乐升半导体
LEVETOP

LT7589

串口屏控制芯片

脚位对应软件配置说明

V1.5

www.levetop.cn

Levetop Semiconductor Co.，Ltd.

版本记录

版 本	日 期	说 明
V1.0	2024/12/11	初版
V1.1	2025/06/25	修改部分错误说明
V1.2	2025/12/12	修改部分错误说明
V1.5	2026/03/10	增加 LT7589C 信息 修改 表 1-1：LT7589 型号 增加 图 1-4：LT7589C (LQFP128) 脚位图

版权说明

目　　录

1. 前言

LT7589 是一款高效能 Uart TFT 串口屏控制芯片。内部结合了 32bit RISC MCU 及具有 TFT LCD 图形显示控制器的 GUI（Graphical User Interface），主要的功能就是提供 Uart 串口通讯，让主控端 MCU 透过简易的串口指令就能轻易的将要显示的信息呈现到 TFT 屏上。LT7589 内部硬件还提供 JPG 图片解码、PIP（Picture-in-Picture）、几何图形绘图等功能，能够提升 TFT 屏显示效率，及降低 MCU 处理图形显示所花费的时间，LT7589 支持的显示分辨率由 480*480（QVGA）到 1280*800，用于 16/24bits 的 RGB 接口显示屏。

由于含有高容量的 Flash、SRAM 及众多 IO 接口，LT7589 可以将部分接口资源拿来使用，或是将 LT7589 作为主控的 MCU，将主控及 TFT 显示功能由一颗 LT7589A/B/C 来完成，本说明书介绍在使用 LT7589 的脚位时所对应的软件配置。

LT7589 有三种封装及型号，分别如下：

- QFN96 (10*10 mm²) – LT7589A-L01 ; LT7589A-L02
- LQFP128 (14*14 mm²) – LT7589B
- LQFP128 (14*14 mm²) – LT7589C

QFN96 (10*10 mm²)

LQFP128 (14*14 mm²)

LQFP128 (14*14 mm²)

图 1-1：LT7589A、LT7589B 和 LT7589C 和外观图

表 1-1：LT7589 型号

型号	封装	内建 Flash	内建 SRAM	内建显示内存	分辨率	色 彩
LT7589A(L01)	QFN96	2M Bytes	768KBytes	128Mb	1280*800	16.7M 色 αRGB 8:8:8:8
LT7589A(L02)	QFN96	2M Bytes	768KBytes	256Mb	1280*800	16.7M 色 αRGB 8:8:8:8
LT7589B	LQFP128	2M Bytes	768KBytes	128Mb	1280*800	16.7M 色 αRGB 8:8:8:8
LT7589C	LQFP128	2M Bytes	768KBytes	256Mb	1280*800	16.7M 色 αRGB 8:8:8:8

