乐升半导体
LEVETOP

LT7689

串口屏控制芯片

脚位对应软件配置说明

V1.2

www.levetop.cn

Levetop Semiconductor Co.，Ltd.

版本记录

版 本	日 期	说 明
V1.0	2024/4/24	初版
V1.1	2024/5/31	修改 GPIOD 部分函数、更新 SWD 配置说明。
V1.2	2024/8/1	修改第 12 章参考例程 2

版权说明

本文件之版权属于 乐升半导体 所有，若需要复制或复印请事先得到 乐升半导体 的许可。本文件记载之信息虽然都有经过校对，但是 乐升半导体 对文件使用说明的规格不承担任何责任，文件内提到的应用程序仅用于参考，乐升半导体 不保证此类应用程序不需要进一步修改。乐升半导体 保留在不事先通知的情况下更改其产品规格或文件的权利。有关最新产品信息，请访问我们的网站 Http://www.levetop.cn 。

Levetop Semiconductor

目　录

版本记录......2
版权说明......2
目　　录......3
1. 前言......4
2. SPI 端口......6
3. 串口 1 端口......8
4. ADC 端口......9
5. DAC 端口......10
6. WAKEUP 端口......10
7. IIC 端口......10
8. GINT[1-5]/UART[2-3] 端口......11
9. SWD 端口......12
10. GINT1[3]/PWM1[1] 端口......14
11. LT768_PWM1，LT768_PWM0 端口......14
12. PD[0-2]，PD[8-9]，PD[16-18] 端口......15
13. PIT 定时器......16
14. EFlash 应用说明......17

Levetop Semiconductor

© Levetop Semiconductor Co, Ltd.

1. 前言

LT7689 是一款高效能 Uart TFT 串口屏控制芯片。其内部结合了 Cortex-M4 MCU 及 2D TFT 图形显示加速器，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。除了自带高效能 M4 核 MCU 之外，内部硬件还提供图形加速、PIP（Picture-in-Picture）、几何图形绘图等功能，能够提升 TFT 显示效率，及降低 MCU 处理图形显示所花费的时间，LT7689 支持的显示分辨率由 320*240（QVGA）到 1280*1024（SXGA），16/18/24bits 的 RGB 接口显示屏。

由于含有高容量的 Flash、SRAM 及众多 IO 接口，LT7689 可以将部分接口资源拿来使用，或是将 LT7689 作为主控的 MCU，将主控及 TFT 显示功能由一颗 LT7689 来完成，本说明书介绍在使用 LT7689 的脚位时所对应的软件配置。

LT7689 封装与外观如下：

- QFN-96pin (10*10 mm^2)

图 1-1：LT7689 外观图

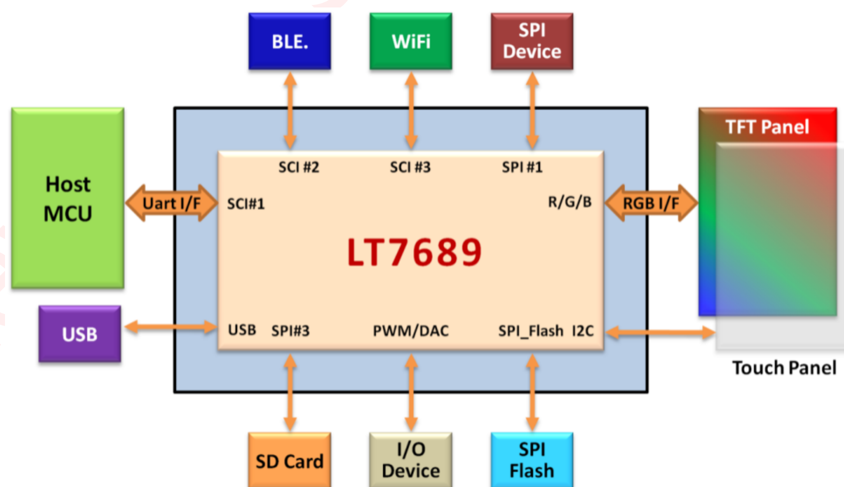

图 1-2：LT7689 应用架构

© Levetop Semiconductor Co, Ltd.

