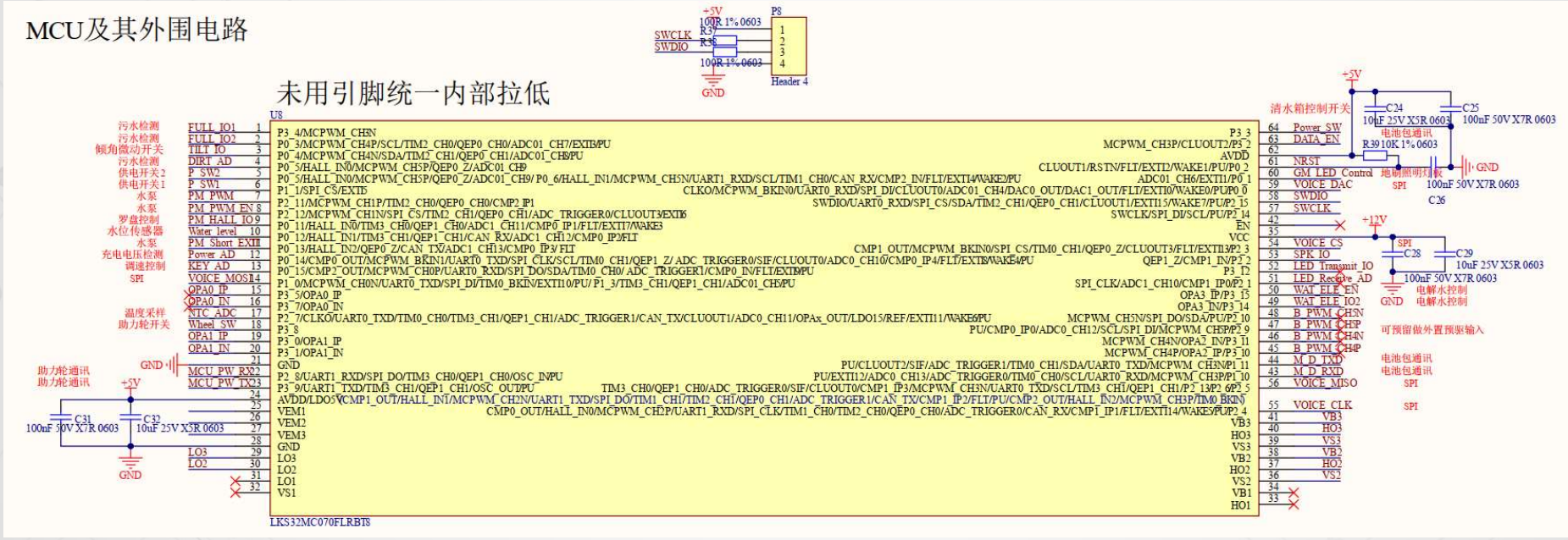
基于LKS32MC07x的洗地机主控解决方案

洗地机主控方案特点

- MCU集成了预驱，可直接驱动地刷电机；
- 两路DAC，一路可做音频信号的输出，另一路可做地刷电机的过流保护。
- 集成多路比较器，可做多路硬件过流保护。
- 运放为差分运放，抗干扰性强，电流采样信号好。
- 集成LDO5V、运放、比较器、DAC，集成程度高，节省BOM成本；