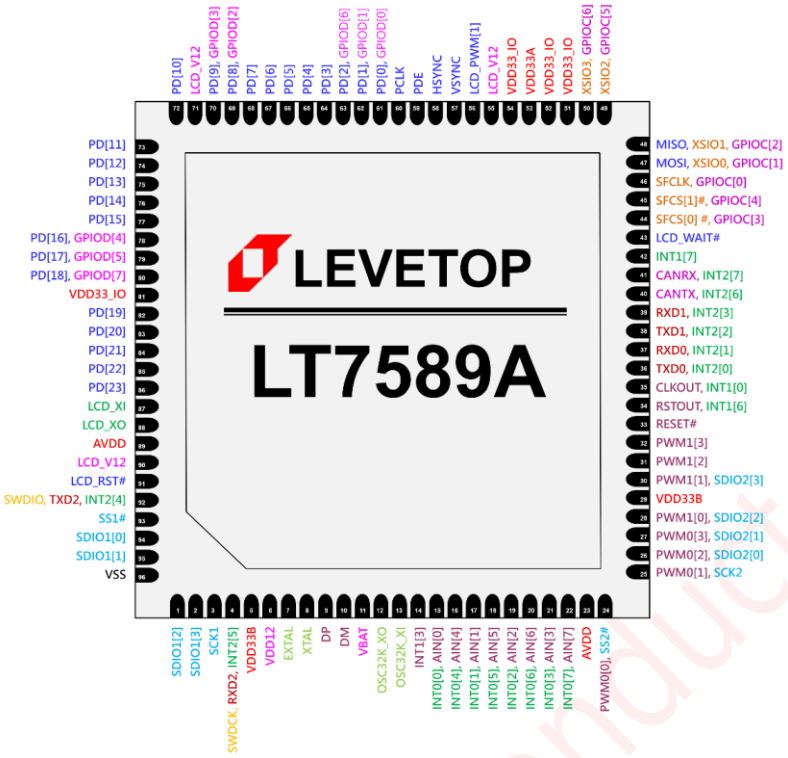

图 1-2：LT7589A (QFN96) 脚位图

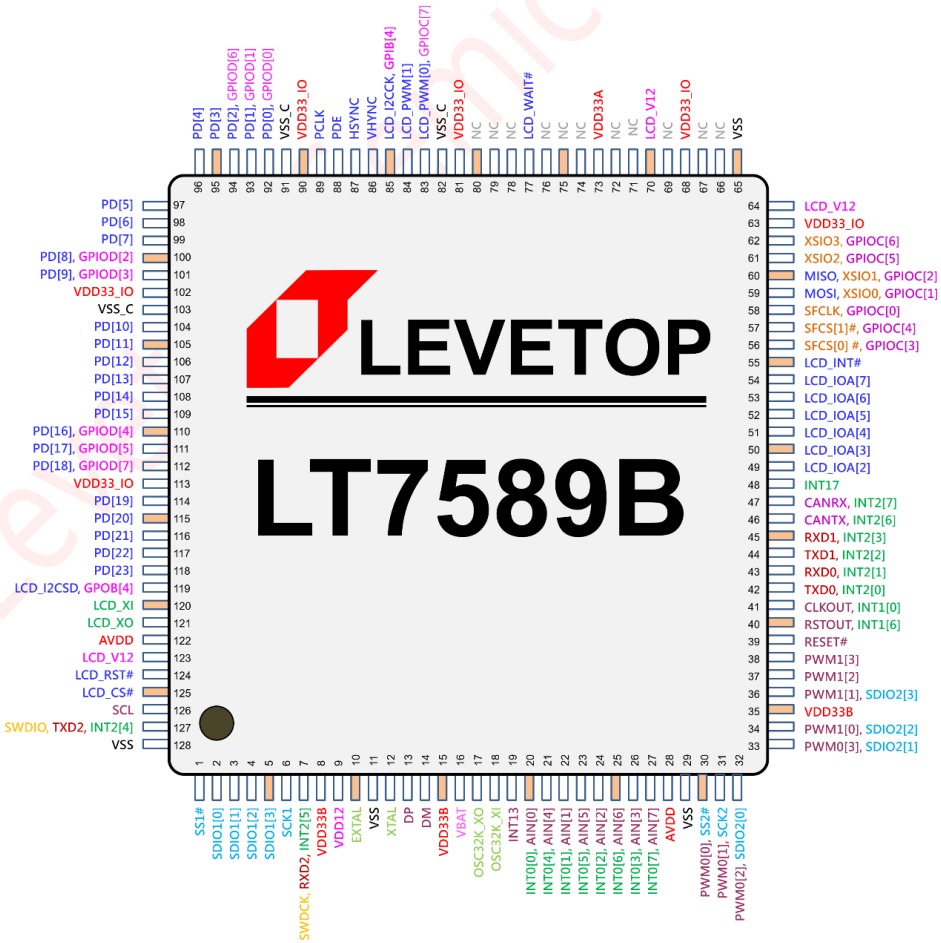

图 1-3：LT7589B (LQFP128) 脚位图

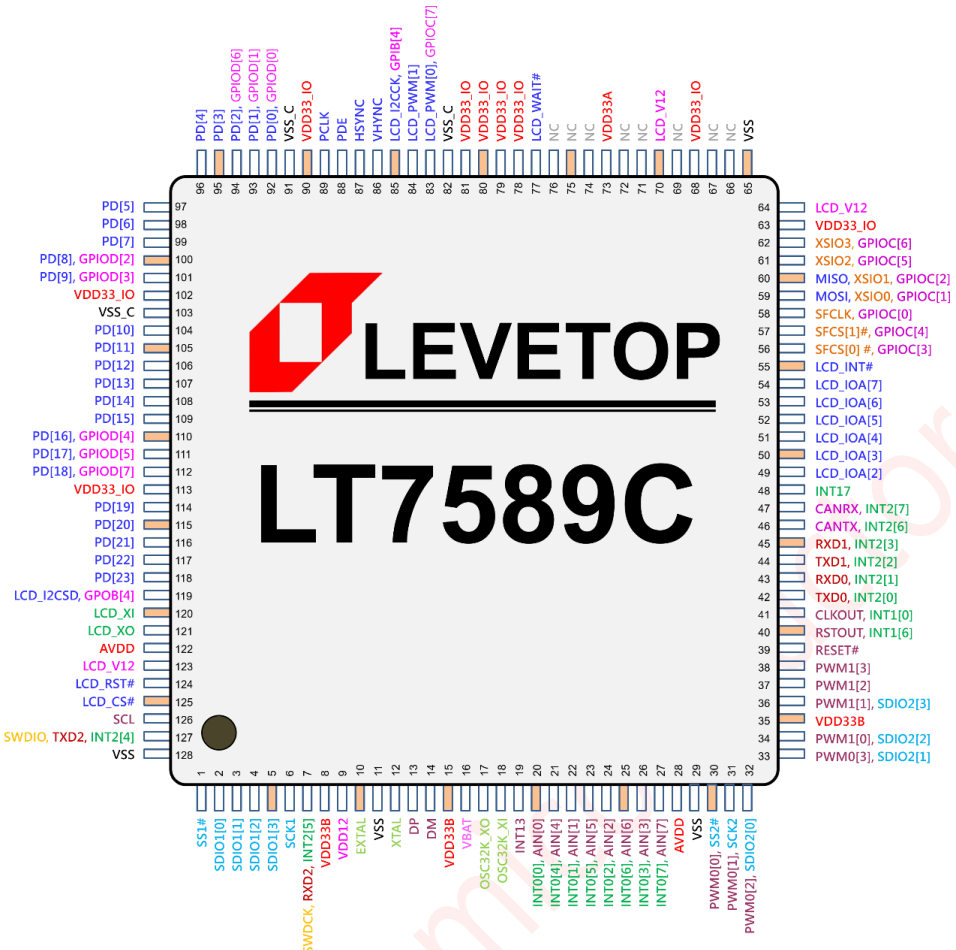

图 1-4：LT7589C (LQFP128) 脚位图

2. AIN[X]/INT0[X] 端口

对应芯片脚位 LT7589A: Pin15-Pin22；LT7589B/C: Pin20-Pin27

2.1. AIN[x]配置 ADC 功能

此组端口默认是 ADC 功能，ADC 程序配置请参考 SDK 的 ADC demo 程序，以 AIN[2]为例:

```
void ADC_Init(void)
{
    PIN_ADC_CH2_Default();  //保留 AIN[2]功能
    u16 PreClk_DIVx = 11;
    ADC->U32_ADC_CFGR2.QPR = PreClk_DIVx;                  // 预分频时钟位
    ADC->U32_ADC_CFGR2.STCNT = 2*(168/(PreClk_DIVx+1)); // 启动计数位
    ADC->U32_ADC_CFGR1.CONT = 0;                           //关闭连续转换
    ADC->U32_ADC_CFGR1.SEQ_LEN = 0;                        //转换序列为 1
    ADC->U32_ADC_CFGR1.RES = 0;                            //12 位
    ADC->U32_ADC_CFGR1.ALIGN = 0;                          //右对齐
//  ADC->U32_ADC_CHSELR1.CCW0 = 2;                         //选择转换通道 0 来转换 AIN[2]
    ADC->U32_ADC_SMPR.SMP = 6-2;                           //采样时间
    ADC->U32_ADC_CR.ADEN = 1;                              //使能 ADC
    while(ADC->U32_ISR.ADRDY==0);                  //等待 ADC 就绪
