DWIN T5L Smart LCMs' UART C51 Program

CONTENTS

1. Introduction 1

1.1 DWIN protocol 2

1.2 UART example 3

2. UART 2 8

2.1 How to initial UART 2 8

2.2 Loading UART 2 to send buffer 12

2.3 UART 2 data upload automatically 13

2.4 UART2 interrupt receiving function 16

2.4.1 Interrupt control SFR 17

2.4.2 Interrupt priority 18

3. UART 3 21

3.1 How to initial UART 3 21

3.2 Loading UART 3 to send buffer 23

3.3 UART3 interrupt receiving function 24

4. UART 4 26

4.1 How to initial UART 4 26

4.2 Loading UART 4 to send buffer 28

4.3 UART4 interrupt receiving function 29

4.4 UART4 interrupt sending function 30

5. UART 5 31

5.1 How to initial UART 5 31

5.2 Loading UART 5 to send buffer 33

5.3 UART5 interrupt receiving function 34

5.4 UART5 interrupt sending function 35

6. Data upload automatically 36

7. Deal UART data 37

8. UART data handle 39

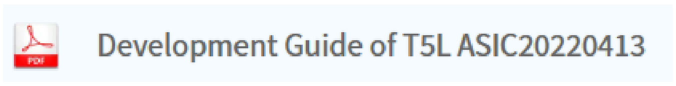
9. UART handle function 41

1. Introduction

The DWIN T5L chip provides a total of five serial ports: UART1, UART2, UART3, UART4, and UART5. However, the UART1 serial port is occupied by the GUI core and is reserved for external use, such as connecting to a WiFi module or for downloading and debugging purposes. In other words, our C51 code cannot utilize the UART1 serial port. The remaining UART2 to UART5 ports are all allocated to our C51 core, which is quite abundant.

Among them, UART2 and UART3 share functionality with general-purpose I/O pins and can be selected using the MUX_SEL register. On the other hand, UART4 and UART5 are dedicated serial ports and do not share functionality with other peripheral features. The usage of these four serial ports is essentially the same. Once you understand one, the others can be learned accordingly.

Related document of UART please refer to: “Development-Guide-of-T5L-ASIC20220413”


Development Guide of T5L ASIC20220413

Download from: https://www.dwin-global.com/development-guide/

1.1 DWIN protocol

Regarding DWIN protocol, please refer to the document: “T5L_DGUSII-Application-Development-Guide-V2.8” T5L_DGUSII Application Development Guide V2.8

Download from: https://www.dwin-global.com/development-guide/

The system debugging UART2 mode is fixed to 8N1, and the baud rate can be set. The data frame consists of 5 data blocks.

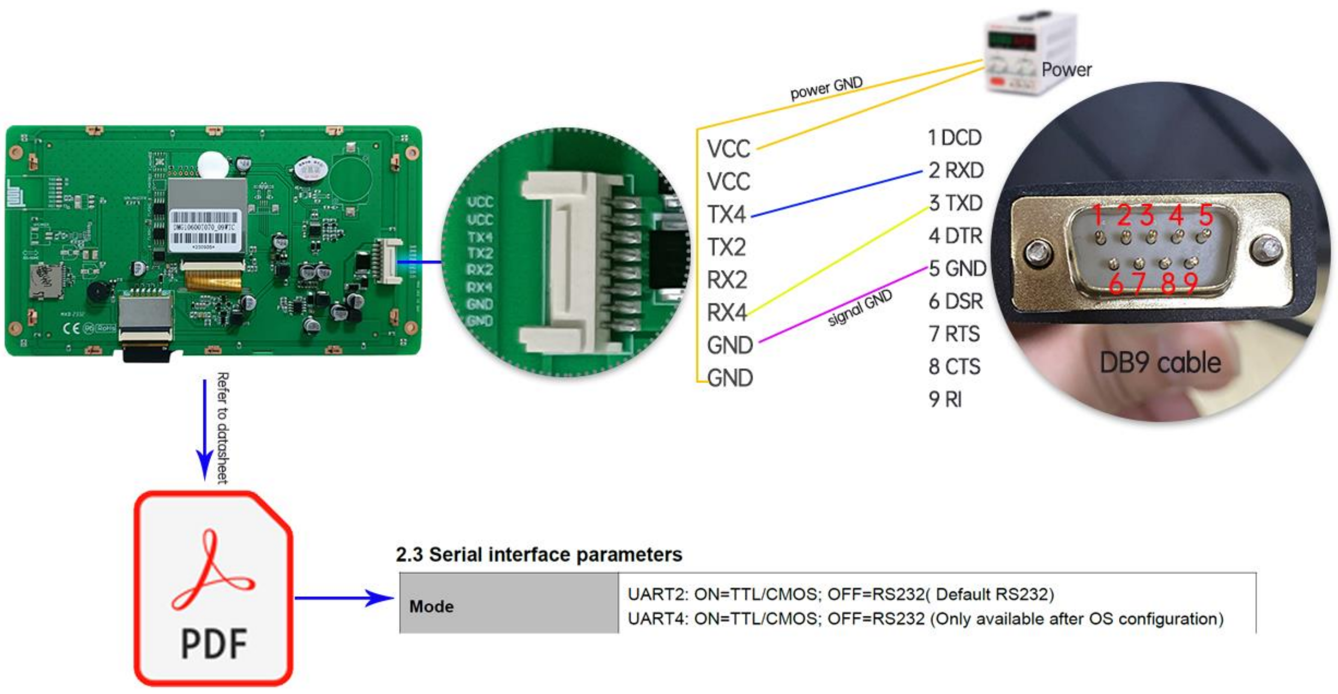
Data Block	1	2	3	4	5
Definition	Frame Header	Length	Instruction	Data	CRC Check (optional)
Length	2	1	1	Max.249	2
Description	0x5AA5	Including instruction, data and CRC check	0x82 Write 0x83 Read		CRC-16(x16+x15+x2+1)

1.2 UART example

To facilitate a rapid and convenient understanding of DWIN screen configurations, examples are provided for four screen types. These examples elucidate the identification of the available serial ports, specifying their types, and offering guidance on the wiring procedures.

LCD screen without shell: by referencing the datasheet, accurately determine the quantity and IDs of the UART. Whether it's UART 2 or UART 4, the wiring method remains consistent: the screen's RX corresponds to the TX, and the screen's TX corresponds to the RX. Following this, connect the GND (power GND & signal GND) and VCC to establish the connection.

Taking the diagram as an example, both UART 2 and UART 4 default to RS232 level. If connecting UART 4 to a computer, which also operates at RS232 level using a DB9 cable, following the definitions of the DB9's 9-pin connection as depicted in the diagram, using DuPont wires to connect RX4 to TX and TX4 to RX, then connect GND and power. After downloading and initializing the C51 code for Serial UART 4, it will facilitate communication betweenUART 4 and the computer.

