通用异步串行接口3

程晨闻
东南大学电气工程学院

➢ TM4C1294中的UART模块使用方法

– 1. 使能UART模块的时钟（操作**RCGCUART** 寄存器）

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
```

– 2. 使能UART模块引脚用到的GPIO模块（使用**RCGCGPIO** 寄存器）

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
```

– 3. 配置GPIO的复用功能，使其与UART模块相连

```
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
GPIOPinConfigure(GPIO_PA0_U0RX); GPIOPinConfigure(GPIO_PA1_U0TX);
```

– 4. 计算配置波特率分频器UARTIBRD和UARTFBRD

– 5. **禁用UART模块**，配置数据格式等参数（UARTLCRH），配置时钟源UARTCC，**使能UART模块**

```
UARTConfigSetExpClk(UART0_BASE, g_ui32SysClock, 115200,
                    (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
                     UART_CONFIG_PAR_NONE));
```

– 6. 使用UART模块发送和接收数据

➢ 内容概要

–通用异步接收发送模块UART的使用方法

–开发板UART功能的硬件连接与调试

➢ TM4C1294中的UART模块使用方法

– UART模块的中断使用方法：

UART模块支持多种中断，可以配合中断响应，提高程序的执行效率。
UART中断屏蔽寄存器UARTIM，可以设置开启或者关闭哪些中断。

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x038
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved														DMATXIM	DMARXIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITIM	EOTIM	OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRIM	DCDIM	CTSIM	RIIM
Type	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RXIM：接收中断。如果RxFIFO中有数据，且数据的数量超过了**UARTIFLS**寄存器定义的值，就产生中断。
RTIM：接收超时中断。如果RxFIFO中有数据，且连续32个内部周期没有收到新数据，则产生中断。
TXIM：发送中断。如果UARTCTL的EOT设置为1，那么等所有的数据都发送完，就产生中断。如果UARTCTL的EOT设置为0，那么发送FIFO中的数据个数，少于**UARTIFLS**寄存器定义的值，就产生中断。

➢ TM4C1294中的UART模块使用方法

– UART模块的中断使用方法：

UART模块支持多种中断，可以配合中断响应，提高程序的执行效率。

UART中断屏蔽寄存器UARTIM，可以设置开启或者关闭哪些中断。

RXIM：接收中断。如果RxFIFO中有数据，且数据的数量超过了UARTIFLS寄存器定义的值，就产生中断。

TXIM：发送中断。如果UARTCTL的EOT设置为1，那么等所有的数据都发送完，就产生中断。如果UARTCTL的EOT设置为0，那么发送FIFO中的数据个数，少于UARTIFLS寄存器定义的值，就产生中断。

UARTIFLS FIFO中断等级寄存器UARTIFLS，规定了触发接收和发送中断的条件：

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x034
Type RW, reset 0x0000.0012

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										RXIFLSEL			TXIFLSEL		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

➢ TM4C1294中的UART模块使用方法

– UART模块的中断使用方法：

UART模块支持多种中断，可以配合中断响应，提高程序的执行效率。

UARTIFLS FIFO中断等级寄存器UARTIFLS，规定了触发接收和发送中断的条件：

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x034
Type RW, reset 0x0000.0012

For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										RXIFLSEL			TXIFLSEL		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

RXIFLSEL RW 0x2 UART Receive Interrupt FIFO Level Select

The trigger points for the receive interrupt are as follows:

Value	Description
0x0	RX FIFO ≥ ⅛ full
0x1	RX FIFO ≥ ¼ full
0x2	RX FIFO ≥ ½ full (default)
0x3	RX FIFO ≥ ¾ full
0x4	RX FIFO ≥ ⅞ full
0x5-0x7	Reserved

TXIFLSEL RW 0x2 UART Transmit Interrupt FIFO Level Select

The trigger points for the transmit interrupt are as follows:

Value	Description
0x0	TX FIFO ≤ ⅞ empty
0x1	TX FIFO ≤ ¾ empty
0x2	TX FIFO ≤ ½ empty (default)
0x3	TX FIFO ≤ ¼ empty
0x4	TX FIFO ≤ ⅛ empty
0x5-0x7	Reserved