}
u16 Get_ADC_Val(ADC_CH ch)      //读取对应 ADC 脚位的值。
{
    u16 Val = 0;
    ADC->U32_ADC_CHSELR1.CCW0 = ch;                        //选择转换通道
    ADC->U32_ADC_CR.ADSTART = 1;                           //开始转换
    while(ADC->U32_ISR.EOSEQ==0);                  //等待转换序列结束
    ADC->ADC_ISR |= ADC_FLAG_EOSEQ;                        //清除结束标志
    Val = ADC->ADC_FIFO;                                   //读数据
    return Val;
}
void ADC_Demo(void)
{
    ADC_Init();
    LTPrintf("ADC2_value = %d \r\n",Get_ADC_Val(ADC_CH2));
}
```

2.2. INT0[x] 端口配置为 EPORT 端口

INT0[x]端口默认是 ADC 功能，配置为 EPORT 功能需要先设置 Eport 复用函数，程序参考如下：

```
//ADC[0-7] 配置复用为 Eport0[0-7]----------------
void PIN_EPORT0_0(void);
void PIN_EPORT0_1(void);
void PIN_EPORT0_2(void);
void PIN_EPORT0_3(void);
void PIN_EPORT0_4(void);
void PIN_EPORT0_5(void);
void PIN_EPORT0_6(void);
void PIN_EPORT0_7(void);
```

注意：EPORT0[X]端口需要设置复用，EPORT1[X]/EPORT2[X] 端口不需要。

EPORT 端口输入输出寄存器：EPORT0->EPDDR
EPORT 端口输出高低电平寄存器：EPORT0->EPDR
EPORT 端口高低电平寄存器：EPORT0->EPPDR
EPORT 端口上拉使能寄存器：EPORT0->EPPUE

例程 1 配置 AIN[0]/INT0[0] 输出高低电平

```
PIN_EPORT0_0();