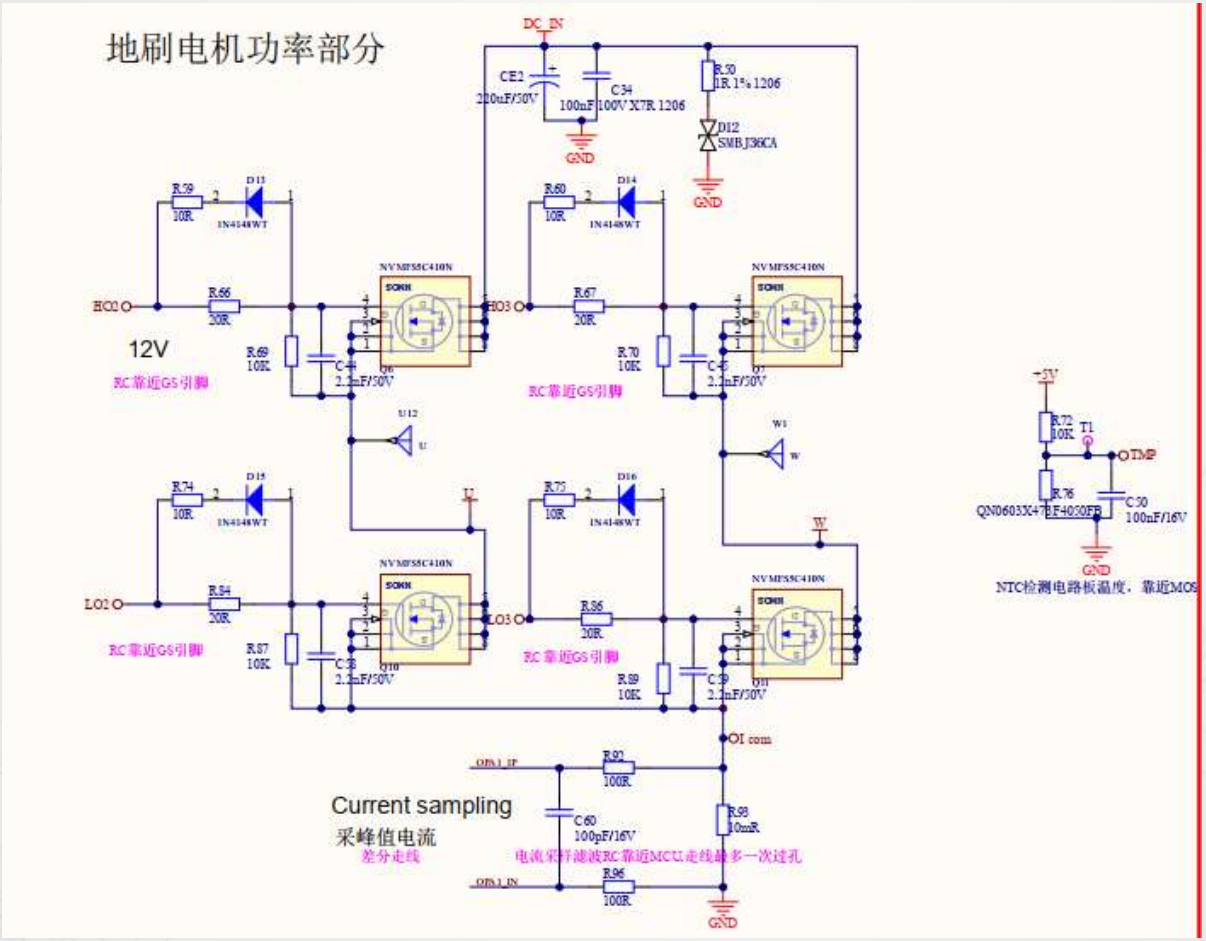
洗地机主控方案硬件电路设计 - 1

- MCU使用LKS32MC070FLRBT8



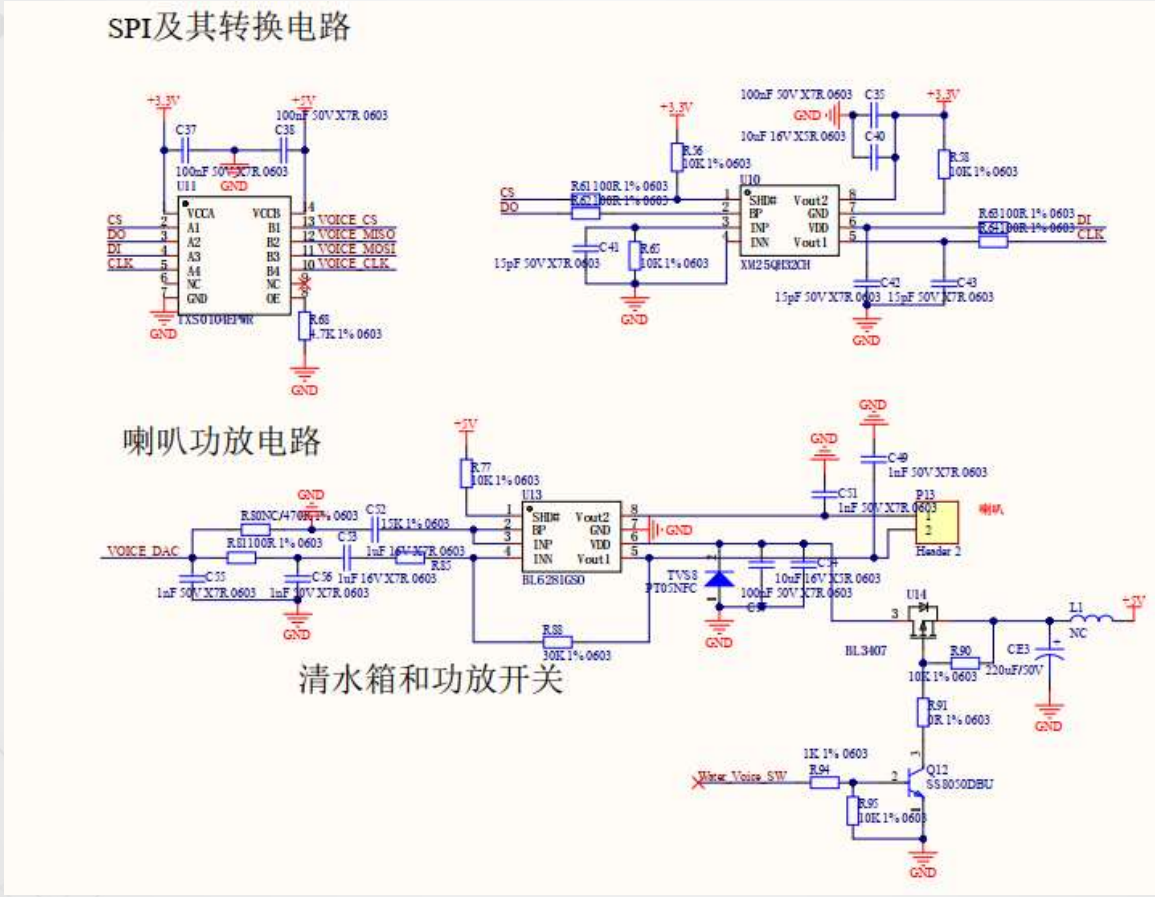
洗地机主控硬件电路设计 - 2

