**DWIN T5L Screen C51 Development**

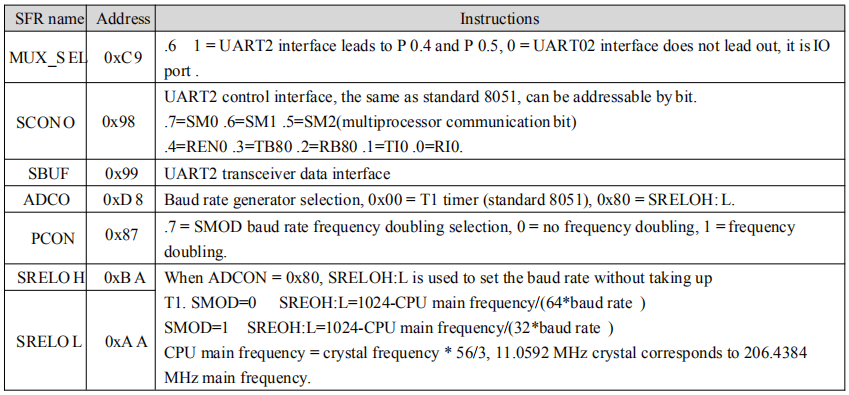
**9. UART**

# Introduction

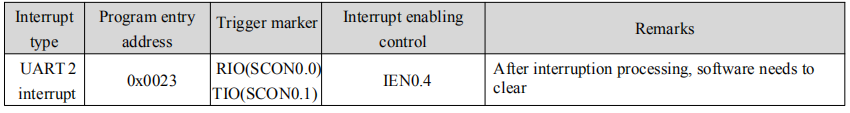
In this chapter, we will learn about the UART serial port. The serial port peripherals are particularly important in the field of embedded development. DWIN T5L chip provides a total of five serial ports: UART1, UART2, UART3, UART4, and UART5, but the UART1 serial port is occupied by the GUI core, which is used for external WIFI module or downloading and debugging, that is to say, our C51 code cannot use the UART1 serial port, and the remaining four serial ports of UART2-UART5 are all reserved for our C51 core.

The two serial ports of UART2 and UART3 are multiplexed with the common IO pin function, and which function to use is selected through the MUX\_SEL register. The two serial ports of UART4 and UART5 are dedicated serial ports, which are not duplicated with other peripheral functions. The usage of these four serial ports is basically the same, as long as you learn one, the other analogies are fine. The registers related to the UART serial port are shown below:

1. **The relevant settings of UART2 interruption**

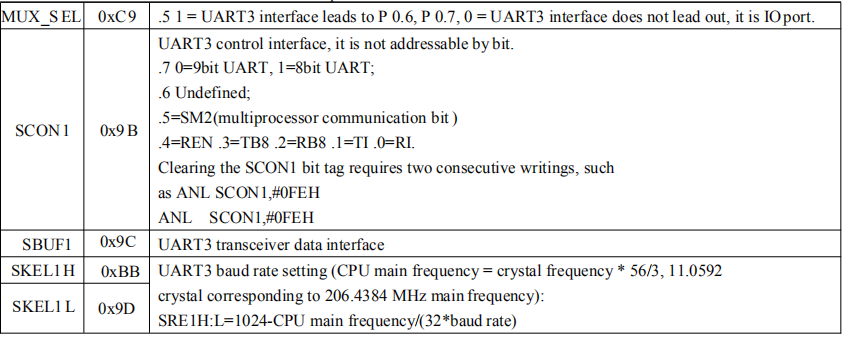


1.1 The relevant settings of UART2 register

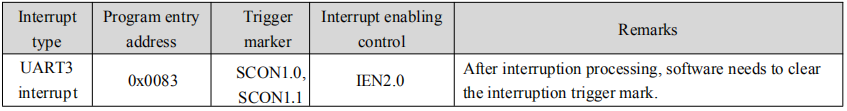


1.2 The relevant settings of UART2 interruption

1. **The relevant settings of UART3**

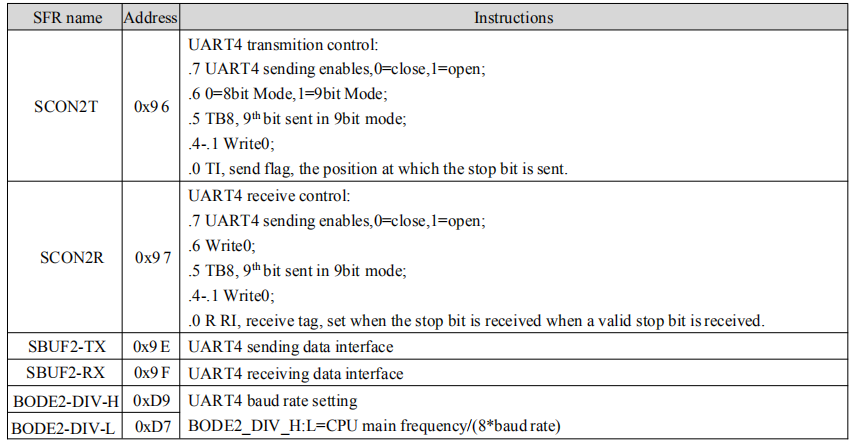


1.3 The relevant settings of UART3 register

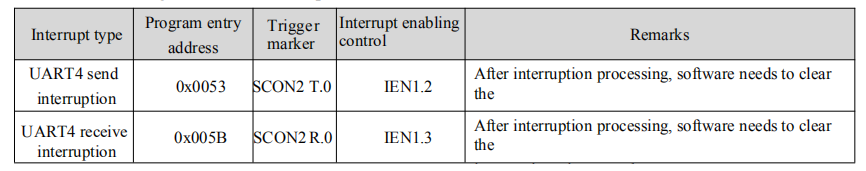


1.4 The relevant settings of UART3 interruption

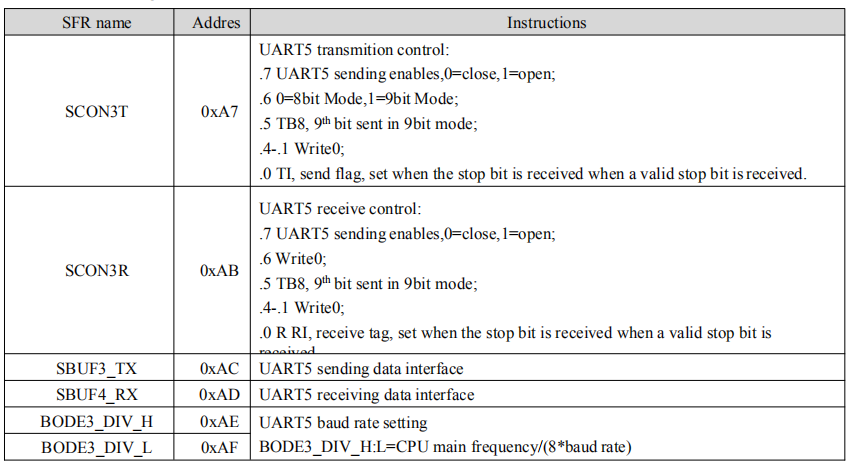
1. **The relevant settings of UART4**



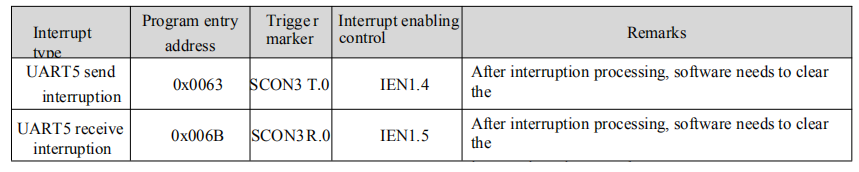
1.5 The relevant settings of UART4 register