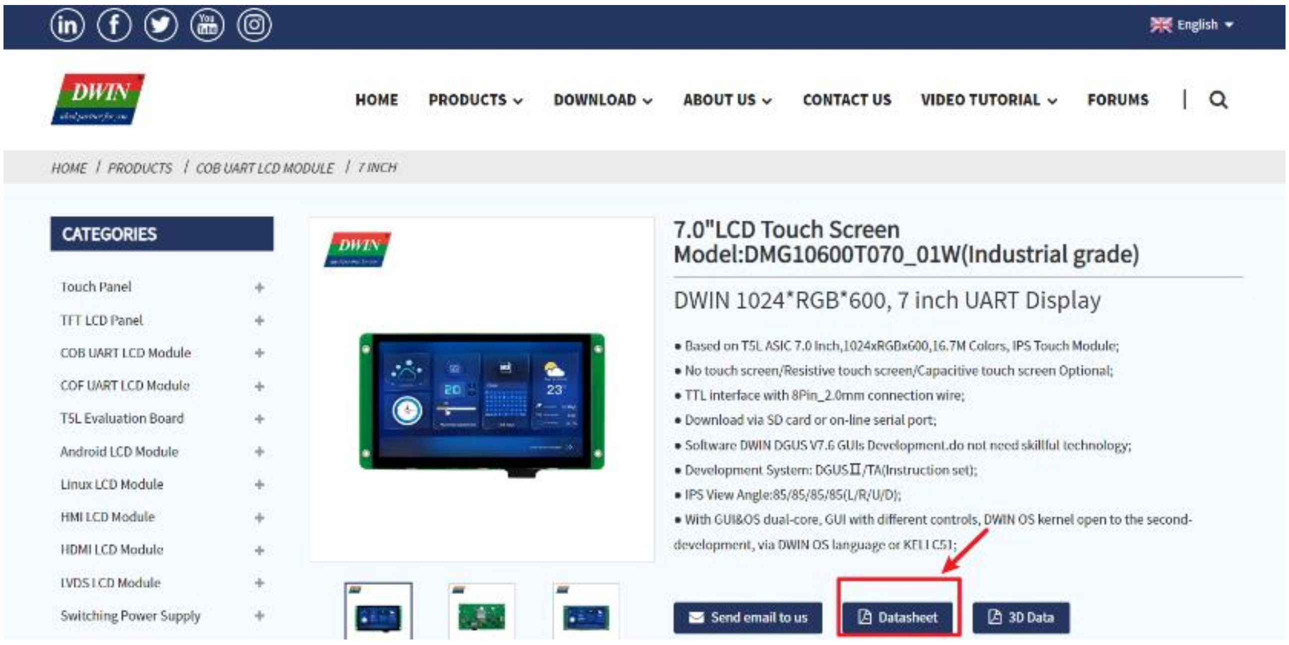


2.3 Serial interface parameters

Mode	UART2: ON=TTL/CMOS; OFF=RS232(Default RS232) UART4: ON=TTL/CMOS; OFF=RS232 (Only available after OS configuration)

Datasheet download from: [DWIN official website] – [Products] – datasheet under features

Official website link: www.dwin-global.com



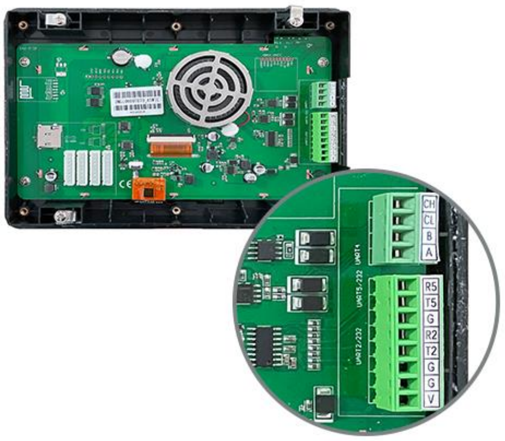
7.0"LCD Touch Screen Model:DMG10600T070_01W(Industrial grade)

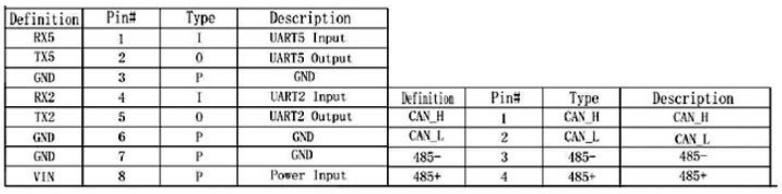
DWIN 1024*RGB*600, 7 inch UART Display

- Based on T5L ASIC 7.0 Inch,1024xRGBx600,16.7M Colors, IPS Touch Module;
- No touch screen/Resistive touch screen/Capacitive touch screen Optional;
- TTL interface with 8Pin_2.0mm connection wire;
- Download via SD card or on-line serial port;
- Software DWIN DGUS V7.6 GUIs Development.do not need skillful technology;
- Development System: DGUSⅡ/TA(Instruction set);
- IPS View Angle:85/85/85/85(L/R/U/D);
- With GUI&OS dual-core, GUI with different controls, DWIN OS kernel open to the second-development, via DWIN OS language or KEIL C51;

Screen with shell: similar to the above product without shell, refer to the datasheet to find the descriptions and definitions related to the UART ports.

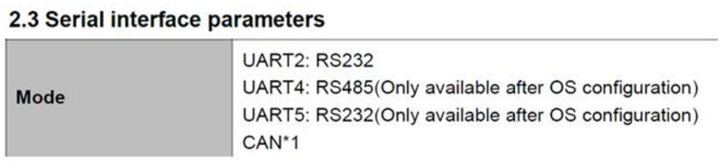
In the diagram, both UART 2 and UART 5 are operate at RS232 level (RX connect to TX, TX connect to RX, GND, VCC), while UART 4 operates at RS485 (B/485- connect to 485-, A/485+ connect to 485+).





Definition	Pin#	Type	Description
RX5	1	I	UART5 Input
TX5	2	O	UART5 Output
GND	3	P	GND
RX2	4	I	UART2 Input
TX2	5	O	UART2 Output
GND	6	P	GND
GND	7	P	GND
VIN	8	P	Power Input

Definition	Pin#	Type	Description
CAN_H	1	CAN_H	CAN_H
CAN_L	2	CAN_L	CAN_L
485-	3	485-	485-
485+	4	485+	485+

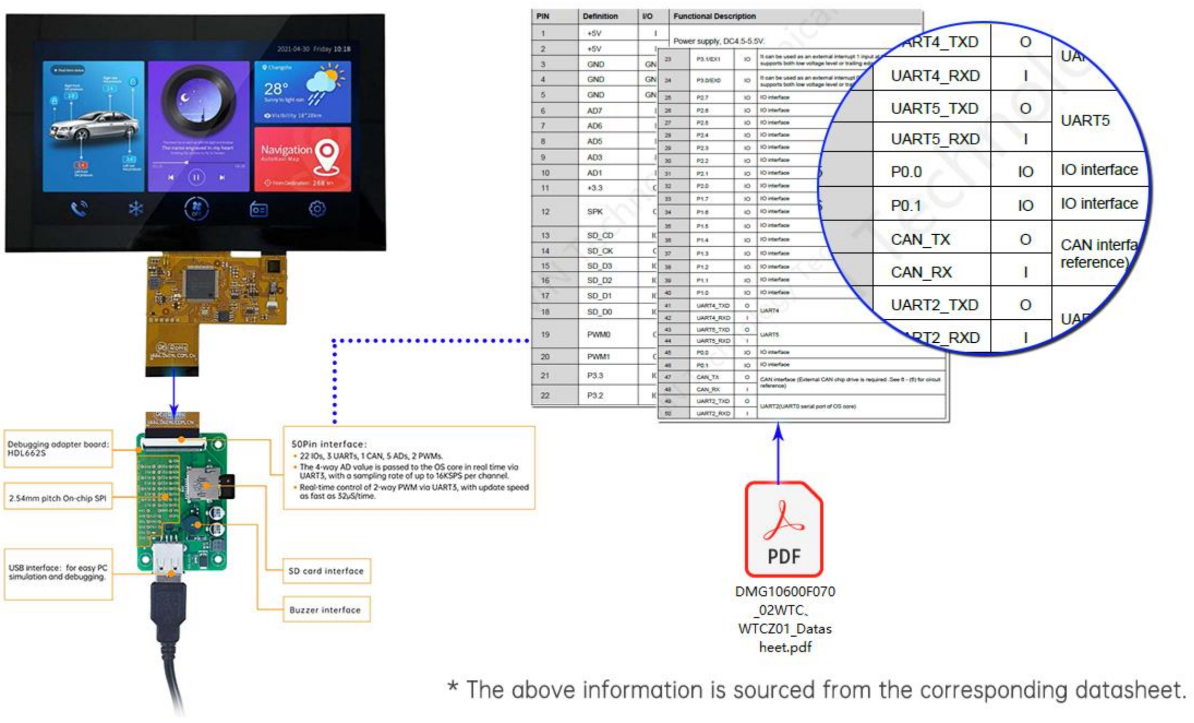


2.3 Serial interface parameters

Mode	UART2: RS232 UART4: RS485(Only available after OS configuration) UART5: RS232(Only available after OS configuration) CAN*1

* The above information is sourced from the corresponding datasheet. PDF DMG10600T070_A5WTC_Datasheet.pdf

COF screen: the user interface of COF screen differs from that of the COB structure, but you can find the corresponding interface definitions in the datasheet. Equipped with DWIN's HDL662S adapter board, it becomes remarkably convenience to utilize various UARTs and resources.