- 地刷电机驱动部分



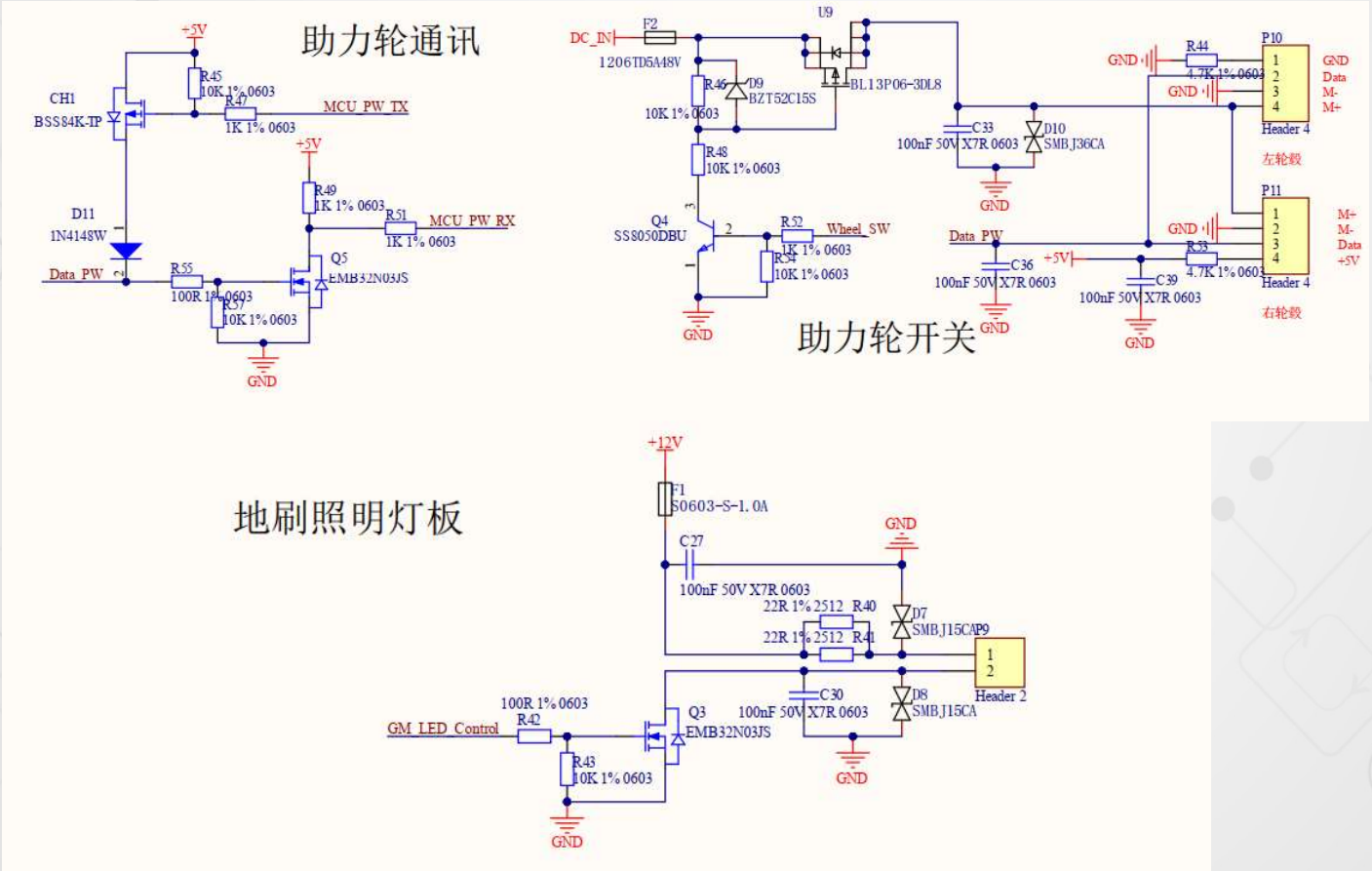
洗地机主控硬件电路设计 - 3

- SPI通讯电路，喇叭功放电路，清水箱和功放开关。



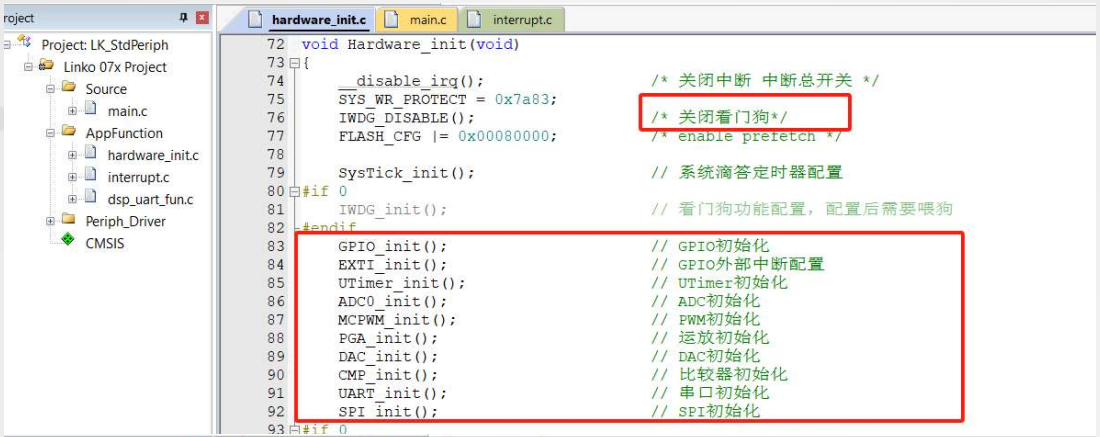