➢ TM4C1294中的UART模块使用方法

– UART模块的中断使用方法：

UART模块支持多种中断，可以配合中断响应，提高程序的执行效率。

UARTIFLS FIFO中断等级寄存器UARTIFLS，规定了触发接收和发送中断的条件：

TivaWare提供了**UARTFIFOLevelSet**函数，设置**UARTIFLS**寄存器

```
void
UARTFIFOLevelSet(uint32_t ui32Base, uint32_t ui32TxLevel,
                 uint32_t ui32RxLevel)
{
    // Check the arguments.
    ASSERT(_UARTBaseValid(ui32Base));
    ASSERT((ui32TxLevel == UART_FIFO_TX1_8) ||
           (ui32TxLevel == UART_FIFO_TX2_8) ||
           (ui32TxLevel == UART_FIFO_TX4_8) ||
           (ui32TxLevel == UART_FIFO_TX6_8) ||
           (ui32TxLevel == UART_FIFO_TX7_8));
    ASSERT((ui32RxLevel == UART_FIFO_RX1_8) ||
           (ui32RxLevel == UART_FIFO_RX2_8) ||
           (ui32RxLevel == UART_FIFO_RX4_8) ||
           (ui32RxLevel == UART_FIFO_RX6_8) ||
           (ui32RxLevel == UART_FIFO_RX7_8));
    // Set the FIFO interrupt levels.
    HWREG(ui32Base + UART_O_IFLS) = ui32TxLevel | ui32RxLevel;
}
```

```
#define UART_FIFO_TX1_8     0x00000000  // Transmit interrupt at 1/8 Full
#define UART_FIFO_TX2_8     0x00000001  // Transmit interrupt at 1/4 Full
#define UART_FIFO_TX4_8     0x00000002  // Transmit interrupt at 1/2 Full
#define UART_FIFO_TX6_8     0x00000003  // Transmit interrupt at 3/4 Full
#define UART_FIFO_TX7_8     0x00000004  // Transmit interrupt at 7/8 Full

#define UART_FIFO_RX1_8     0x00000000  // Receive interrupt at 1/8 Full
#define UART_FIFO_RX2_8     0x00000008  // Receive interrupt at 1/4 Full
#define UART_FIFO_RX4_8     0x00000010  // Receive interrupt at 1/2 Full
#define UART_FIFO_RX6_8     0x00000018  // Receive interrupt at 3/4 Full
#define UART_FIFO_RX7_8     0x00000020  // Receive interrupt at 7/8 Full
```

➢ TM4C1294中的UART模块使用方法

– UART模块的中断使用方法：

发生中断后，需要**查看中断的状态**，并且**清除中断**

在UARTMIS寄存器中查看中断的状态：

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x040
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved														DMATXMIS	DMARXMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITMIS	EOTMIS	OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	DSRMIS	DCDMIS	CTSMIS	RIMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTMIS为接收超时中断状态
TXMIS为发送中断状态
RXMIS为接收中断状态。

➢ TM4C1294中的UART模块使用方法

– UART模块的中断使用方法：

发生中断后，需要**查看中断的状态**，并且**清除中断**

使用UARTICR寄存器清除中断：

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x044
Type W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved														DMATXIC	DMARXIC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITIC	EOTIC	OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC
Type	RO	RO	RO	RW	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

把**RTIX**、**TXIC**、**RXIC**位写1，分别可以清除接收超时中断、发送中断和接收中断。

➢ TM4C1294中的UART模块使用方法

– **UART模块的中断使用方法:**

TivaWare提供了**UARTIntEnable**来使能中断
提供了**UARTIntStatus**函数查看中断的状态
提供了**UARTIntClear**函数，清除中断。
UARTIntEnable函数的定义如下:

```
void
UARTIntEnable(uint32_t ui32Base, uint32_t ui32IntFlags)
{   // Check the arguments.
    ASSERT(_UARTBaseValid(ui32Base));
    // Enable the specified interrupts.
    HWREG(ui32Base + UART_O_IM) |= ui32IntFlags;
}
```

```
#define UART_INT_DMATX        0x20000     // DMA TX interrupt
#define UART_INT_DMARX        0x10000     // DMA RX interrupt
#define UART_INT_9BIT         0x1000      // 9-bit address match interrupt
#define UART_INT_OE           0x400       // Overrun Error Interrupt Mask
#define UART_INT_BE           0x200       // Break Error Interrupt Mask
#define UART_INT_PE           0x100       // Parity Error Interrupt Mask
#define UART_INT_FE           0x080       // Framing Error Interrupt Mask
#define UART_INT_RT           0x040       // Receive Timeout Interrupt Mask
#define UART_INT_TX           0x020       // Transmit Interrupt Mask
#define UART_INT_RX           0x010       // Receive Interrupt Mask
#define UART_INT_DSR          0x008       // DSR Modem Interrupt Mask
#define UART_INT_DCD          0x004       // DCD Modem Interrupt Mask
#define UART_INT_CTS          0x002       // CTS Modem Interrupt Mask
#define UART_INT_RI           0x001       // RI Modem Interrupt Mask
```