Debugging adapter board: HDL662S
2.54mm pitch On-chip SPI
USB interface: for easy PC simulation and debugging.
50Pin interface:
22 IOs, 3 UARTs, 1 CAN, 5 ADs, 2 PWMs.
The 4-way AD value is passed to the OS core in real time via UART3, with a sampling rate of up to 16KSPS per channel.
Real-time control of 2-way PWM via UART3, with update speed as fast as 32μS/time.
SD card interface
Buzzer interface
PDF
DMG10600F070_02WTC、WTCZ01_Datasheet.pdf

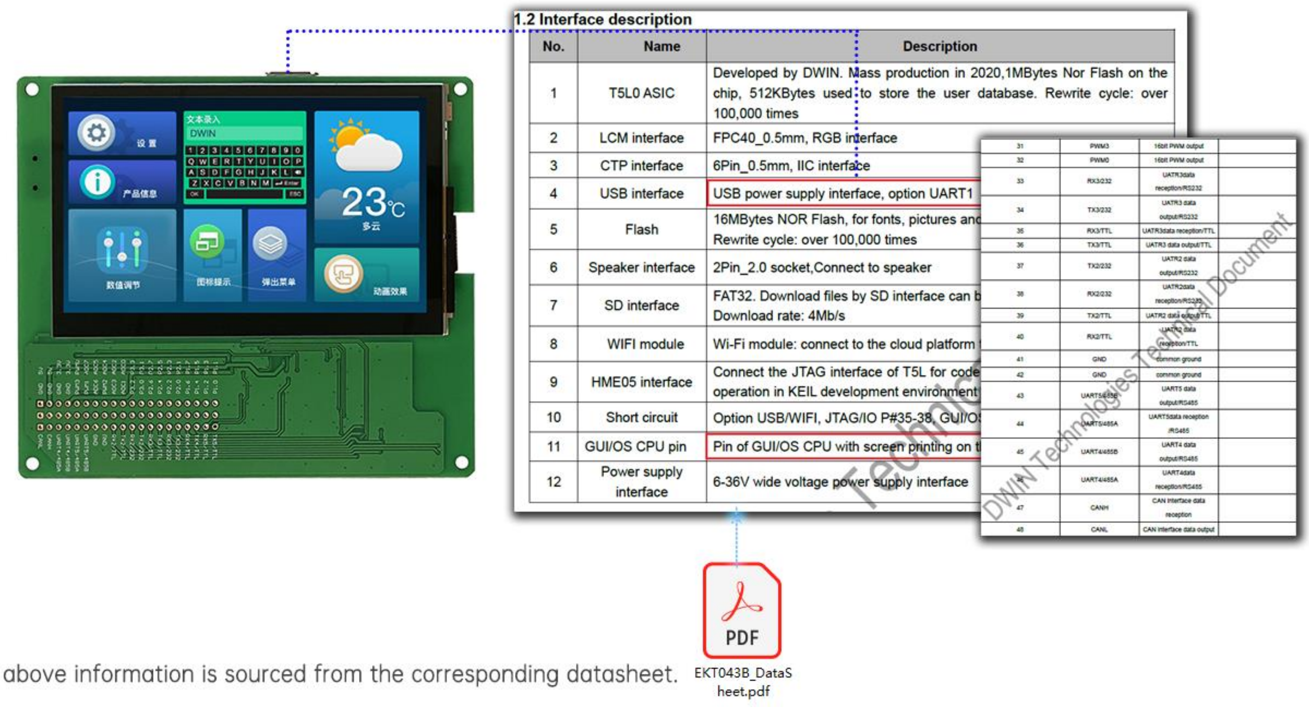
PIN	Definition	I/O	Functional Description
1	+5V	I	Power supply, DC4.5-5.5V.
2	+5V	I	
3	GND	GN	
4	GND	GN	
5	GND	GN	
6	AD7	I	
7	AD6	I	
8	AD5	I	
9	AD3	I	
10	AD1	I	
11	+3.3	O	
12	SPK	O	
13	SD_CD	IO	
14	SD_CK	O	
15	SD_D3	IO	
16	SD_D2	IO	
17	SD_D1	IO	
18	SD_D0	IO	
19	PWM0	O	
20	PWM1	O	
21	P3.3	IO	
22	P3.2	IO	

23	P3.1/EX1	IO	It can be used as an external interrupt 1 input and supports both low voltage level or trailing edge
24	P3.0/EX0	IO	It can be used as an external interrupt 0 supports both low voltage level or trailing
25	P2.7	IO	IO interface
26	P2.6	IO	IO interface
27	P2.5	IO	IO interface
28	P2.4	IO	IO interface
29	P2.3	IO	IO interface
30	P2.2	IO	IO interface
31	P2.1	IO	IO interface
32	P2.0	IO	IO interface
33	P1.7	IO	IO interface
34	P1.6	IO	IO interface
35	P1.5	IO	IO interface
36	P1.4	IO	IO interface
37	P1.3	IO	IO interface
38	P1.2	IO	IO interface
39	P1.1	IO	IO interface
40	P1.0	IO	IO interface
41	UART4_TXD	O	UART4
42	UART4_RXD	I	
43	UART5_TXD	O	UART5
44	UART5_RXD	I	
45	P0.0	IO	IO interface
46	P0.1	IO	IO interface
47	CAN_TX	O	CAN interface (External CAN chip drive is required .See 6 - (6) for circuit reference)
48	CAN_RX	I	
49	UART2_TXD	O	UART2(UART0 serial port of OS core)
50	UART2_RXD	I	

ART4_TXD	O	UA
UART4_RXD	I	
UART5_TXD	O	UART5
UART5_RXD	I	
P0.0	IO	IO interface
P0.1	IO	IO interface
CAN_TX	O	CAN interfa reference)
CAN_RX	I	
UART2_TXD	O	UA
RT2_RXD	I	

* The above information is sourced from the corresponding datasheet.

Evaluation board: this type of screen readily introduces various UARTs and resources, making it straightforward and easy to understand. USB port is UART 1 that is occupied by the GUI core, makes online debugging easy.



1.2 Interface description

No.	Name	Description
1	T5L0 ASIC	Developed by DWIN. Mass production in 2020,1MBytes Nor Flash on the chip, 512KBytes used to store the user database. Rewrite cycle: over 100,000 times
2	LCM interface	FPC40_0.5mm, RGB interface
3	CTP interface	6Pin_0.5mm, IIC interface
4	USB interface	USB power supply interface, option UART1
5	Flash	16MBytes NOR Flash, for fonts, pictures and Rewrite cycle: over 100,000 times
6	Speaker interface	2Pin_2.0 socket,Connect to speaker
7	SD interface	FAT32. Download files by SD interface can b Download rate: 4Mb/s
8	WIFI module	Wi-Fi module: connect to the cloud platform
9	HME05 interface	Connect the JTAG interface of T5L for code operation in KEIL development environment
10	Short circuit	Option USB/WIFI, JTAG/IO P#35-38, GUI/O
11	GUI/OS CPU pin	Pin of GUI/OS CPU with screen printing on t
12	Power supply interface	6-36V wide voltage power supply interface

31	PWM3	16bit PWM output	
32	PWM0	16bit PWM output	
33	RX3/232	UATR3data reception/RS232	
34	TX3/232	UATR3 data output/RS232	
35	RX3/TTL	UATR3data reception/TTL	
36	TX3/TTL	UATR3 data output/TTL	
37	TX2/232	UATR2 data output/RS232	
38	RX2/232	UATR2data reception/RS232	
39	TX2/TTL	UATR2 data output/TTL	
40	RX2/TTL	UATR2 data reception/TTL	
41	GND	common ground	
42	GND	common ground	
43	UART5/485B	UART5 data output/RS485	
44	UART5/485A	UART5data reception /RS485	
45	UART4/485B	UART4 data output/RS485	
46	UART4/485A	UART4data reception/RS485	
47	CANH	CAN interface data reception	
48	CANL	CAN interface data output	

* The above information is sourced from the corresponding datasheet.

2. UART 2

2.1 How to initial UART 2

Refer to the page 27 of "Development-Guide-of-T5L-ASIC20220413".

UART2 related SFR are shown in the following table.

