduction, please read the WIT_C_SDKAPI function description document.

```
int main(void)
{
    float fAcc[3], fGyro[3], fAngle[3];
    int i;
    SysTick_Init();
    Usart1Init(115200);
    Usart2Init(9600);
    RS485_IO_Init();
    WitInit(WIT_PROTOCOL_MODBUS, 0x50);
    WitSerialWriteRegister(SensorUartSend);
    WitRegisterCallBack(CopeSensorData);
    printf("\r\n********************** wit-motion modbus example  ************************\r\n");
    AutoScanSensor();
```

Usart1Init(115200); //Initialize the print data serial port
Usart2Init(9600); //Initialize the serial port connected to the module
RS485_IO_Init(); //Initialize the half-duplex 485T/R control pin
WitInit(WIT_PROTOCOL_MODBUS, 0x50); //Initialize the modbus protocol and set the device address
WitSerialWriteRegister(SensorUartSend); //Register write callback function
WitRegisterCallBack(CopeSensorData); //register to get sensor data callback function
AutoScanSensor(); //Automatically search for sensors

Receive sensor data

Obtain the data

We will open up an array to store the read data in the array, and read the corresponding data directly according to the index. Since the module used this time is the modbus protocol, when you need to ask, you need to provide the data start address and the amount of data to be read. For example, the example calls WitReadReg(AX,12) every 500ms; the starting address is AX (acceleration X value) to continuously read 12 data volumes, then the final effect is to read the three-dimensional acceleration, three-dimensional angular velocity, three-dimensional angle, and three-dimensional magnetic field of the module data into the specified index array. Finally print it out.

```
while (1)
{
    WitReadReg(AX, 12);
    delay_ms(500);
    CmdProcess();
    if(s_cDataUpdate)
    {
        for(i = 0; i < 3; i++)
        {
            fAcc[i] = sReg[AX+i] / 32768.0f * 16.0f;
            fGyro[i] = sReg[GX+i] / 32768.0f * 2000.0f;
            fAngle[i] = sReg[Roll+i] / 32768.0f * 180.0f;
        }
        if(s_cDataUpdate & ACC_UPDATE)
        {
            printf("acc:%.3f %.3f %.3f\r\n", fAcc[0], fAcc[1], fAcc[2]);
            s_cDataUpdate &= ~ACC_UPDATE;
        }
        if(s_cDataUpdate & GYRO_UPDATE)
        {
            printf("gyro:%.3f %.3f %.3f\r\n", fGyro[0], fGyro[1], fGyro[2]);
            s_cDataUpdate &= ~GYRO_UPDATE;
        }
        if(s_cDataUpdate & ANGLE_UPDATE)
        {
            printf("angle:%.3f %.3f %.3f\r\n", fAngle[0], fAngle[1], fAngle[2]);
            s_cDataUpdate &= ~ANGLE_UPDATE;
        }
        if(s_cDataUpdate & MAG_UPDATE)
        {
            printf("mag:%d %d %d\r\n", sReg[HX], sReg[HY], sReg[HZ]);
            s_cDataUpdate &= ~MAG_UPDATE;
        }
    }
```

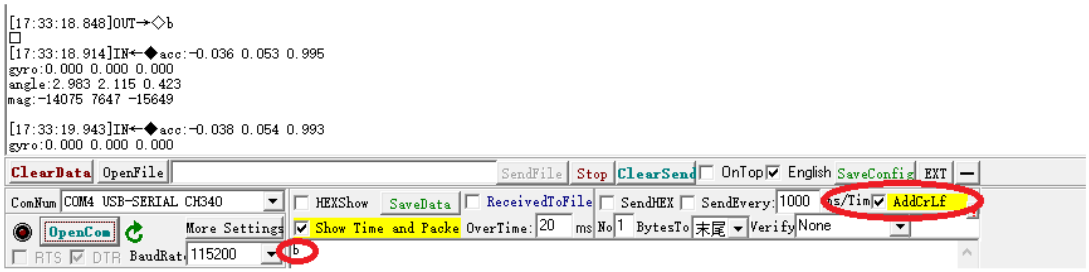
WitReadReg(AX, 12);//Interval reading module, the address starts from AX (acceleration X) to read 12 bytes of data. Refer to REG.H register list

Config the sensor

The parameters of the module can be set through the function CmdProcess();. Such as acceleration calibration, magnetic field calibration and modifying the baud rate.

```
static void ShowHelp(void)
{
    printf("\r\n************************     WIT_SDK_DEMO    ************************");
    printf("\r\n************************          HELP           ************************\r\n");
    printf("UART SEND:a\\r\\n   Acceleration calibration.\r\n");
    printf("UART SEND:m\\r\\n   Magnetic field calibration,After calibration send:   e\\r\\n   to indicate the end\r\n");
    printf("UART SEND:U\\r\\n   Bandwidth increase.\r\n");
    printf("UART SEND:u\\r\\n   Bandwidth reduction.\r\n");
    printf("UART SEND:B\\r\\n   Baud rate increased to 115200.\r\n");
    printf("UART SEND:b\\r\\n   Baud rate reduction to 9600.\r\n");
    printf("UART SEND:s\\r\\n   Sleep sensor, wake up again.\r\n");
    printf("UART SEND:h\\r\\n   help.\r\n");
    printf("******************************************************************************\r\n");
}
```

Open the serial port assistant, send the command b\r\n, and then observe the phenomenon.
Note: When the option of adding carriage return and line feed is checked in the serial port assistant, just send a b command;
If it is not checked, you need to send the command b\r\n..



After sending b\r\n, the module will search for the device again, and compare the sending instructions described in the HELP message. The baud rate indicates that the setting is correct. Use other commands as needed.

Acceleration calibration

Some common API function interfaces are defined in the file wit_c_sdk.c, which only need to be called.

```
/*Acceleration calibration demo*/
int32_t WitStartAccCali(void)
{
    printf("Star ACC Cali\r\n");
    if(WitWriteReg(KEY, KEY_UNLOCK) != WIT_HAL_OK)      return  WIT_HAL_ERROR;// unlock reg
    delay_ms(200);
    if(WitWriteReg(CALSW, CALGYROACC) != WIT_HAL_OK)    return  WIT_HAL_ERROR;
    return WIT_HAL_OK;
}
```

Magnetic Field Calibration

Magnetic field calibration needs to be sent to start calibration first, and then circle the three axes of the sensor to end the calibration. First call WitStartMagCali(); to start the magnetic field calibration, then rotate the sensor three-axis, and then call WitStopMagCali(); to end the calibration.

```
/*Magnetic field calibration demo*/
int32_t WitStartMagCali(void)
{
    if(WitWriteReg(KEY, KEY_UNLOCK) != WIT_HAL_OK)  return WIT_HAL_ERROR;
    delay_ms(200);
    if(WitWriteReg(CALSW, CALMAGMM) != WIT_HAL_OK)  return  WIT_HAL_ERROR;
    delay_ms(200);
    printf("\r\nPlease rotate the three axes of the sensor.\r\n");
    printf("\r\nAfter the rotation is completed, send the command "e\\r\\n"  \
    through the serial port to indicate the end.\r\n");
    return WIT_HAL_OK;
}

int32_t WitStopMagCali(void)
{
    printf("\r\nEnd of magnetic field calibration\r\n");
    WitWriteReg(KEY, KEY_UNLOCK);   // unlock reg
    delay_ms(200);
    WitWriteReg(CALSW, NORMAL);
    delay_ms(200);
    WitWriteReg(SAVE, SAVE_PARAM);
    return WIT_HAL_OK;
}
```