LEVETOP
LT7689
1 LDO4_V18
2 PWM_M1 / GINT[13]
3 SCK1
4 MOSI1
5 SS1#
6 MISO1
7 RXD1
8 TXD1
9 VDD_2
10 SCK3 / GINT[9]
11 MOSI3 / GINT[11]
12 MISO3 / GINT[10]
13 PWM_M2 / GINT[14]
14 AVDD
15 ADCIN[1]
16 ADCIN[0]
17 AVDD
18 DAC_OUT
19 RST#
20 WAKEUP
21 VCC35V
22 LDO3_V12
23 VDD_2
24 LDO1_V11
25 I2C_SD
26 I2C_CK
27 SS3#
28 EXTAL
29 XTAL
30 TXD3 / GINT[1]
31 RXD3 / GINT[2]
32 RXD2 / GINT[3]
33 GINT[4]
34 TXD2 / GINT[5]
35 VBAT
36 RTC_XO
37 RTC_XI
38 AVDD
39 OTG_ID
40 LDO2_V11
41 DM
42 DP
43 GINT[15] / PWM_M3
44 INT#
45 SFCS0#
46 SFCS1#
47 SFCLK
48 SFDO
49 NC
50 SFDI
51 VDD_1
52 VDD_1
53 VSS
54 LDO4_V18
55 VDD_1
56 NC
57 VDD_1
58 LDO4_V18
59 VDD_1
60 TFT_PWM[0]
61 TFT_PWM[1]
62 VSYNC
63 HSYNC
64 PDE
65 PCLK
66 VSS
67 VDD_1
68 PD[0]
69 PD[1]
70 PD[2]
71 PD[3]
72 PD[4]
73 PD[5]
74 PD[6]
75 PD[7]
76 PD[8]
77 PD[9]
78 LDO4_V18
79 PD[10]
80 PD[11]
81 PD[12]
82 PD[13]
83 PD[14]
84 PD[15]
85 PD[16]
86 PD[17]
87 PD[18]
88 VDD_1
89 PD[19]
90 PD[20]
91 PD[21]
92 PD[22]
93 PD[23]
94 TFT_XI
95 TFT_XO
96 VDD_1
0 Thermal Pad(VSS)

图 1-3：LT7689 (QFN96) 脚位图

Levetop Se

2. SPI 端口

对应 LT7689 芯片脚位 SPI1:Pin3-Pin6；SPI3:Pin10-Pin12、Pin27，此两组端口可以配置为硬件 SPI，或者 GPIO 功能。

以 SPI3 为例，配置为 GPIO：

```
SPI_ConfigGpio(SPI3,SPI_SCK,GPIO_OUTPUT);     //SCK 配置为输出
SPI_ConfigGpio(SPI3,SPI_MOSI,GPIO_OUTPUT);    //MOSI3 配置为输出
SPI_ConfigGpio(SPI3,SPI_MISO,GPIO_OUTPUT);    //MISO3 配置为输出
SPI_ConfigGpio(SPI3,SPI_SS,GPIO_OUTPUT);      //SS3 配置为输出

SPI_WriteGpioData(SPI3,SPI_SCK, 1);           //SCK3 输出高电平
SPI_WriteGpioData(SPI3,SPI_SCK, 0);           //SCK3 输出低电平

SPI_ConfigGpio(SPI3,SPI_SCK,GPIO_INPUT);      //SCK3 配置为输入
Temp = SPI_ReadGpioData(SPI3,SPI_SCK);        //读 SCK3 状态
```