SFR name	Address	Instructions
MUX_SEL	0xC9	.6 1 = UART2 interface leads to P 0.4 and P 0.5; 0 = UART02 interface does not lead out, it works as an IO port.
SCONO	0x98	UART2 control interface, the same as standard 8051, can be addressable by bit. .7=SM0 .6=SM1 .5=SM2(multiprocessor communication bit) .4=REN0 .3=TB80 .2=RB80 .1=TI0 .0=RI0.
SBUF0	0x99	UART2 transceiver data interface
ADCON	0xD8	Baud rate generator selection, 0x00 = T1 timer (standard 8051), 0x80 = SRELOH: L.
PCON	0x87	.7 SMOD baud rate frequency doubling selection. 0 = no frequency doubling 1 = frequency doubling.
SRELOH	0xBA	When ADCON = 0x80, SRELOH: L is used to set the baud rate without occupying T1. SMOD=0 SREOH: L=1024-CPU main frequency/ (64*baud rate) SMOD=1 SREOH: L=1024-CPU main frequency/ (32*baud rate) CPU main frequency = crystal frequency * 56/3, 11.0592 MHz crystal corresponds to 206.4384 MHz main

SRELOL	0xAA	frequency.

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART2 interrupt	0x0023	RIO(SCON0.0) TIO(SCON0.1)	IEN0.4	After interruption, software needs to clear the interruption trigger mark.

C51 code reference:

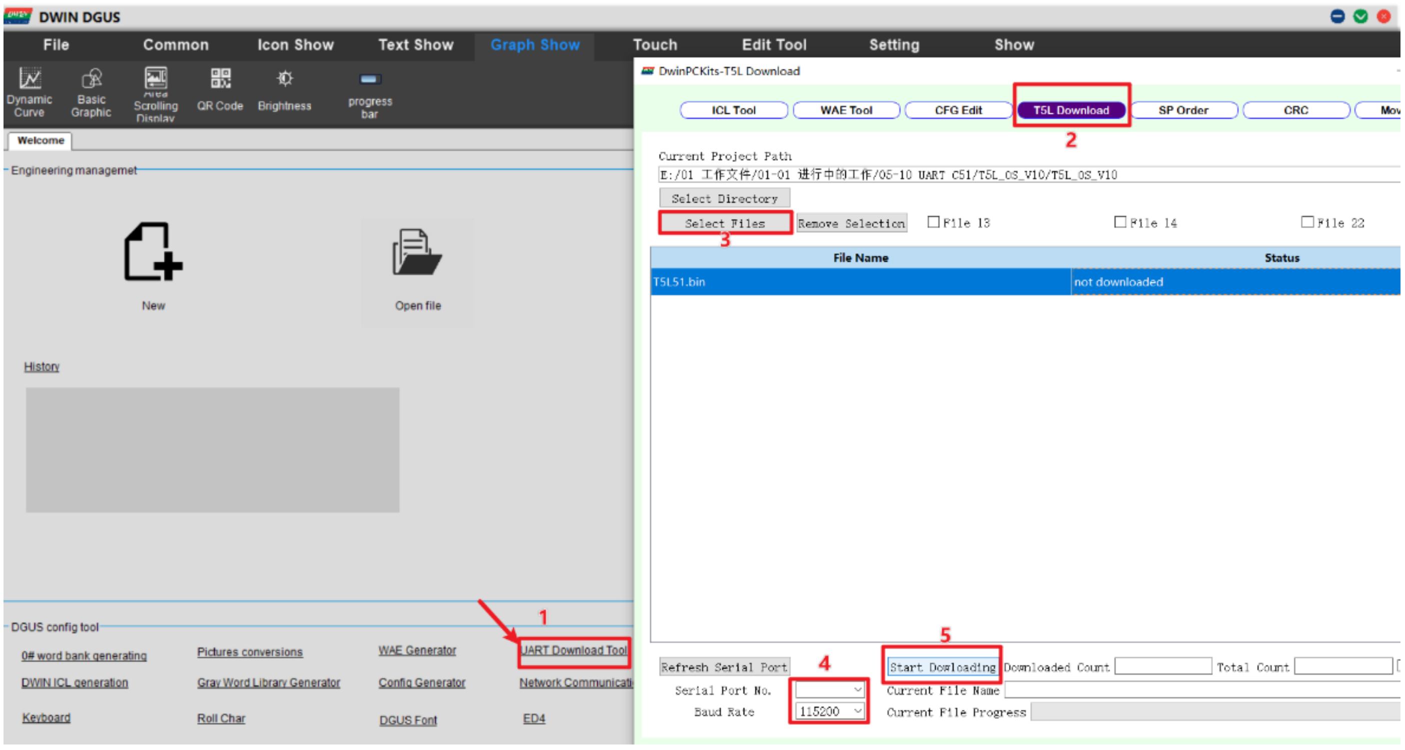
```
#if UART2_ENABLE
void Uart2_Init(u32 baud_rate){
    u16 i;

    Uart2.id = 2;
    Uart2.Tx_Read = 0;
    Uart2.Tx_Write = 0;
    Uart2.Tx_Busy = 0;
    Uart2.Rx_Read = 0;
    Uart2.Rx_Write = 0;
    Uart2.Rx_Busy = 0;
    Uart2.Rx_Flag = Uart_Rev_Pre;

    for(i = 0; i < Uart2_Tx_Lenth; i++) Uart2.Tx_Buffer[i] = 0;
    for(i = 0; i < Uart2_Rx_Lenth; i++) Uart2.Rx_Buffer[i] = 0;
  i=1024-FOSC/64/baud_rate;
    MUX_SEL |= 0x40;    //uart2  lead to
    P0MDOUT &= 0xCF;    //P0 0001 0000
    P0MDOUT |= 0x10;
    ADCON = 0x80;           //UART2 8N1 115200
    SCON0 = 0x50;
    SREL0H = (u8)(i>>8);      //Baud rate = FCLK/64*(1024-SREL)
    SREL0L = (u8)i;    // S_2/115200=224=0X03E4 1024-206438400/(64*115200)=0X03E4
    ES0 = 1;
    EA=1;
}
#endif
```

The compiled output is in HEX format, which needs to be converted to a bin format (like T5L51.bin) using our 'DownLoadFor8051' software, download link: https://www.dwin-global.com/tool-page/

Download code named as T5L51.bin by UART Download tool or SD card.



Current Project Path
E:/01 工作文件/01-01 进行中的工作/05-10 UART C51/T5L_OS_V10/T5L_OS_V10

File Name	Status
T5L51.bin	not downloaded

Serial Port No.
Baud Rate 115200

2.2 Loading UART 2 to send buffer

void Uart2_Tx_write2buff(u8 dat)

```
void Uart2_Tx_write2buff(u8 dat){
    Uart2.Tx_Buffer[Uart2.Tx_Write] = dat;                 //load send buffer

    if(Uart2.Tx_Read > Uart2.Tx_Write){                         // Buffer queue write pointer back, read pointer first
        while((Uart2.Tx_Write) + 1 == Uart2.Tx_Read);   //if the buffer is full, wait for the read pointer to move forward
    }

    ++Uart2.Tx_Write;                                             // write pointer forward
    if(Uart2.Tx_Write >= Uart2_Tx_Lenth){           // write pointer ready to return back
        while(Uart2.Tx_Read == 0);                          // if read pointer - head, wait move forward
        Uart2.Tx_Write = 0;                                     // write pointer return back
    }

    if(Uart2.Tx_Busy == 0){            //free
        Uart2.Tx_Busy = 1;                 //busy
        TI0 = 1;                                  // trigger send interrupt
    }
}
```

2.3 UART 2 data upload automatically

Regarding 83 & 83 commands, refer to the Page 44 of "T5L_DGUSII-Application-Development-Guide-V2.8"

T5L_DGUSII Application Development Guide V2.8

Instruction	Data	Description
0x83	Send: Variable space first address (0x0000-0xFFFF) + write data	Write data string (word data) to variable space starting from the specified address. Do not write the space reserved by the system.
	Answer: 0x4F 0x4B.	Write instruction answer.
0x83	Send: Variable space first address (0x0000- 0xFFFF) + byte length of the read data(0x01-0x7D)	Read word data of the specified length from the specified address of the variable space.
	Answer: Variable space first address + byte length of the variable data + the read variable data	The data is answered.