EPORT0->EPDDR |= (1<<0);          //配置 EPORT0[0] 为输出
EPORT0->EPDR |= (1<<0);           //配置 EPORT0[0] 输出高电平
EPORT0->EPDR &= ~(1<<0);          //配置 EPORT0[0] 输出低电平
```

例程 2 配置 AIN[3]/INT0[3] 输入并读取电平状态

```
unsigned char temp;
PIN_EPORT0_3();
EPORT0->EPDDR &= ~(1<<3);              //配置 EPORT0[3] 为输入
EPORT0->EPPUE|= (1<<3) ;               //配置 EPORT0[3] 内部上拉使能
Temp =(EPORT0->EPPDR)& (1<<3) ;        //读取 EPORT0[3] 状态
```

例程 3 配置 AIN[5]/INT0[5] 为 EPORT 外部上升沿/下降沿中断

```
PIN_EPORT0_5();
EPORT0->EPDDR &= ~(1<<5);              //配置 EPORT0[5] 为输入
EPORT0->EPIER |= (1<<5);               //配置 EPORT0[5] 中断使能
EPORT0->EPPAR &= ~(3<<10);             //clear,
EPORT0->EPPAR |= (1<<10);              //1: Rising edge triggered
//EPORT0->EPPAR |= (2<<10);            //2: Falling edge triggered
//EPORT0->EPPAR |= (3<<10);            //3: Rising and Falling edge triggered
```

```
EPORT0->EPFR = (1<<5);                    //清除 EPORT0[5] 中断标志
EIC->IER |= IRQ(30);                      //配置 EPORT0[5] 全局中断使能

void EPORT0_Handler(void)                 //中断函数
{
  unsigned char temp;
  temp = EPORT0->EPFR;
//User'S code...
  EPORT0->EPFR = temp;
}
```