1.6 The relevant settings of UART4 interruption

1. **The relevant settings of UART5**



1.7 The relevant settings of UART5 register



1.8 The relevant settings of UART5 interruption

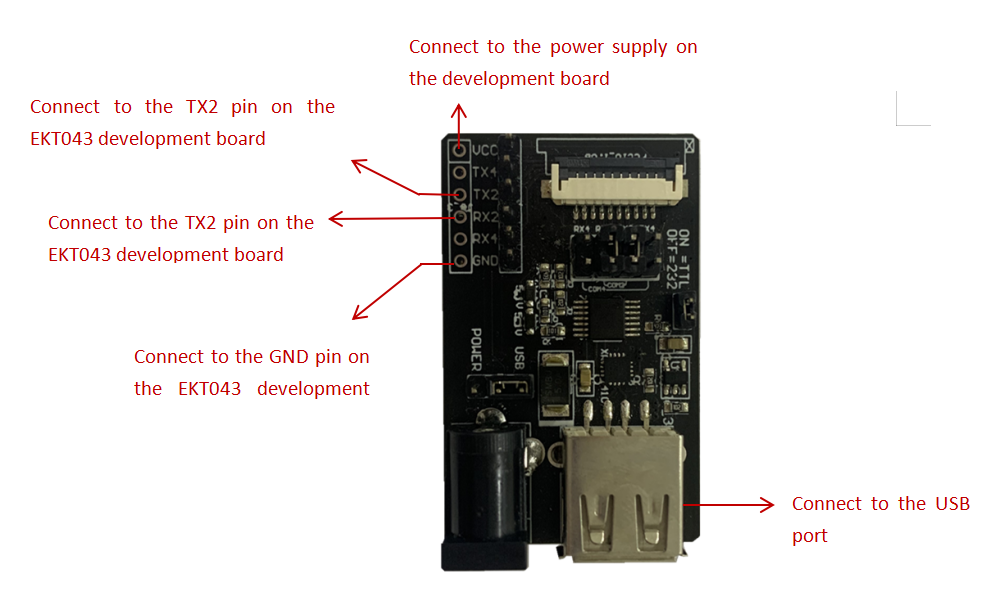
# Routine design

## 2.1 Function introduction

The baud rate of serial port 2 is 115200, enable its accept interrupt, accept serial port data packets ending with "\r\n" or '\n' in the interrupt service routine, after receiving the complete data packet, display the data on the interface, and add the prefix of "T5L\_C51:" and return it to the sender. In order to simulate sending data packets to the serial port 2 of the T5L, we have to use the "serial port debugging assistant" on the PC, and since the usb port on the PC cannot communicate directly with our T5L serial port, we have to use a "USB to TTL hardware module" here, as follows Pictured:



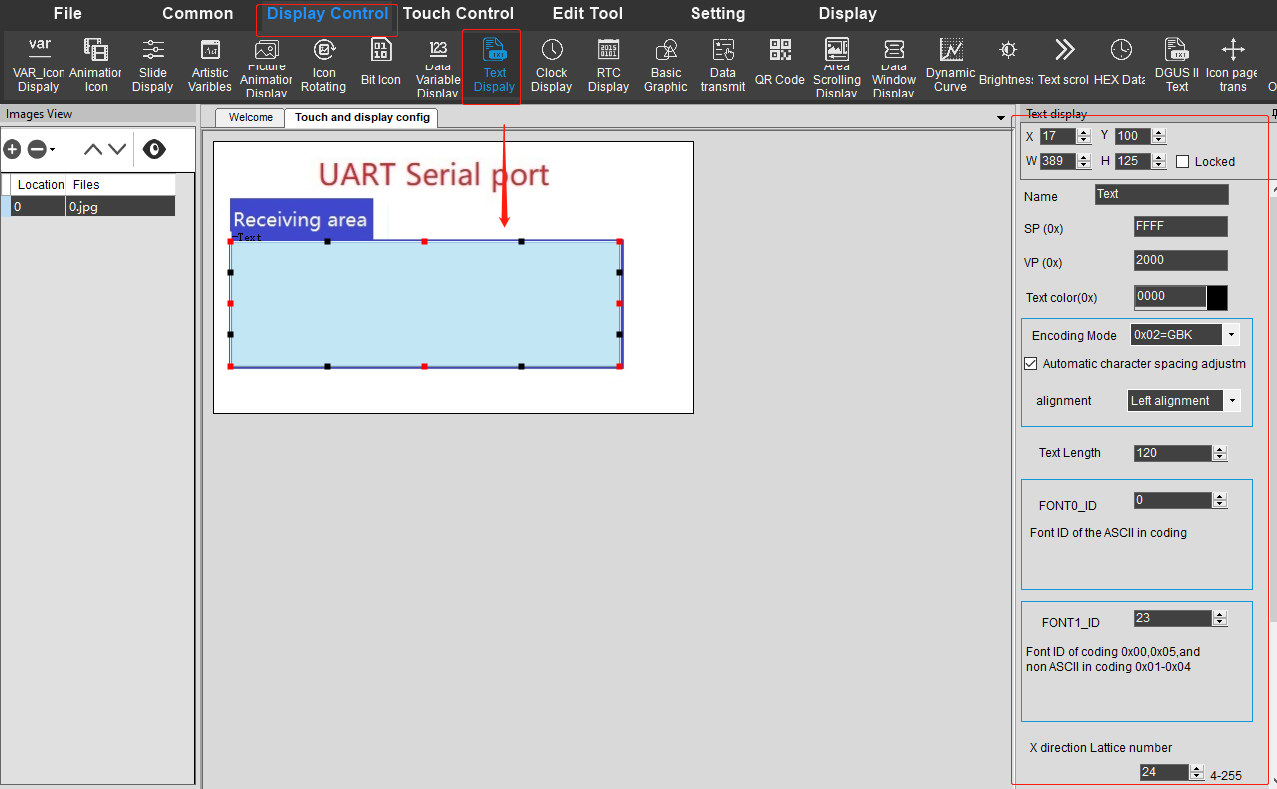
2.1.1 Serial debugging assistant



2.1.2 USB to TTL hardware module and hardware wiring diagram

## 2.2 GUI design

Mainly use a "text display "control, and then startup interface as shown below:



2.2.1 Startup interface

## 2.3 C51 design

We can directly use the C51 project template already created in chapter 1, and then modify it on the basis of that. The amount of code in this chapter is relatively large, so we need to program it in a modular way, mainly the packaging of the UART2 serial driver. We create a UART2 subfolder in the HARDWARE folder, and then create its uart2.c and uart2.h driver files in the UART2 subfolder.

1. **The code of the uart2.h file is as follows:**

#ifndef \_\_UART2\_H\_\_

#define \_\_UART2\_H\_\_

#include "sys.h"

#include <stdio.h>

#define UART2\_INT\_EN 1 //Whether the serial port interrupt is enabled

#define UART2\_PACKET\_OK 0x8000 //Whether the complete packet was received

#define UART2\_PACKET\_LEN 0x7fff //Length of packet

#define UART2\_PACKET\_MAX\_LEN 100 //Defines the maximum length of the serial port package, excluding '\n' or "\r\n" end tags

//Variable declaration

extern xdata u16 uart2\_rx\_sta;

extern xdata u8 uart2\_buf[];

//Function declaration

void uart2\_init(u32 baud);

void u2\_send\_byte(u8 byte);

void u2\_send\_bytes(u8 \*bytes,u16 len);

#endif

1. **The code of the uart2.c file is as follows:**

#include "uart2.h"

#if(UART2\_INT\_EN)

xdata u16 uart2\_rx\_sta;//bit15Used to mark whether a complete data packet has been received, bit[14:0] is used to store the length of the current data packetxdata u8 uart2\_buf[UART2\_PACKET\_MAX\_LEN+2];//Leave 2 blank characters

xdata u8 uart2\_step;

//Serial port 2 interrupt service routine

//When sending data, the interrupt must be turned off, here is only responsible for processing the receiving interrupt.