C51 code reference:

```
void Read_0xF00()
{
          u16  Va=Read_Dgus(0x0f00);
            u16  V1=Read_Dgus(0x0f01);
            if(((u8)(Va>>8))==0x5A)
            {      u8 i=0;u16 Temp=0;
                   u8 Val[100]={0};       //5A A5 06 83 ADDR Len XX XX
                     Val[0] = DTHD1;
               Val[1] = DTHD2;
                     Val[2] = (((u8)V1)<<1)+4;
               Val[3] = 0x83;
                   Val[4]=(u8)Va; Val[5]=(u8)(V1>>8);
                     Val[6]=(u8)V1;
                     for(i=0;i<(u8)V1;i++)
                    {
                          Temp=Read_Dgus(((Val[4]<<8)+Val[5]+i));
                          Val[7+2*i]=(u8)(Temp>>8);
                          Val[8+2*i]=(u8)(Temp);
                    }
#if UART2_ENABLE
            uart_data_send(Val,2,DATA_UPLOAD_UART2,CRC_CHECK_UART2);
#endif
#if UART3_ENABLE
          uart_data_send(Val,3,DATA_UPLOAD_UART3,CRC_CHECK_UART3);
#endif
#if UART4_ENABLE
                    uart_data_send(Val,4,DATA_UPLOAD_UART4,CRC_CHECK_UART4);
#endif
#if UART5_ENABLE
          uart_data_send(Val,5,DATA_UPLOAD_UART5,CRC_CHECK_UART5);
#endif
                    Write_Dgus(0x0f00,0);Write_Dgus(0x0f01,0);
            }
}



/*****************************************************************************/
void  deal_82_cmd(u8 Uart,u8* arr)
{
       u8 i=0;
     if(Crc_check_flog==0) //without CRC check
     {
                  if(arr[4]==0&&arr[5]==6) //upgrade c code
                      {
                           arr[7]=0xA5;
                      }
            Write_Dgusii_Vp_byChar((arr[4]<<8)+arr[5],arr+6,arr[2]-3);
            if(Response_flog)
            {
                 u8 Temp_arr[]={DTHD1,DTHD2,0X03,0X82,0X4F,0X4B};
                 uart_send_str(Uart,Temp_arr,6);
            }
      }else //with CRC check
      {
           u16 Crc=0,Crc_check=0;
           Crc=crc16table((u8*)(&arr[3]),arr[2]-2);
           Crc_check=(u16)(arr[3+arr[2]-1]<<8)+(u16)(arr[3+arr[2]-2]);
           if(Crc==Crc_check)
           {
                Write_Dgusii_Vp_byChar((arr[4]<<8)+arr[5],arr+6,arr[2]-
5);
                if(Response_flog)
                {
                   u8 Temp_arr[]={DTHD1,DTHD2,0X05,0X82,0X4F,0X4B,0XA5,0XEF};
                     uart_send_str(Uart,Temp_arr,8);
                }
           }
             }
}
/*****************************************************************************/
void  deal_83_cmd(u8 Uart,u8* arr,u8* arr1)
{
         u8 i=0;
       if(Crc_check_flog==0) //without CRC check
            {
              for(i=0;i<7;i++)
                                   arr[i]=arr1[i];
                        Read_Dgusii_Vp((arr[4]<<8)+arr[5],(u8*)&arr[7],arr[6]);
                        arr[2]=(2*arr[6])+4;
            uart_send_str(Uart,arr,arr[2]+3);
           }else   //with Crc check
          {
                        u16 Crc=0,Crc_check=0;
                        for(i=0;i<9;i++)
                                   arr[i]=arr1[i];
                        Crc=crc16table((u8*)(&arr[3]),arr[2]-2);
                        Crc_check=(u16)(arr[3+arr[2]-1]<<8)+(u16)(arr[3+arr[2]-2]);
                     if(Crc==Crc_check)
                       {
                              Read_Dgusii_Vp((arr[4]<<8)+arr[5],(u8*)&arr[7],arr[6]);
                              arr[2]=(2*arr[6])+4+2;
                              Crc=crc16table(arr+3,arr[2]-2);
                              arr[arr[2]+1]=Crc& 0x00FF;
                     arr[arr[2]+2]=Crc>> 8;
                   uart_send_str(Uart,arr,arr[2]+3);
                       }
             }
}
```

2.4 UART2 interrupt receiving function

The relevant settings for UART2 interruption refer to the page 27 of "Development-Guide-of-T5L-ASIC20220413".

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART2 interrupt	0x0023	RIO(SCON0.0) TIO(SCON0.1)	IEN0.4	After interruption, software needs to clear the interruption trigger mark.

C51 code reference:

```
void Uart2_TX_RX_ISR_PC(void) interrupt 4
{
    u8 res = 0;
    EA = 0;
    if(RI0 == 1){
        RI0 = 0;
            Uart2.Rx_Buffer[Uart2.Rx_Write] = SBUF0;
            Uart2.Rx_Write++;
            Uart2.Rx_Write %= Uart2_Rx_Lenth;

    }else if(TI0 == 1){
        TI0 = 0;
        if(Uart2.Tx_Read != Uart2.Tx_Write){
            SBUF0 = Uart2.Tx_Buffer[Uart2.Tx_Read];
            Uart2.Tx_Read++;
            Uart2.Tx_Read %= Uart2_Tx_Lenth;
        }else{
            Uart2.Tx_Busy = 0;
        }
    }
    EA = 1;
}
```

2.4.1 Interrupt control SFR

Refer to the page 33 of “Development-Guide-of-T5L-ASIC20220413”.

T5L OS CPU has 12 interrupts. The related control SFRs list is as follows:

SFR name	Address	Instructions
IEN0	0xA8	The interrupt enable controller 0 can be addressed by bit. .7 Interrupt master control bit. 0=all interrupts closed; 1=whether an interrupt is opened is controlled by the control bit of each interrupt; .6 Must write 0; .5 ET2 T2 timer interrupt enable control bit; .4 ES0 UART2 interrupt enable control bit; .3 ET1 T1 timer interrupt enable control bit; .2 EX1 external interrupt 1 (P3.1 pin) interrupt enabling control bit; .1 ET0 T0 timer interrupt enable control bit; .0 EX0 external interrupt 1 (P3.0 pin) interrupt enabling control bit.
IEN1	0xB8	The interrupt enable controller 1 can be addressed by bit. .7-.6 Write 0; .5 ES3R UART5 receiving interrupt enabled control bit; .4 ES3T UART5 receiving interrupt enabled control bit; .3 ES2R UART4 receiving interrupt enabled control bit; .2 ES2R UART4 receiving interrupt enabled control bit; .1 ECAN CAN communication interrupt enabling control bit; .0 Write 0.
IEN2	0x9A	Interrupt enabling controller 2 .7-.1 Must write 0 .0 ESI USRT3 interrupt enabling control bit
IEN3	0xD1	Interrupt enabling controller 3, must write 0x00
IP0	0xA9	Interrupt priority controller 0
IP1	0xB9	Interrupt priority controller 1

2.4.2 Interrupt priority

The interrupt priority of T5L OS CPU is handled according to the following rules.

- Twelve interrupts are divided into six groups with two interrupts in each group. The priority in the same group is fixed. For example, the priority of interrupt 0 is higher than that of UART3, as shown in the table below.

Grouping	IP0 correspondence	IP1 correspondence	Interrupt correspondence	
			High priority	Low priority
G0	.0	.0	External interrupt 0	UART3 interrupt
G1	.1	.1	T0 timer interrupt	CAN communication interrupt
G2	.2	.2	External interrupt 1	UART4 send interrupt
G3	.3	.3	T1 timer interrupt	UART4 receive interrupt
G4	.4	.4	UART2 interrupt	UART5 send interrupt
G5	.5	.5	T2 timer interrupt	UART5 receive interrupt

- There are four levels of priority among the six groups, which can be configured by the corresponding bits of IP0 and IP1 according to the table below.

Inter group priority	IP1 counterpart	IP0 counterpart
3(highest)	1	1
2	1	0
1	0	1
0(lowest)	0	0

For example, if you want to set the G2 group's T2 timer interrupt and UART5 receive interrupt priority to the highest, you can set 1P1=0x20, 1P0=0x20.

- If the configurations have the same priority (IP1 = 0x00 IP0 = 0x00), the G0 group has the highest priority and the G5 group has the lowest priority. The interrupt priority weights with the same configuration are as follows:

Weight	11	10	9	8	7	6	5	4	3	2	1	0

Priority	Maximum											Minimum
Interrupt	EX0	UART3	ET0	CAN	EX1	UART4-TX	ET1	UART4-RX	UART2	UART5-TX	ET2	UART5-RX

- High priority interrupts can be embedded in low priority interrupts (i.e. interrupts with small weights can be interrupted by interrupts with large weights), and at most four levels can be embedded.