例程 4 配置 AIN[5]/INT0[5] 为高/低电平中断

```
PIN_EPORT0_5();
EPORT0->EPDDR &= ~(1<<5);                 //配置 EPORT0[5] 为输入
//EPORT0->EPPUE|= (1<<5);                 //配置 EPORT0[5] 内部上拉使能，低电平中断时打开
//EPORT0->EPPUE &= ~(1<<5);               //配置 EPORT0[5] 关闭内部上拉，高电平中断时打开
EPORT0->EPDDR &= ~(1<<5);                 //配置 EPORT0[5] 为输入
EPORT0->EPPAR &= ~(3<<10);                //0:HIGH/LOW level-sensitive,
EPORT0->EPLPR |= (1<<5);                  //HIGH level-sensitive if EPPAR = 0；高电平中断时打开
//EPORT0->EPLPR &= ~(1<<5);               //LOW level-sensitive if EPPAR = 0；低电平中断时打开
EPORT0->EPIER |= (1<<5);                  //配置 EPORT0[5] 中断使能
EIC->IER |= IRQ(30);                      //配置 EPORT0[5] 全局中断使能

void EPORT0_Handler(void)
{
    EPORT0->EPIER &= ~(1<<5);   //关闭 EPORT0[5] 中断
    //User'S code...
}
```

Note：Eport 的高低电平中断不能一直打开，进中断后关闭中断，需要时再打开。

	15	14	13	12	11	10	9	8
Read: Write:	EPPA7[1:0]		EPPA6[1:0]		EPPA5[1:0]		EPPA4[1:0]	
Reset:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
Read: Write:	EPPA3[1:0]		EPPA2[1:0]		EPPA1[1:0]		EPPA0[1:0]	
Reset:	0	0	0	0	0	0	0	0

图 2-1：EPPAR 寄存器

3. PWM 端口

LT7589x 有 2 类 PWM 端口，分别对应芯片内部的 MCU 部分和显示驱动部分。

3.1. MCU 部分 PWM 端口

(PWM0[0-3], PWM1[0-1])/QSPI2, PWM1[2-3] 端口对应芯片脚位 LT7589A: Pin24-Pin28, Pin30-Pin32; LT7589B/C: Pin30-Pin34, Pin36-Pin38

此组端口默认是 PWM 功能，PWM 功能配置如下，以 PWM1[3] 为例：

```
void PWM_OutputInit(uint16_t PWM_Width)
{
    // prescale: PWM_Prescaler+1
    // PWM_Csr: 0=1/2, 1=1/4, 2=1/8, 3=1/16, others=1
    // period: WM_Period+1
    // Actual PWM output: 75Mhz/(PWM_Prescaler+1)/(PWM_Csr)/(PWM_Period+1)
    // prescale: PWM_Prescaler+1
    // CP: 0=stop, others=CP+1
    // CSR: 0=2, 1=4, 2=8, 3=16, others=1
    // PCNR: 0=stop,
    // PCMR: CMR+1
    // period: PCNR+1
    // Actual PWM output: 168Mhz/CSR/CP/PCNR
    PIN_PWM_Default();
    PWM1->U32_PCSR.CSR3 = 2;        //时钟分频(设置 2=8 分频,即时钟为 sys_clk:168/8=21Mhz)
    PWM1->U32_PPR.CP1 = 0;          //时钟预分频(0=未分频，最终输出时钟还是 21Mhz)
    PWM1->U32_PCR.CH3MOD = 1;       //定时器输出反转开关(1：反转开，0：反转关)
    PWM1->PCNR3 = 1050;             //周期时间 (21Mhz/1050=20Khz，即输出为 20Khz 的频率)
    PWM1->PCMR3 = 525;              //设置占空比，即最后的占空比为（525/1050）*100% = 50%
    PWM1->U32_PPCR.PDDR_3 = 1;      //设置 PWM13 为输出
    PWM1->U32_PCR.CH3EN = 1;        //使能 PWM13
}
void PWM_OutputClose(void)
{
    PWM1->U32_PCR.CH3EN = 0;
}
void PWM_OutputOpen(void)
{
    PWM1->U32_PCR.CH3EN = 1;
}
```

```
void PWM_Output_DutySet(uint16_t duty)
{
    if (duty == 0)
        LT_BacklightClose();
    else
    {
        PWM1->PCMR3 = duty;        //high leve = duty+1
        LT_BacklightOpen();
    }
}
```