➢ TM4C1294中的UART模块使用方法

– **UART模块的中断使用方法:**

TivaWare提供了**UARTIntEnable**来使能中断
提供了**UARTIntStatus**函数查看中断的状态
提供了**UARTIntClear**函数，清除中断。

UARTIntStatus**函数的实现方式如下:**

```
uint32_t
UARTIntStatus(uint32_t ui32Base, bool bMasked)
{   // Check the arguments.
    ASSERT(_UARTBaseValid(ui32Base));
    // Return either the interrupt status or the raw interrupt status as
    // requested.
    if(bMasked)
    {
        return(HWREG(ui32Base + UART_O_MIS));
    }
    else
    {
        return(HWREG(ui32Base + UART_O_RIS));
    }
}
```

➢ TM4C1294中的UART模块使用方法

– **UART模块的中断使用方法：**

TivaWare提供了**UARTIntEnable**来使能中断
提供了**UARTIntStatus**函数查看中断的状态
提供了**UARTIntClear**函数，清除中断。

UARTIntClear函数用于清除中断，其实现方式如下

```
void
UARTIntClear(uint32_t ui32Base, uint32_t ui32IntFlags)
{
    //
    // Check the arguments.
    //
    ASSERT(_UARTBaseValid(ui32Base));

    //
    // Clear the requested interrupt sources.
    //
    HWREG(ui32Base + UART_O_ICR) = ui32IntFlags;
}
```