Note:

The T5L OS CPU is fast (1uS can execute 130-150 instructions on average) and the interrupt execution time is short, so the real-time performance is already very high.

It is not recommended to use interrupt embedding that makes program architecture more complex.

Users can directly turn off interrupts (EA=0) when each interrupt service program is executed, and turn on interrupts (EA=1) when exiting.

3. UART 3

3.1 How to initial UART 3

Refer to the page 28 of "Development-Guide-of-T5L-ASIC20220413".

UART3 related SFRs are shown in the following table.

SFR name	Address	Instructions
MUX_SEL	0xC9	.5 1 = UART3 interface leads to P 0.6, P 0.7, 0 = UART3 interface does not lead out, it is IO port.
SCON1	0x9B	UART3 control interface, it is not addressable by bit. .7 0=9bit UART; 1=8bit UART; .6 Undefined; .5=SM2(multiprocessor communication bit) .4=REN .3=TB8 .2=RB8 .1=TI .0=RI. Clearing the SCON1 bit mark requires two consecutive writings, such as ANL SCON1,#0FEH ANL SCON1,#0FEH
SBUF1	0x9C	UART3 transceiver data interface
SREL1H	0xBB	UART3 baud rate setting (CPU main frequency = crystal frequency * 56/3, 11.0592 crystal corresponding to 206.4384 MHz main frequency): SRE1H:L=1024-CPU main frequency/(32*baud rate)
SREL1L	0x9D	

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART3 interrupt	0x0083	SCON1.0,SCON1.1	IEN2.0	After interruption, software needs to clear the interruption trigger mark.

C51 code reference:

```
#if UART3_ENABLE
void Uart3_Init(u32 baud_rate){
        u16 i=0;
    Uart3.id = 2;
    Uart3.Tx_Read = 0;
    Uart3.Tx_Write = 0;
    Uart3.Tx_Busy = 0;
    Uart3.Rx_Read = 0;
    Uart3.Rx_Write = 0;
    Uart3.Rx_Busy = 0;
    Uart3.Rx_Flag = Uart_Rev_Pre;

    for(i = 0; i < Uart3_Tx_Lenth; i++) Uart3.Tx_Buffer[i] = 0;
    for(i = 0; i < Uart3_Rx_Lenth; i++) Uart3.Rx_Buffer[i] = 0;
      i=1024-FOSC/32/baud_rate;
        SREL1H = (u8)(i>>8);      //Baud rate = FCLK/64*(1024-SREL)
        SREL1L = (u8)i;    // S_2/115200=224=0X03C8 1024-206438400/(32*115200)=0X03C8
            MUX_SEL|=0x20;    //UART3 lead to
            P0MDOUT|=0x40;    //P0.6 TXD push pull
            SCON1 = 0x90;     //receive enable and mode settings
            IEN2 |= 0x01;     // interrupt enable
        EA  = 1;
}
#endif
```

3.2 Loading UART 3 to send buffer

void Uart3_Tx_write3buff(u8 dat)

```
void Uart3_Tx_write3buff(u8 dat){
    Uart3.Tx_Buffer[Uart3.Tx_Write] = dat;                  // load sending buffer

    if(Uart3.Tx_Read > Uart3.Tx_Write){                     // Buffer queue write pointer back, read pointer first
        while((Uart3.Tx_Write) + 1 == Uart3.Tx_Read);   // if the buffer is full, wait for the read pointer to move forward
    }

    ++Uart3.Tx_Write;                                        // write pointer forward
    if(Uart3.Tx_Write >= Uart3_Tx_Lenth){                   // write pointer is ready to return back
        while(Uart3.Tx_Read == 0);                           // if read pointer - head, wait move forward
        Uart3.Tx_Write = 0;                                  // write pointer return back
    }

    if(Uart3.Tx_Busy == 0){             //free
        Uart3.Tx_Busy = 1;              //busy
        SCON1|=0x02 ;                   // trigger send interrupt
    }
}
```

3.3 UART3 interrupt receiving function

Regarding the interrupt control SFR & priority refer to the chapter 2.4.1. & chapter 2.4.2.

The relevant settings for UART3 interruption refer to the page 38 of “Development-Guide-of-T5L-ASIC30330413”.

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART3 interrupt	0x0083	SCON1.0,SCON1.1	IEN3.0	After interruption, software needs to clear the interruption trigger mark.

C51 code reference:

```
void Uart3_RX_ISR_PC(void)      interrupt 16   //UART 3 receive & send interrupt
{
       if(SCON1&0x01)
           {

                   SCON1&=0xFE;
//                     SCON1&=0xFE;
                Uart3.Rx_Buffer[Uart3.Rx_Write] = SBUF1;
                Uart3.Rx_Write++;
                Uart3.Rx_Write %= Uart3_Rx_Lenth;
           }else if(SCON1&0x02)
           {
                   SCON1&=0xFD ;
                     SCON1&=0xFD ;
                     if(Uart3.Tx_Read != Uart3.Tx_Write){
                 SBUF1 = Uart3.Tx_Buffer[Uart3.Tx_Read];
                 Uart3.Tx_Read++;
                 Uart3.Tx_Read %= Uart3_Tx_Lenth;
                }else{
                     Uart3.Tx_Busy = 0;
                }
           }
}
```

4. UART 4

4.1 How to initial UART 4

Refer to the page 38 of "Development-Guide-of-T5L-ASIC30330413".

UART4 related SFR are shown in the following table.

SFR name	Address	Instructions

SCON3T	0x96	UART4 sending control:
		.7 UART4 sending enable. 0=close;1=open;
		.6 0=8bit mode,1=9bit mode;
		.5 TB8, 9th bit sent in 9bit mode;
		.4-.1 Write0;
		.0 TI, send mark. The position at which the stop bit is sent.

SCON3R	0x97	UART4 receive control:
		.7 UART4 sending enable. 0=close;1=open;
		.6 Write0;
		.5 RB8, 9th bit received in 9bit mode;
		.4-.1 Write 0;
		.0 R RI, receive mark. Set when the stop bit is received when a valid stop bit is received.

SBUF3_TX	0x9E	UART4 sending data interface
SBUF3_RX	0x9F	UART4 receiving data interface
BODE3_DIV_H	0XD9	UART4 baud rate setting BODE3_DIV_H:L=CPU main frequency/(8*baud rate)
BODE3_DIV_L	0XD7	

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART4 send interrupt	0x0053	SCON3T.0	IEN1.3	After interruption, clear the interruption trigger mark by software.
UART4 receive Interrupt	0x005B	SCON3R.0	IEN1.3	After interruption, clear the interruption trigger mark by software.

C51 code reference:

```
#if UART4_ENABLE
void Uart4_Init(u32 baud_rate){
    u16 i;

    Uart4.id = 2;
    Uart4.Tx_Read = 0;
    Uart4.Tx_Write = 0;
    Uart4.Tx_Busy = 0;
    Uart4.Rx_Read = 0;
    Uart4.Rx_Write = 0;
    Uart4.Rx_Busy = 0;
    Uart4.Rx_Flag = Uart_Rev_Pre;

    for(i = 0; i < Uart4_Tx_Lenth; i++) Uart4.Tx_Buffer[i] = 0;
    for(i = 0; i < Uart4_Rx_Lenth; i++) Uart4.Rx_Buffer[i] = 0;
    P0MDOUT |=0x03;
  i=FOSC/8/baud_rate;
    SCON2T= 0x80    ;// send enable and mode settings
    SCON2R= 0x80;// receive enable and mode settings
    ES2R = 1;// interrupt receive enable
  ES2T = 1;// interrupt send enable
    BODE2_DIV_H = (u8)(i>>8);     //Baud rate = CPU dominant frequency ÷(8*baud rate)
    BODE2_DIV_L = (u8)i;    //
            P0MDOUT|=(1<<0); //p0^0 push-pull
            TR4=0;
   EA=1;
}
#endif
```

4.2 Loading UART 4 to send buffer

void Uart4_Tx_write4buff(u8 dat)

```
void Uart4_Tx_write4buff(u8 dat){
    Uart4.Tx_Buffer[Uart4.Tx_Write] = dat;                  //load sending buffer

    if(Uart4.Tx_Read > Uart4.Tx_Write){                          // Buffer queue write pointer back, read pointer first
        while((Uart4.Tx_Write) + 1 == Uart4.Tx_Read);   // if the buffer is full, wait for the read pointer to move forward
    }

    ++Uart4.Tx_Write;                                                   // write pointer forward
    if(Uart4.Tx_Write >= Uart4_Tx_Lenth){            // write pointer is ready to return back
        while(Uart4.Tx_Read == 0);                          // if read pointer - head, wait move forward
        Uart4.Tx_Write = 0;                                     // write pointer return back
    }

    if(Uart4.Tx_Busy == 0){             //free
        Uart4.Tx_Busy = 1;                  //busy
        TR4=1;          //   485 enable sending
        SCON2T|=0x1 ;                                   // trigger send interrupt
    }
}
```

4.3 UART4 interrupt receiving function

Regarding the interrupt control SFR & priority refer to the chapter 2.4.1. & chapter 2.4.2.