以 PWM 配置为 GPIO 输出/输入,PWM0[3] 为例:

```
//PWM0[3] 配置为输出
PWM_GPIO_Init(PWM0, PWM_CH3, PWM_GPIO_Output, PWM_GPIO_No_PullUp);
PWM_GPIO_Write_Pin(PWM0, PWM_CH3, Reset);//PWM0[3] 输出低电平
PWM_GPIO_Write_Pin(PWM0, PWM_CH3, Set);  //PWM0[3] 输出高电平

unsigned char temp;

//PWM0[3] 配置为输入
PWM_GPIO_Init(PWM0, PWM_CH3, PWM_GPIO_Input, PWM_GPIO_No_PullUp);
Temp = PWM_GPIO_Read_Pin(PWM0, PWM_CH3);
```

此组端口配置为 QSPI 功能，对应端口(PWM0[0-3], PWM1[0-1])/QSPI2

```
void PIN_QSPI2(void)                    //配置端口复用为 QSPI2
{
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x01);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x02);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x03);
    OPTION->CCR |= (1 << 8);
    OPTION->CCR &= 0xFFFC;
}
```

QSPI 功能请参考串口屏工程中驱动 QSPI 屏部分代码:

```
void LCD_QSPI_STD_Init(void);
uint16_t LCD_QSPI_ReadWriteByte(uint32_t TxData);
```

此组端口配置为普通 4 线 SPI 功能，对应端口 PWM0[0-3]，SPI 功能请参考串口屏工程中驱动 SPI 屏部分代码:

```
void LCD_SPI_Port_Init(void);
```

uint16_t LCD_SPI_ReadWriteByte(uint32_t TxData);

3.2. 显示驱动部分 PWM 端口

LCD_PWM[0-1]对应芯片脚位 LT7589A: Pin56；LT7589B/C: Pin83-Pin84

开发板上控制背光亮度的就是 LCD_PWM[1] ，对应函数：

```
void LT758_PWM1_Init
(
  uint8_t on_off
,uint8_t Clock_Divided
,uint8_t Prescalar
,uint16_t Count_Buffer
,uint16_t Compare_Buffer
);
void LT758_PWM1_Duty(uint16_t Compare_Buffer);
```

4. RSTOUT/INT1[6], CLKOUT/INT1[0], INT13 端口

对应芯片脚位 LT7589A: Pin34, Pin35, Pin14；LT7589B/C: Pin40, Pin41, Pin19

RSTOUT/INT1[6]和 CLKOUT/INT1[0]端口默认配置为 RSTOUT/CLKOUT 功能，需通过程序配置为 EPORT1[6]/EPORT1[0]，INT13 默认是 EPORT1[3]。

```
void PIN_EPORT1_6(void)  //RSTOUT 端口配置为 EPORT1[6]
{
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x01);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x02);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x03);
    OPTION->CCR |= (1 << 11);
    OPTION->CCR &= 0xFFFC;
}

void PIN_EPORT1_0(void)  //CLKOUT 端口配置为 EPORT1[0]
{
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x01);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x02);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x03);
    OPTION->CCR |= (1 << 10);
    OPTION->CCR &= 0xFFFC;
}
```