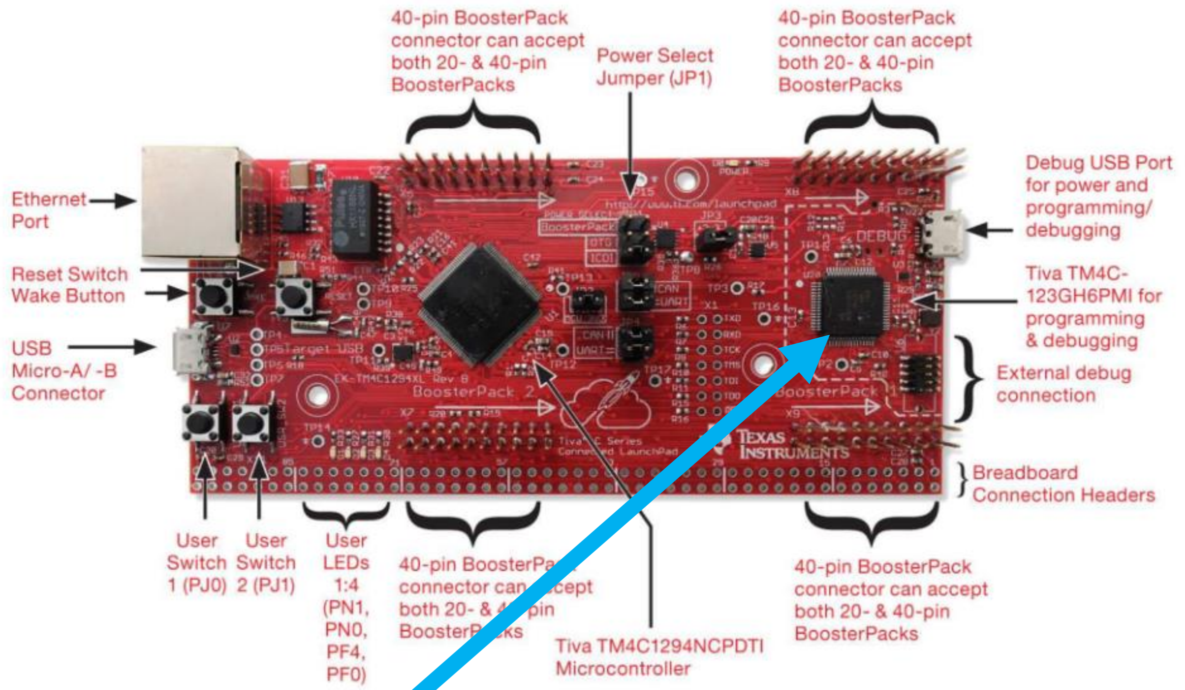
➢ TM4C1294中的UART模块使用方法

– **UART模块的中断使用方法:**

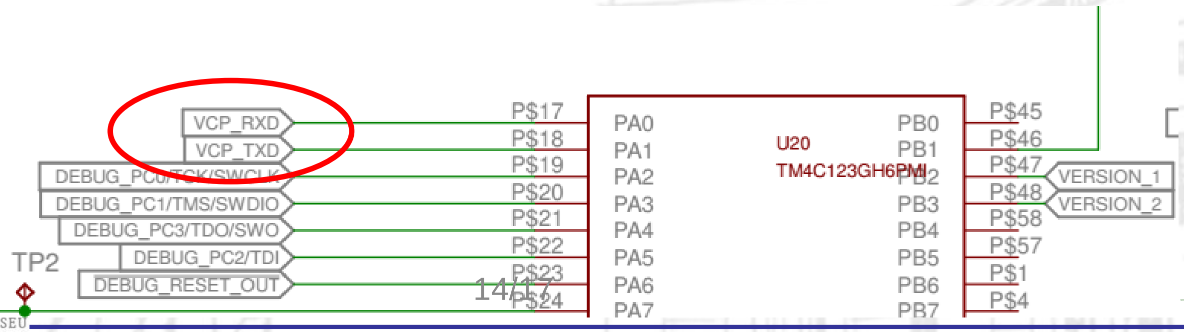
```
IntMasterEnable();
IntEnable(INT_UART0);
UARTIntRegister(UART0_BASE,UARTIntHandler);
UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);
```

```
void
UARTIntHandler(void)
{
    uint32_t ui32Status;
    ui32Status = UARTIntStatus(UART0_BASE, true);
    UARTIntClear(UART0_BASE, ui32Status);

    while(UARTCharsAvail(UART0_BASE))
    {
        UARTCharPutNonBlocking(UART0_BASE,
                                           UARTCharGetNonBlocking(UART0_BASE));

        GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);

        SysCtlDelay(g_ui32SysClock / (1000 * 3));
        GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
    }
}
```