The relevant settings of UART4 interruption refer to the page 38 of “Development-Guide-of-T5L-ASIC30330413”.

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART4 send interrupt	0x0053	SCON3T.0	IEN1.3	After interruption, clear the interruption trigger mark by software.
UART4 receive Interrupt	0x005B	SCON3R.0	IEN1.3	After interruption, clear the interruption trigger mark by software.

C51 code reference:

```
void Uart4_RX_ISR_PC(void) interrupt 11
{
    u8 res = 0;
    EA = 0;
    SCON2R&=0xFE;
            Uart4.Rx_Buffer[Uart4.Rx_Write] = SBUF2_RX;
            Uart4.Rx_Write++;
            Uart4.Rx_Write %= Uart4_Rx_Lenth;


    EA = 1;
}
```

4.4 UART4 interrupt sending function

Regarding the interrupt control SFR & priority refer to the chapter 2.4.1. & chapter 2.4.2.

The relevant settings of UART4 interruption refer to the page 38 of "Development-Guide-of-T5L-ASIC30330413".

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART4 send interrupt	0x0053	SCON3T.0	IEN1.3	After interruption, clear the interruption trigger mark by software.
UART4 receive Interrupt	0x005B	SCON3R.0	IEN1.3	After interruption, clear the interruption trigger mark by software.

C51 code reference:

```c
void Uart4_TX_ISR_PC(void) interrupt 10
{
    u8 res = 0;
    EA = 0;
        SCON2T&=0xFE ;
        if(Uart4.Tx_Read != Uart4.Tx_Write){
            SBUF2_TX = Uart4.Tx_Buffer[Uart4.Tx_Read];
            Uart4.Tx_Read++;
            Uart4.Tx_Read %= Uart4_Tx_Lenth;
        }else{
            TR4=0;       //485 enable send
            Uart4.Tx_Busy = 0;
        }

    EA = 1;
}
```

5. UART 5

5.1 How to initial UART 5

Refer to the page 30 of "Development-Guide-of-T5L-ASIC30330413".

The relevant settings of UART5 are as follows:

SFR name	Address	Instructions

SCON3T	0xA7	UART5 sending control:
		.7 UART5 sending enables. 0=close;1=open;
		.6 0=8bit mode,1=9bit mode;
		.5 TB8, 9th bit sent in 9bit mode;
		.4-.1 Write 0;
		.0 TI, send flag. The position at which the stop bit is sent.

SCON3R	0xAB	UART5 receive control:
		.7 UART5 sending enables. 0=close;1=open;
		.6 Write 0;
		.5 TB8, 9th bit received in 9bit mode;
		.4-.1 Write 0;
		.0 R RI, receive mark. Set when the stop bit is received when a valid stop bit is received.

SBUF3_TX	0xAC	UART5 sending data interface
SBUF4_RX	0xAD	UART5 receiving data interface
BODE3_DIV_H	0xAE	UART5 baud rate setting BODE3_DIV_H: L=CPU main frequency/ (8*baud rate)

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART5 send interrupt	0x0063	SCON3T.0	IEN1.4	After interruption, clear the interruption trigger mark by software.
UART5 receive Interrupt	0x006B	SCON3R.0	IEN1.5	After interruption, clear the interruption trigger mark by software.

C51 code reference:

```
#if UART5_ENABLE
void Uart5_Init(u32 baud_rate){
    u16 i;

    Uart5.id = 2;
    Uart5.Tx_Read = 0;
    Uart5.Tx_Write = 0;
    Uart5.Tx_Busy = 0;
    Uart5.Rx_Read = 0;
    Uart5.Rx_Write = 0;
    Uart5.Rx_Busy = 0;
    Uart5.Rx_Flag = Uart_Rev_Pre;

    for(i = 0; i < Uart5_Tx_Lenth; i++) Uart5.Tx_Buffer[i] = 0;
    for(i = 0; i < Uart5_Rx_Lenth; i++) Uart5.Rx_Buffer[i] = 0;
    P0MDOUT |=0x03;
  i=FOSC/8/baud_rate;
    SCON3T= 0x80    ;// send enable and mode settings
    SCON3R= 0x80;// receive enable and mode settings
    ES3R = 1;// interrupt receive enable
  ES3T = 1;// interrupt send enable
    BODE3_DIV_H = (u8)(i>>8);     // baud rate = CPU dominant frequency ÷(8*baud rate)
    BODE3_DIV_L = (u8)i;    //
            P0MDOUT|=(1<<1); //p0^1 push-pull
            TR5=0;
  EA=1;
}
#endif
```

5.2 Loading UART 5 to send buffer

void Uart5_Tx_write5buff(u8 dat)

```
void Uart5_Tx_write5buff(u8 dat){
    Uart5.Tx_Buffer[Uart5.Tx_Write] = dat;                 //load sending buffer

    if(Uart5.Tx_Read > Uart5.Tx_Write){                        // Buffer queue write pointer back, read pointer first
        while((Uart5.Tx_Write) + 1 == Uart5.Tx_Read);   // if the buffer is full, wait for the read pointer to move forward
    }

    ++Uart5.Tx_Write;                                               // write pointer forward
    if(Uart5.Tx_Write >= Uart5_Tx_Lenth){            // write pointer is ready to return back
        while(Uart5.Tx_Read == 0);                          // if read pointer - head, wait move forward
        Uart5.Tx_Write = 0;                                    // write pointer return back
    }

    if(Uart5.Tx_Busy == 0){            //free
        Uart5.Tx_Busy = 1;                 //busy
        TR5=1;          //   485 enable send
        SCON3T|=0x1 ;                                    // trigger send interrupt
    }
}
```

5.3 UART5 interrupt receiving function

Regarding the interrupt control SFR & priority refer to the chapter 2.4.1. & chapter 2.4.2.

The relevant settings of UART5 interruption refer to the page 30 of "Development-Guide-of-T5L-ASIC30330413".

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART5 send interrupt	0x0063	SCON3T.0	IEN1.4	After interruption, clear the interruption trigger mark by software.
UART5 receive Interrupt	0x006B	SCON3R.0	IEN1.5	After interruption, clear the interruption trigger mark by software.

C51 code reference:

```c
void Uart5_RX_ISR_PC(void) interrupt 13
{
    u8 res = 0;
    EA = 0;
    SCON3R&=0xFE;
            Uart5.Rx_Buffer[Uart5.Rx_Write] = SBUF3_RX;
            Uart5.Rx_Write++;
            Uart5.Rx_Write %= Uart5_Rx_Lenth;


    EA = 1;
}
```

5.4 UART5 interrupt sending function

Regarding the interrupt control SFR & priority refer to the chapter 2.4.1. & chapter 2.4.2.

The relevant settings of UART5 interruption refer to the page 30 of "Development-Guide-of-T5L-ASIC30330413".

Interrupt type	Program entry address	Trigger marker	Interrupt enabling control	Remarks
UART5 send interrupt	0x0063	SCON3T.0	IEN1.4	After interruption, clear the interruption trigger mark by software.
UART5 receive Interrupt	0x006B	SCON3R.0	IEN1.5	After interruption, clear the interruption trigger mark by software.