void uart2\_isr() interrupt 4

{

u8 res;

if(RI0)//The serial port accepting the interrupt

{

RI0 = 0;//Clear accept interrupt flag

res = SBUF0;//Read serial data

if(uart2\_rx\_sta&UART2\_PACKET\_OK)//The received data has not been processed yet

return;

if(uart2\_step==0)//The process of accepting data

{

if(res=='\r')//If the "\r\n" end marker is received, it is considered that the packet acceptance is complete

uart2\_step = 1;//Enter the process of accepting the '\n' token

else if(res=='\n')//If the '\n' end marker is received, it is also considered that the packet is accepted as complete

uart2\_rx\_sta |= UART2\_PACKET\_OK;//Mark packet acceptance complete

else//Accept data

{

if(uart2\_rx\_sta>=UART2\_PACKET\_MAX\_LEN)

uart2\_rx\_sta = 0;//The data is too large, discard it, and start receiving from the beginning

uart2\_buf[uart2\_rx\_sta++] = res;//Store valid data

}

}else if(uart2\_step==1)//The process of judging the end tag

{

uart2\_step = 0;

if(res=='\n')

uart2\_rx\_sta |= UART2\_PACKET\_OK;//Mark packet acceptance complete

else

uart2\_rx\_sta = 0;//The next character of '\r' is not '\n', it is considered that the reception is wrong, and the reception starts from the beginning

}

}

}

#endif

//Serial port 2 initialization

void uart2\_init(u32 baud)

{

MUX\_SEL |= 0x40;//Setting bit6 to 1 means to export the uart2 interface to P0.4 and P0.5 P0MDOUT &= 0xCF;

P0MDOUT |= 0x10;//Set the corresponding IO port output and input

ADCON = 0x80;//Select SREL0H:L as baud rate generator

SCON0 = 0x50;//Accept enable and mode settings

PCON &= 0x7F;//SMOD=0

//Baud rate setting, the formula is:

//SMOD=0 SREL0H:L=1024-main frequency/(64\*baud rate),SMOD=1 SREL0H:L=1024-main frequency/(32\*baud rate)

baud = 1024-(u16)(3225600.0f/baud);

SREL0H = (baud>>8)&0xff;

SREL0L = baud&0xff;

#if(UART2\_INT\_EN)

ES0 = 1;//interrupt enable

EA = 1;

//xdata variables must be initialized in functions

uart2\_rx\_sta = 0;

uart2\_step = 0;

#else

ES0 = 0;

#endif

}

//Send a byte

void u2\_send\_byte(u8 byte)

{

ES0 = 0;//Close the serial port 2 interrupt first

SBUF0 = byte;

while(!TI0);

TI0 = 0;

#if(UART2\_INT\_EN)

ES0 = 1;//Re-open interrupt

#endif

}

//Send data

void u2\_send\_bytes(u8 \*bytes,u16 len)

{

u16 i;

ES0 = 0;//Close the serial port 2 interrupt first

for(i=0;i<len;i++)

{

SBUF0 = bytes[i];

while(!TI0);

TI0 = 0;

}

#if(UART2\_INT\_EN)

ES0 = 1;//Re-open interrupt

#endif

}

//Implement printf function with uart2 serial port

char putchar(char c)

{

u2\_send\_byte(c);

return c;

}

**The code of the main.c file is as follows:**

#include "sys.h"

#include "uart2.h"

void main(void)

{

u16 len;

sys\_init();//System initialization

uart2\_init(115200);//Initialize serial port 2

while(1)

{

if(uart2\_rx\_sta&UART2\_PACKET\_OK)//Received serial data packet

{

len = uart2\_rx\_sta&UART2\_PACKET\_LEN;//Get the length of the serial port data packet, excluding the length of the "\r\n" or '\n' terminator

uart2\_buf[len++] = 0;//Add 2 null characters at the end

uart2\_buf[len++] = 0;

printf("T5L\_C51:%s\r\n",uart2\_buf);//Return the received packet with the "T5L\_C51:" prefix to the sender

sys\_write\_vp(0x2000,uart2\_buf,len/2+1);//At the same time, the data packets are displayed on the interface

uart2\_rx\_sta = 0;//Cleared to 0 means that the serial port package has been processed

}

}

}