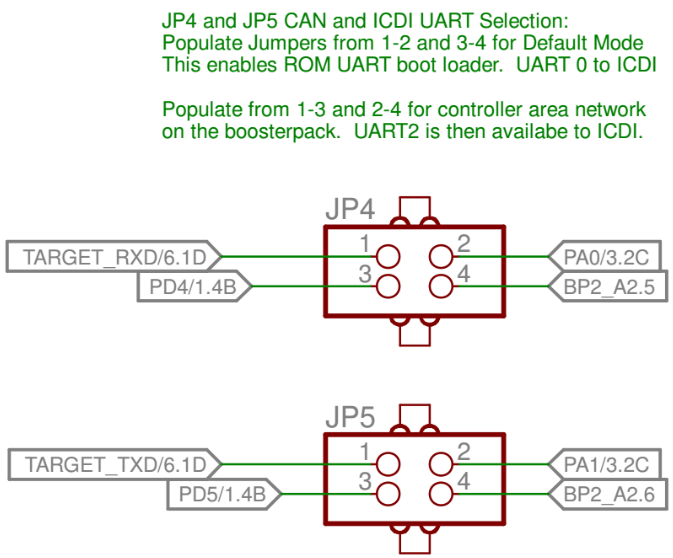
➢ 开发板的UART功能的硬件连接与调试

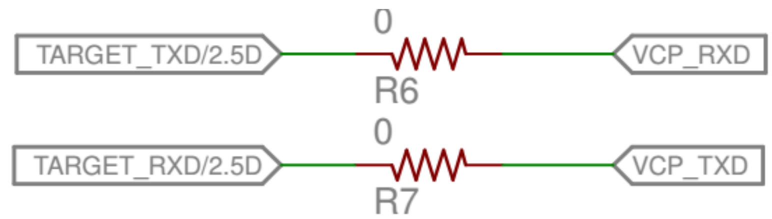


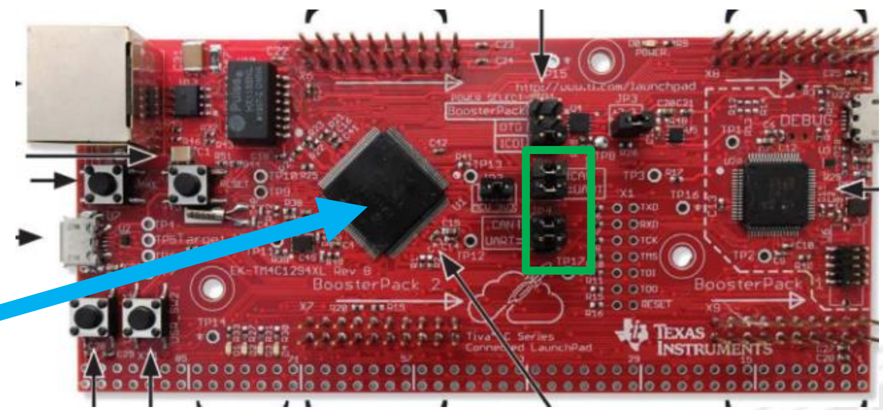
调试器的TM4C123微控制器的PA0引脚和PA1引脚分别是接收线VCP_RXD和发送线VCP_TXD。



➢ 开发板的UART功能的硬件连接与调试





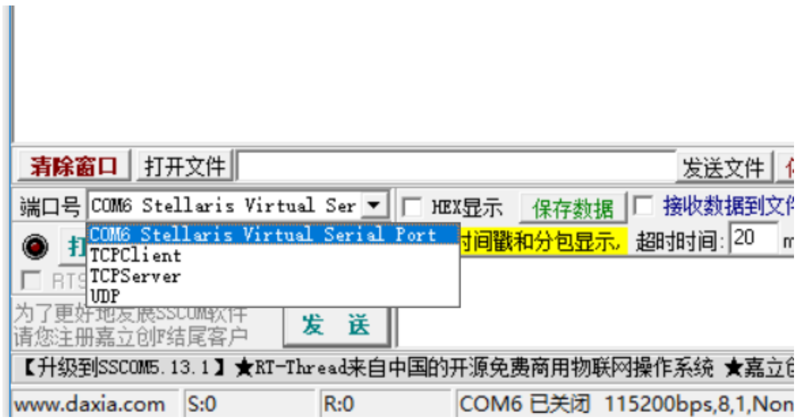


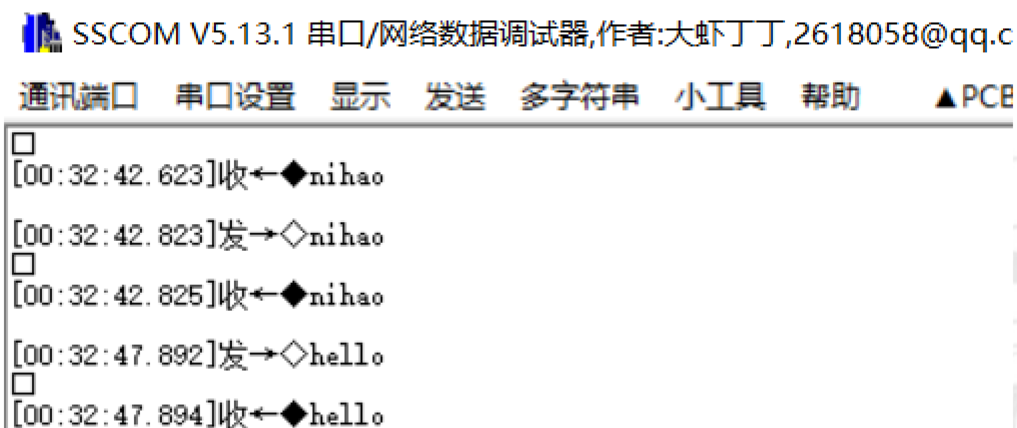
- TM4C1294微控制器的PA0引脚和PA1引脚连接至跳线插座JP4和JP5。
- 如果把跳线横向短接，即12短接，34短接
- 那么TM4C1294微控制器的PA0引脚（UART0模块的接收线）就连到了信号TARGET_RXD上，
- TM4C1294 微 控 制 器 的 PA1 引 脚 （UART0 模 块 的 发 送 线）就 连 到 了 信 号 TARGET_TXD上。

➢ 开发板的UART功能的硬件连接与调试

打开CCS软件，uart_echo工程，编译后运行

打开串口调试助手sscom5.1。选择端口号COM? Stellaris Virtual Serial Port，波特率选 115200，然后打开串口。





谢谢！