C51 code reference:

```
void Uart5_TX_ISR_PC(void) interrupt 12
{
    u8 res = 0;
    EA = 0;
        SCON3T&=0xFE ;
        if(Uart5.Tx_Read != Uart5.Tx_Write){
            SBUF3_TX = Uart5.Tx_Buffer[Uart5.Tx_Read];
            Uart5.Tx_Read++;
            Uart5.Tx_Read %= Uart5_Tx_Lenth;
        }else{
            TR5=0;      //485 close sending
            Uart5.Tx_Busy = 0;
        }

    EA = 1;
}
```

6. Data upload automatically

```
void uart_data_send(u8* Arr,u8 Uartnum,bit Outo_send,bit Crc_ck )
{
            u8   i=0;
            u16  V1=Read_Dgus(0x0f01);
          Auto_data_upload=Outo_send;
                    if(Auto_data_upload)
                    {
                         Crc_check_flog=Crc_ck;
                         if(Crc_check_flog) // with Crc check
                         {
                               u16 Crc=0;
                               Arr[2] = (((u8)V1)<<1)+6;
                             Crc=crc16table(Arr+3,Arr[2]-2);
                     Arr[Arr[2]+1]=Crc& 0x00FF;
                     Arr[Arr[2]+2]=Crc>> 8;
                uart_send_str(Uartnum,Arr,Arr[2]+3);
                               Arr[2] = (((u8)V1)<<1)+4;
                         }else
             {
                  uart_send_str(Uartnum,Arr,Arr[2]+3);
                         }
                    }
}
```

7. Deal UART data

(u8* Arr_rece,u16* Data_len,u8 Uart_num,bit Response,bit Crc_ck)

Arr_rece: UART receive array

Uart_num: UART number

Response: with response or not

Crc_ck: with CRC check or not

Regarding 83 & 83 commands, refer to the Page 44 of "T5L_DGUSII-Application-Development-Guide-V2.8"

T5L_DGUSII Application Development Guide V2.8

Instruction	Data	Description
0x83	Send: Variable space first address (0x0000-0xFFFF) + write data	Write data string (word data) to variable space starting from the specified address. Do not write the space reserved by the system.
	Answer: 0x4F 0x4B.	Write instruction answer.
0x83	Send: Variable space first address (0x0000- 0xFFFF) + byte length of the read data(0x01-0x7D)	Read word data of the specified length from the specified address of the variable space.
	Answer: Variable space first address + byte length of the variable data + the read variable data	The data is answered.

C51 code reference:

```
void  deal_uart_data(u8* Arr_rece,u8 Uart_num,bit Response,bit Crc_ck)
{
            u16 N=0;    bit Flog=1;

                    if((Arr_rece[0]==DTHD1)&&(Arr_rece[1]==DTHD2))   //5A A5 07 82 1000 0001 0002
                    {        if(Arr_rece[3]==0x82)
                    {
                            Response_flog=Response;//response note
                            Crc_check_flog=Crc_ck;//Crc note
               deal_82_cmd(Uart_num,Arr_rece);//handle 82 command

                    }                    // 0  1  2  3  4 5  6  7 8  9 10
                    else if(Arr_rece[3]==0x83)        //5A  A5 08 83 1000 02 0001 0002
                    {

                            u8 Val[Uartx_Frame_Data_Lenth]={0};
                             Crc_check_flog=Crc_ck;//Crc note
               deal_83_cmd(Uart_num,Val,Arr_rece);//handle 83 command
                    }
               }

}
```

8. UART data handle

void Uart_Handle_Frame(struct Uartx_Define Uart_number,struct Uartx_Frame_Data Uart_data,void (*pSendFuc)(u8),u16 Uart_buff_lenth)

void (*pSendFuc)(u8): UART send function

```
void Uart_Handle_Frame(P_S_UART Uart_number,P_D_UART Uart_data,u8 Uart_num,u16 Uart_buff_lenth,bit Response,bit Crc_ck) {
    u8  c;
    u16 i=0,Crc=0;

    while( ((*Uart_number).Rx_Write - (*Uart_number).Rx_Read + Uart_buff_lenth) % Uart_buff_lenth){
        EA = 0;
        c = (*Uart_number).Rx_Buffer[(*Uart_number).Rx_Read++];        //read data
        (*Uart_number).Rx_Read %= Uart_buff_lenth;
        (*Uart_number).Rx_Busy = 0;
        EA = 1;
        if((*Uart_number).Rx_Flag == Uart_Rev_Pre){                    // start to receive frame data
            (*Uart_data).dataLen = 0;
            (*Uart_data).dataCode = 0x00;
            (*Uart_data).varAddr = 0x0000;
            (*Uart_data).varIndex = 0;
            if(c == DTHD1)
            {
                (*Uart_number).Rx_Flag = Uart_Rev_GetFH1;
                (*Uart_data).varData[(*Uart_data).varIndex++] = c;
            }
        }else if((*Uart_number).Rx_Flag == Uart_Rev_GetFH1){       // detected frame 1
            if(c == DTHD2){
                (*Uart_number).Rx_Flag = Uart_Rev_GetFH2;
                (*Uart_data).varData[(*Uart_data).varIndex++] = c;
            }else{
                (*Uart_number).Rx_Flag = Uart_Rev_Pre;
            }
        }else if((*Uart_number).Rx_Flag == Uart_Rev_GetFH2){       // detected frame 2
            if(c < 3 || c > Uartx_Frame_Data_Lenth){
                (*Uart_number).Rx_Flag = Uart_Rev_Pre;
            }else{
                (*Uart_data).dataLen = c;
                (*Uart_data).varData[(*Uart_data).varIndex++] = c;
            (*Uart_number).Rx_Flag = Uart_Rev_GetLen;
        }
    }else if((*Uart_number).Rx_Flag == Uart_Rev_GetLen){        // detected length
        if(c != 0x82 && c != 0x83){
            if(c==0x80||c==0x81){
                (*Uart_data).dataCode = c;
                (*Uart_data).varData[(*Uart_data).varIndex++] = c;
                (*Uart_number).Rx_Flag=Uart_Rev_GetAddr2;
                (*Uart_data).dataLen+=2;
            }else
             (*Uart_number).Rx_Flag = Uart_Rev_Pre;
        }else{
            (*Uart_data).dataCode = c;
            (*Uart_data).varData[(*Uart_data).varIndex++] = c;
            (*Uart_number).Rx_Flag = Uart_Rev_GetCode;
        }
    }else if((*Uart_number).Rx_Flag == Uart_Rev_GetCode){   // detected operate command
        (*Uart_data).varAddr = ((u16)c) << 8;
        (*Uart_data).varData[(*Uart_data).varIndex++] = c;
        (*Uart_number).Rx_Flag = Uart_Rev_GetAddr1;
    }else if((*Uart_number).Rx_Flag == Uart_Rev_GetAddr1){  // detected VP high bit
        (*Uart_data).varAddr |= c;
        (*Uart_data).varData[(*Uart_data).varIndex++] = c;
        (*Uart_number).Rx_Flag = Uart_Rev_GetAddr2;
    }else if((*Uart_number).Rx_Flag == Uart_Rev_GetAddr2){  // detected VP low bit
        (*Uart_data).varData[(*Uart_data).varIndex++] = c;
        if((*Uart_data).varIndex == (*Uart_data).dataLen + 3){
            (*Uart_number).Rx_Flag = Uart_Rev_Done;
            break;
        }
    }
  }

  if((*Uart_number).Rx_Flag == Uart_Rev_Done){              //handle 1 frame data
    (*Uart_number).Rx_Flag = Uart_Rev_Pre;
    deal_uart_data((*Uart_data).varData,Uart_num,Response,Crc_ck);

  }
}
```

9. UART handle function

```
void Uart_Handle_all()
{
    Uart_Handle_Frame(&Uart2,&Uart2_Frame,2,Uart2_Rx_Lenth,RESPONSE_UART2,CRC_CHECK_UART2);
    Uart_Handle_Frame(&Uart3,&Uart3_Frame,3,Uart3_Rx_Lenth,RESPONSE_UART3,CRC_CHECK_UART3);
    Uart_Handle_Frame(&Uart4,&Uart4_Frame,4,Uart4_Rx_Lenth,RESPONSE_UART4,CRC_CHECK_UART4);
    Uart_Handle_Frame(&Uart5,&Uart5_Frame,5,Uart5_Rx_Lenth,RESPONSE_UART5,CRC_CHECK_UART5);
}
```