配置为硬件 SPI：

```
#define SS3_RESET    SPI_WriteGpioData(SPI3,SPI_SS,Bit_RESET)
#define SS3_SET      SPI_WriteGpioData(SPI3,SPI_SS,Bit_SET)
void SPI3_Init(void)
{
    SPI_InitTypeDef SPI_InitStruct;
    SPI_InitStruct.SPI_Direction          = SPI_Direction_2Lines_FullDuplex;
    SPI_InitStruct.SPI_Mode               = SPI_Mode_Master;
    SPI_InitStruct.SPI_DataSize           = SPI_DataSize_8b;
    SPI_InitStruct.SPI_CPOL               = SPI_CPOL_Low;
    SPI_InitStruct.SPI_CPHA               = SPI_CPHA_1Edge;
    SPI_InitStruct.SPI_BaudRatePrescaler  = SPI_BaudRatePrescaler_4;
    SPI_InitStruct.SPI_FirstBit           = SPI_FirstBit_MSB;
    SPI_Init(SPI3, &SPI_InitStruct);         //根据 SPI_InitStruct 中指定的参数初始化外设 SPIx 寄存器
    SPI_Cmd(SPI3,ENABLE);
    SPI_ConfigGpio(SPI3,SPI_SS,GPIO_OUTPUT);
}
```

Levetop Semiconductor

© Levetop Semiconductor Co, Ltd.

```
u16 SPI3_ReadWriteByte(u16 TxData)
{
    while((SPI3->SPISRHW)&(SPISR_TXFFULL_MASK))          //等待发送区空
    {
    }
    SPI3->SPIDR=TxData;                                  //发送一个 byte
    while((SPI3->SPISRHW) & (SPISR_RXFEMP_MASK))         //等待接收完一个 byte
    {
    }
    return SPI3->SPIDR;                                  //返回收到的数据
}

void SPI3_DataWrite(u8 data)
{
    SS3_RESET;
    SPI3_ReadWriteByte(data);
    SS3_SET;
}

u8 SPI3_DataRead(void)
{
    u8 temp = 0;
    SS3_RESET;
    temp = SPI3_ReadWriteByte(0xff);
    SS3_SET;
    return temp;
}

void SPI3_WriteBuf(u8 *buf,u16 len)
{
    u16 i = 0;
    SS3_RESET;
    for(i = 0;i<len;i++)
        SPI3_ReadWriteByte(buf[i]);
    SS3_SET;
}
```


© Levetop Semiconductor Co, Ltd.

Levetop Semiconductor

3. 串口 1 端口

对应芯片脚位 Pin7-Pin8，此组端口可以配置为串口功能或者 GPIO 功能。

配置为串口：

```
void Uart1_Init(u32 pclk,u32 bound)
{
    u32 band_rate=0;
    SCI1->SCIBRDF=(((pclk*8/bound)+1)/2)&0x003f;
    band_rate =(pclk*4/bound)>>6;
    SCI1->SCIBDH=(u8)((band_rate>>8)&0x00ff);
    SCI1->SCIBDL=(u8)(band_rate&0x00ff);
    SCI1->SCICR2 |= (1<<2)|(1<<3)|(1<<5);
    NVIC_Init(0, 0, SCI1_IRQn, 2);
}

void SCI1_IRQHandler(void)
{
    u8 i = 0,u8Tmp;
    if ((SCI1->SCISR1 & 0x20)== 0x20)
    {
        u8Tmp = SCI1->SCIDRL;
        //User's Code...
    }
}
```

配置为 GPIO：

```
UART_ConfigGpio(SCI1,UART_TX,GPIO_OUTPUT);     //TXD1 配置为输出
UART_ConfigGpio(SCI1,UART_RX,GPIO_OUTPUT);     //RXD1 配置为输出

UART_WriteGpioData(SCI1,UART_TX,1);            //TXD1 输出高电平
UART_WriteGpioData(SCI1,UART_TX,0);            //TXD1 输出低电平

UART_ConfigGpio(SCI1,UART_RX,GPIO_INPUT);      //RXD1 配置为输入
Temp = UART_ReadGpioData(SCI1,UART_RX);        //读 RXD1 状态
```

Levetop Semiconductor

© Levetop Semiconductor Co, Ltd.