EPORT1[X] 程序可以参考第 2 章 Eport 配置程序。

5. TXD0/INT2[0], RXD0/INT2[1], TXD1/INT2[2], RXD1/INT2[3] 端口

对应芯片脚位 LT7589A: Pin36-Pin39；LT7589B/C: Pin42-Pin45

此 2 组端口默认为串口功能，以 Uart0 为例，程序如下：

```
void UartTFT_SCI0_Init(uint32_t bound)
{
    uint16_t SBR = 0;
    EIC->U32_IER.IE_SCI0 = 0;

  SCI0->CTRL = 0x0;
  SBR = Find_SCI_Optimal_SBR(bound);
  Bit_Clear(SCI0->BAUD,SCI_BAUD_SBR_MASK);   // Baud rate = IPS/((OSR+1)*SBR) ,1<SBR<8191
  Bit_Set(SCI0->BAUD,SCI_BAUD_SBR(SBR));
 SCI0->OSR = SCI_OSR( SYS_CLOCK / ((SBR) * bound) - 1);// 3<=OSR<=255，如果小于 3 则会被设为 15

    Bit_Set(SCI0->CTRL,SCI_CTRL_RIE_MASK);   // 接受中断使能
    Bit_Set(SCI0->CTRL,SCI_CTRL_RE_MASK); //接受使能
    Bit_Set(SCI0->CTRL,SCI_CTRL_TE_MASK);   //发送使能

    EIC->U32_IER.IE_SCI0 = 1;
}

void UART0_SendByte( UINT8 data)
{
    while((SCI0->STAT&SCI_STAT_TDRE_MASK)==0);
    SCI0->DATA = data&0xff;
}

//Uart0 发送函数
void UART0_SendBytes(uint8_t *buffer, uint16_t length)
{
    UINT16 i = 0;
    for ( ; i < length; i++)
    {
        while((SCI0->STAT&SCI_STAT_TDRE_MASK)==0);
        SCI0->DATA = buffer[i]&0xff;
    }
}
```

```
//Uart0 中断接收函数
void SCI0_Handler(void)
{
    uint8_t ch = 0;
    if(Bit_Read(SCI0->STAT,SCI_STAT_RDRF_MASK))
    {
        ch = (SCI0->DATA &0xFF );
        SCI0->DATA = ch+1;
        //User’ S Code...
        //LTPrintf("ch %x \r\n", ch);
    }
}
```

此 2 组端口配置为 Eport，程序如下：

```
void PIN_EPORT2_0(void)        //配置 TXD0 为 EPORT2[0]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 0);
}
void PIN_EPORT2_1(void)        //配置 RXD0 为 EPORT2[1]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 1);
}
void PIN_EPORT2_2(void)        //配置 TXD1 为 EPORT2[2]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 2);
}
void PIN_EPORT2_3(void)        //配置 RXD1 为 EPORT2[3]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 3);
}
```

EPORT2[X] 程序可以参考第 2 章 Eport 配置程序。

6. CANTX/INT2[6], CANRX/INT2[7] 端口

对应芯片脚位 LT7589A: Pin40-Pin41；LT7589B/C: Pin46-Pin47

此组端口默认为 CAN 功能，程序请参考 CAN 测试 demo

此组端口配置为 Eport，程序如下：

```
void PIN_EPORT2_6(void)        //配置 CANTX 为 EPORT2[6]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 6);
}

void PIN_EPORT2_7(void)        //配置 CANRX 为 EPORT2[7]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 7);
}
```