4. ADC 端口

对应芯片脚位 Pin15-Pin16，此组端口只能配置为 ADC 输入，不能配置为 GPIO 功能。ADC 配置函数：

```
#define ADCCH_0          0
#define ADCCH_1          8
void ADC_Initial(void)
{
    ADC->CFGR2 |= ((10-1)<<8);              //10MHz/10=1MHz
    ADC->CFGR2 |= 0x20;                     //启动计数位
    ADC->CFGR1 &=~(1<<20);                  //不连续转换
    ADC->CFGR1 &=~(7<<24);                  //序列长度为 1 单位
    ADC->CFGR1 &=~(3<<8);                   //分辨率:12
    ADC->CFGR1 &=~(1<<10);                  //右对齐
    ADC->SMPR = 0x20;                       //采样时间
}
u16 Get_ADC_Val(u8 ADC_Channel_x)
{
    u32 adcisr = 0,i;
    u16 Val = 0;
    ADC_Initial();
    ADC->CHSELR1 = ADC_Channel_x;
    ADC_Cmd(ADC_EN);
    for (i = 0; i < 4; i++){
        ADC_StartConv();
        DelayMS(1);
        while (EOSEQ !=(ADC->ISR & EOSEQ));
        ADC->ISR |= EOSEQ;
        ADC_StopConv();
        if(i == 0){                         //第一个数丢掉
            adcisr = (ADC->uFIFO_DAT.FIFO)&0xFFFF;
            adcisr = 0;
            continue;
        }
        adcisr += (ADC->uFIFO_DAT.FIFO)&0xFFFF;
    }
    ADC_Cmd(ADC_DIS);
    return ((adcisr/3)&0x0fff);
}
Temp = Get_ADC_Val(ADCCH_0);                //读 AIN0 状态
Temp = Get_ADC_Val(ADCCH_1);                //读 AIN1 状态
```


© Levetop Semiconductor Co, Ltd.

Levetop Semiconductor

5. DAC 端口

对应芯片脚位 Pin18，此端口默认是 DAC 输出，可以配置为输出高低电平。

DAC 音频输出配置参数串口屏工程代码的音频输出部分函数。
GPIO 输出高低电平函数配置：

```
DAC_Init(RIGHTALIGNED_12BITS, TRIGGER_SOFTWARE, DET_ON_RISING);
Send_DAC_data(4095);          //DAC_OUT 输出高电平
Send_DAC_data(0);             //DAC_OUT 输出低电平
```

6. WAKEUP 端口

对应芯片脚位 Pin20，此端口默认是休眠唤醒功能，可配置为输入检测功能。

当配置为休眠唤醒功能，需要接 GND，外接高电平唤醒；当配置为输入功能时，接 10K 电阻到 VCC，可以检测低电平输入。

```
If ((CPM->CPM_PADWKINTCR&0x02000000)==0)
{
    Temp   = 1;                 //检测到低电平，标志位置 1
}
```

7. IIC 端口

对应芯片脚位 Pin25-Pin26，此组端口默认是 IIC 功能，可配置为 GPIO 功能。
配置为 GPIO 函数：

```
I2C_ConfigGpio(I2C_SDA,GPIO_OUTPUT);        //SDA 配置为输出
I2C_ConfigGpio(I2C_SCL,GPIO_OUTPUT);        //SCL 配置为输出

I2C_WriteGpioData(I2C_SDA,1);               //SDA 输出高电平
I2C_WriteGpioData(I2C_SDA,0);               //SDA 输出低电平

I2C_ConfigGpio(I2C_SCL,GPIO_INPUT);         //SCL 配置为输入
Temp = I2C_ReadGpioData(I2C_SCL);           //读 SCL 状态
```

配置为 IIC 功能，GPIO 模拟 IIC 功能参考串口屏工程代码的音 CTP 采集程序，硬件 IIC 功能参考 i2c_drv.c 中的驱动函数。

© Levetop Semiconductor Co, Ltd.

Levetop Semiconductor

8. GINT[1-5]/UART[2-3] 端口

对应芯片脚位 Pin30-Pin34，此组端口默认是 GINT 功能，其中 Pin30-Pin32、Pin34 可以配置复用为 UART2、UART3，可以全部配置，或者单独配置一个端口。

配置此组端口为 GPIO 或者外部中断函数例程：

```
EPORT_ConfigGpio(EPORT_PIN1, GPIO_OUTPUT);      //GINT1 配置为输出
EPORT_ConfigGpio(EPORT_PIN2, GPIO_OUTPUT);      //GINT2 配置为输出

EPORT_WriteGpioData(EPORT_PIN1, 1);             //GINT1 输出高电平
EPORT_WriteGpioData(EPORT_PIN1, 0);             //GINT1 输出高电平

EPORT_ConfigGpio(EPORT_PIN3, GPIO_INPUT);       //GINT3 配置为输入
Temp = EPORT_ReadGpioData(EPORT_PIN3);          //读 GINT3 状态

EPORT_ConfigGpio(EPORT_PIN4, GPIO_INPUT);       //GINT4 配置为输入
EPORT_Init(EPORT_PIN4, RISING_EDGE_INT);        //GINT4 配置外部中断，上升沿触发
void EPORT0_4_IRQHandler(void)
{
    EPORT->EPFR |= 0x01<< 4;                    //Clear flag by writing 1 to it.
    //User's Code...
}
```

Pin30-Pin32, Pin34 复用为 UART2, UART3 说明：

```
SCI_CONTROL_REG[29:24]--SCI_GINT_SWAP[5:0]      //对应寄存器，SCI 和 GINT 复用使能信号
                                                //0 = 不复用（GINT 功能生效）
                                                //1 = 复用（SCI 功能生效）

//六个 bit 对应 3 路串口：
    bit0: sci1 rxd
    bit1: sci3 txd
    bit2: sci3 rxd
    bit3: sci2 rxd
    bit4: sci1 txd
    bit5: sci2 txd
```

注意：脚位是 GINT 时用 EPORT 对应函数配置软件功能，是串口时用 UART 对应函数配置软件功能。

© Levetop Semiconductor Co, Ltd.

Pin30:TXD3, Pin31:RXD3, Pin32:RXD2, Pin34:TXD2 默认是 GINT 功能，可以配置对应寄存器的 bit1-bit3/bit5 复用为串口功能。

```
IOCTRL->SCI_CONTROL_REG |= (((u32)0x2E) << 24);

//UART 端口配置为 GPIO:
UART_ConfigGpio(SCI3,UART_TX,GPIO_OUTPUT);      //TXD3 配置为输出
UART_ConfigGpio(SCI3,UART_RX,GPIO_OUTPUT);      //RXD3 配置为输出
UART_ConfigGpio(SCI2,UART_TX,GPIO_OUTPUT);      //TXD2 配置为输出
UART_ConfigGpio(SCI2,UART_RX,GPIO_OUTPUT);      //RXD2 配置为输出

UART_WriteGpioData(SCI2,UART_TX,1);             //TXD2 输出高电平
UART_WriteGpioData(SCI2,UART_TX,0);             //TXD2 输出低电平

UART_ConfigGpio(SC2,UART_RX,GPIO_INPUT);        //RXD2 配置为输入
Temp = UART_ReadGpioData(SCI2,UART_RX);         //读 RXD2 状态
```