EPORT2[X] 程序可以参考第 2 章 Eport 配置程序。

7. BOOT/INT1[7]端口对应芯片脚位

对应芯片脚位 LT7589A: Pin42；LT7589B/C: Pin48

此端口默认为 EPORT1[7]，程序可以参考第 2 章 Eport 配置程序。BOOT/INT1[7] 是启动位检测端口，上电期间不可以为低电平。

8. LCD_GPIO[10-15] 端口

对应芯片脚位 LT7589B/C: Pin49-Pin54

此组端口默认为 GUI 部分 GPIOA[2-7]，可以通过程序配置为低速 GPIO 端口。
以 GPIOA[2-3]为例，GPIOA[2]配置为输出，GPIOA[3]配置为输入，程序如下：

```
Set_GPIO_A_In_Out(0xFB);        //配置 1 为输入，0 为输出

//GPIOA[2]输出高低电平配置
uint8_t TTemp = 0xff;
#define GPIO_A2_Low             {TTemp &=~0x04 ; Write_GPIO_A_7_0(TTemp);}
#define GPIO_A2_High            {TTemp |=0x04 ; Write_GPIO_A_7_0(TTemp);}

//读取 GPIOA[3]电平状态
uint8_t RTemp = Read_GPIO_A_7_0()&0x08;
```

Note：这一组端口没有外部中断功能

9. PGMIO/TXD2/INT2[4], PGMCK/RXD2/INT2[5] 端口

对应芯片脚位 LT7589A: Pin4, Pin92；LT7589B/C: Pin7, Pin127

此 2 个端口默认为 PGM 下载功能，可通过程序配置为 UART2 或者 EPORT2[4-5]，

配置为 UART2 功能：

```
void PIN_RXD2_TXD2(void)
{
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x01);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x02);
    OPTION->CCR = ((OPTION->CCR & 0xFFFC) | 0x03);
    OPTION->CCR |= (1 << 12);
    OPTION->CCR &= 0xFFFC;
}

void PIN_TXD2_Default(void)          //先配置 PGMIO 为 TXD2
{
    Bit_Clear(CCM->EPORT2FCR, 1 << 4);
}

void PIN_RXD2_Default(void)          //先配置 PGMCK 为 RXD2
{
    Bit_Clear(CCM->EPORT2FCR, 1 << 5);
}
```

串口功能可参考第 5 章串口配置函数。

配置为 EPORT2[4-5] 功能：

```
PIN_RXD2_TXD2(void);                 //先配置为串口
void PIN_EPORT2_4(void)              //先配置 PGMIO 为 EPORT2[4]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 4);
}

void PIN_EPORT2_5(void)              //先配置 PGMCK 为 EPORT2[5]
{
    Bit_Set(CCM->EPORT2FCR, 1 << 5);
}
```

10. QSPI1 端口

对应芯片脚位 LT7589A: Pin1-Pin3,Pin93-Pin95；LT7589B/C: Pin1-Pin6

此组端口只能作为 QSPI 功能。

11. EFlash 应用说明

```
#define flh_sAddr     0x6007F000              //(512K 地址开始)
Uint32_t DATA[5] = {x};
void SaveData(void)                           //4K 空间用于存储
{
    uint8_t buff[4096] = {0};

    memcpy(&buff[0], DATA[0], 4);
    memcpy(&buff[4], DATA[1], 4);
    memcpy(&buff[8], DATA[2], 4);
    memcpy(&buff[12], DATA[3], 4);
    memcpy(&buff[16], DATA[4], 4);

    EFlash_Init();
    EFLASH_Write(flh_sAddr,buff,4096);
}

uint8_t LT_TpGetAdjdata(void)
{
    uint8_t i, buff[4096] = {0};
    EFlash_Read(flh_sAddr,buff,20);
    memcpy(DATA[0], &buff[0], 4);
    memcpy(DATA[1], &buff[4], 4);
    memcpy(DATA[2], &buff[8], 4);
    memcpy(DATA[3], &buff[12], 4);
    memcpy(DATA[4], &buff[16], 4);
    return 1;
}
```