9. SWD 端口

对应芯片脚位 Pin13:SWDIO, Pin43:SWDCLK，此组端口默认是 SWD 下载/调试功能，可以配置复用为 PWM 和 EPORT，当配置复用功能后 SWD 功能不可用。配置复用的函数如下：

```
void SWD_Enable_Cmd(bool index)
{
    if (index == TRUE){
        IOCTRL->SWAP_CONTROL_REG |= (1<<1);
    }
    else{
        IOCTRL->SWAP_CONTROL_REG &= ~(1<<1);
    }
}
SWD_Enable_Cmd(TRUE);       //TRUE：使能 SWD 功能
SWD_Enable_Cmd(FALSE);      //FALSE：SWD 功能关闭，开启 GINT 或 PWM 功能
```

配置为 Eport 功能：

```
EPORT_ConfigGpio(EPORT_PIN14, GPIO_OUTPUT);     //GINT14 配置为输出
EPORT_ConfigGpio(EPORT_PIN15, GPIO_OUTPUT);     //GINT15 配置为输出

EPORT_WriteGpioData(EPORT_PIN14, 1);            //GINT14 输出高电平
EPORT_WriteGpioData(EPORT_PIN14, 0);            //GINT14 输出低电平
```


© Levetop Semiconductor Co, Ltd.

Levetop Semiconductor

```
EPORT_ConfigGpio(EPORT_PIN15, GPIO_INPUT);        //GINT15 配置为输入
Temp = EPORT_ReadGpioData(EPORT_PIN15);           //读 GINT15 状态
```

配置为 PWM 功能：

```
PWM_OutputInit(PWM_PORT2,2,PWM_CLK_DIV_1,4,2,0);        //配置 Pin13 为 PWM2 功能
```

Levetop Semiconductor

10. GINT1[3]/PWM1[1] 端口

对应芯片脚位 Pin2，此组端口默认是输出 PWM 信号作为 LT7689 显示驱动部分的晶振使用，当 Pin94、Pin95 接外部晶振电路时，此端口可做为 EPORT 或 PWM 使用，可参考上一章的配置函数。

11. LT768_PWM1，LT768_PWM0 端口

对应芯片脚位 Pin60-Pin61，此组端口是 LT7689 显示驱动部分的 PWM 输出，可以用来控制背光输出。配置函数如下：

```
void LT768_PWM1_Init
(
    u8_t on_off,              // 0: Disable PWM0; 1: Enable PWM0
    u8_t Clock_Divided,       // PWM clock division; Value range 0~3(1,1/2,1/4,1/8)
    u8_t Prescalar,           // Clock division; Value range 1~256
    u16_t Count_Buffer,       // PWM output period
    u16_t Compare_Buffer      // Duty cycle
);
```

PWM 频率 = 系统时钟/((2^Clock_Divided)*(Prescalar)*(Count_Buffer)) //2^Clock_Divided 是 2 的次方

LT768x 系统时钟 = CCLK; //内核时钟频率.

例程 1，PWM 频率 5KHz，占空比分别为 80%

LT768_PWM1_Init(1,0,200,100,80);

例程 2，PWM 频率 25KHz，占空比分别为 50%

LT768_PWM1_Init(1,0,5,800,400);

注意：

1. LT768_PWM0 是显示驱动部分的测试脚位，不能接高电平。
2. 当显示驱动部分在更新显存数据时，LT768_PWMX 不能修改参数，所有 LT768_PWMX 只能用来控制 PWM 参数变动较少的电路，如果背光之类，不建议控制参数变动较多的电路，如蜂鸣器之类。

Levetop Semiconductor

© Levetop Semiconductor Co, Ltd.

12. PD[0-2]，PD[8-9]，PD[16-18] 端口

对应芯片脚位 Pin68-Pin70；Pin76-Pin77；Pin85-Pin87，此组端口默认 RGB 数据的低 8 位，当 RGB 数据配置为 RGB565 输出时，此组端口可配置为 GPIO 功能。

GPIOD[7:0] 对应 PD[18, 2, 17, 16, 9, 8, 1, 0]

GPIO 配置函数：

```
void Set_GPIO_D_In_Out(unsigned char temp);
//GPIO-D_dir[7:0] : General Purpose I/O direction control.
    0: Output        1: Input
```

参考例程 1：

配置 PD0-2, PD8 输出高低电平，

```
Set_GPIO_D_In_Out(0x00);        //GPIOD[0-7] 输出
u8_t TTemp = 0xff;
/*RST 接 PD0 既 GPIOD0*/
#define  LCD_RST_RESET          {TTemp &=~0x01 ; Write_GPIO_D_7_0(TTemp);}
#define  LCD_RST_SET            {TTemp |=0x01 ; Write_GPIO_D_7_0(TTemp);}
/*CS 接 PD1 既 GPIOD1*/
#define  LCD_CS_RESET           {TTemp &=~0x02 ; Write_GPIO_D_7_0(TTemp);}
#define  LCD_CS_SET             {TTemp |=0x02 ; Write_GPIO_D_7_0(TTemp);}
/*SCL 接 PD2 既 GPIOD6*/
#define  LCD_SCL_RESET          {TTemp &=~0x40 ; Write_GPIO_D_7_0(TTemp);}
#define  LCD_SCL_SET            {TTemp |=0x40 ; Write_GPIO_D_7_0(TTemp);}
/*SDA 接 PD8 既 GPIOD3*/
#define  LCD_SDA_RESET          {TTemp &=~0x04 ; Write_GPIO_D_7_0(TTemp);}
#define  LCD_SDA_SET            {TTemp |=0x04 ; Write_GPIO_D_7_0(TTemp);}
```

参考例程 2：

配置 PD9 输入，

```
Set_GPIO_D_In_Out(0x08);        //GPIOD[3]输入，其他为输出
TTemp = Read_GPIO_D_7_0()&0x08;
```


© Levetop Semiconductor Co, Ltd.

Levetop Semiconductor

13. PIT 定时器

LT7689 芯片内部有 2 组 PIT 定时器。参考例程如下：

```
void PIT1_Init(void)                    //10MS PIT 中断函数
{
    NVIC_Init(1, 0, PIT1_IRQn, 2);
    PIT1->PCSR &= (~PCSR_EN);
    PIT1->PCSR = (5 << 8) | PCSR_OVW | PCSR_PIE | PCSR_RLD | PCSR_PDBG;
    #if OSC_Frequency                   //150M 主频
    PIT1->PMR = 23437;                  //10ms
    #else                               //120M 主频
    PIT1->PMR = 18749;                  //10ms
    #endif
    PIT1->PCSR |= PCSR_EN;
}

void PIT1_IRQHandler(void)
{
    PIT1->PCSR |= (1 << 2);             //Clear PIF interrupt flag
    //User's Code...
}
```


© Levetop Semiconductor Co, Ltd.

14. EFlash 应用说明

芯片内部 Flash 有 512KBytes，程序设置的 EFlash 地址要在 504KBytes (0x0807E000)之前 (最后 8KBytes 保存了部分系统数据）。参考例程如下：

```
#define flh_sAddr 0x0807D800
u32 DATA[5] = {x};
voidSaveData(void)
{
      u8_tbuff[20] = {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(g_sys_clk / 1000);
      EFLASH_SetWritePermission();
      reVal = EFLASH_Write(flh_sAddr, buff, 20);
      EFLASH_ClrWritePermission();
}

u8_t LT_TpGetAdjdata(void)
{
      u8_t i, buff[20] = {0};
      for (i = 0; i < 17; i++)
            buff[i] = EFLASH_ByteRead(flh_sAddr + i);

      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;
}
```

Levetop Semiconductor

© Levetop Semiconductor Co, Ltd.