洗地机主控硬件电路设计 - 4

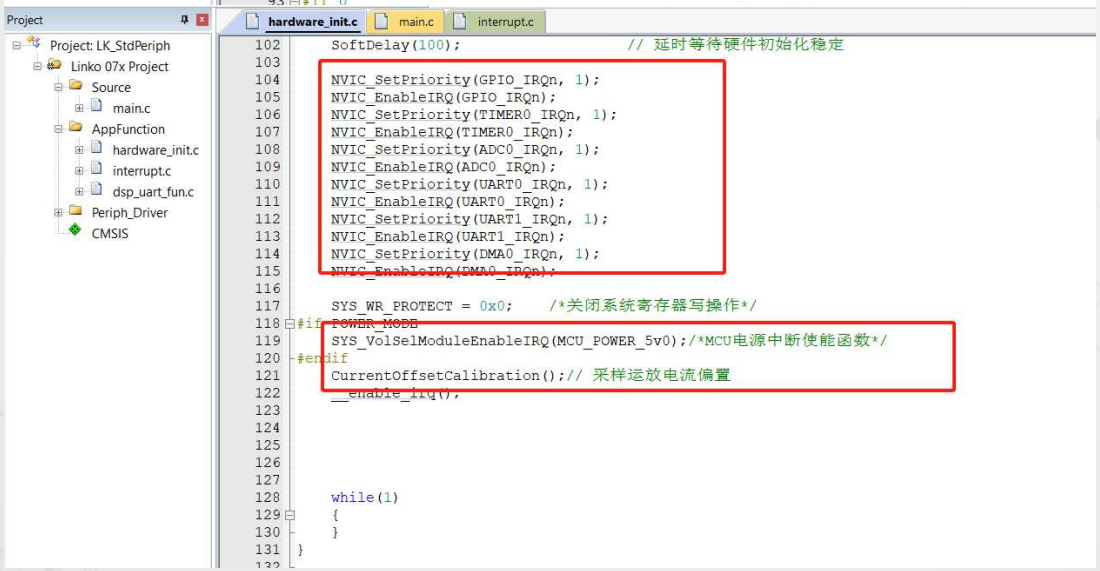
- 助力轮通讯电路，地刷板照明灯电路。



洗地机主控方案软件框架

- 各模块初始化，中断优先级配置和使能，MCU欠压检测中断使能，运放电流偏置采样。





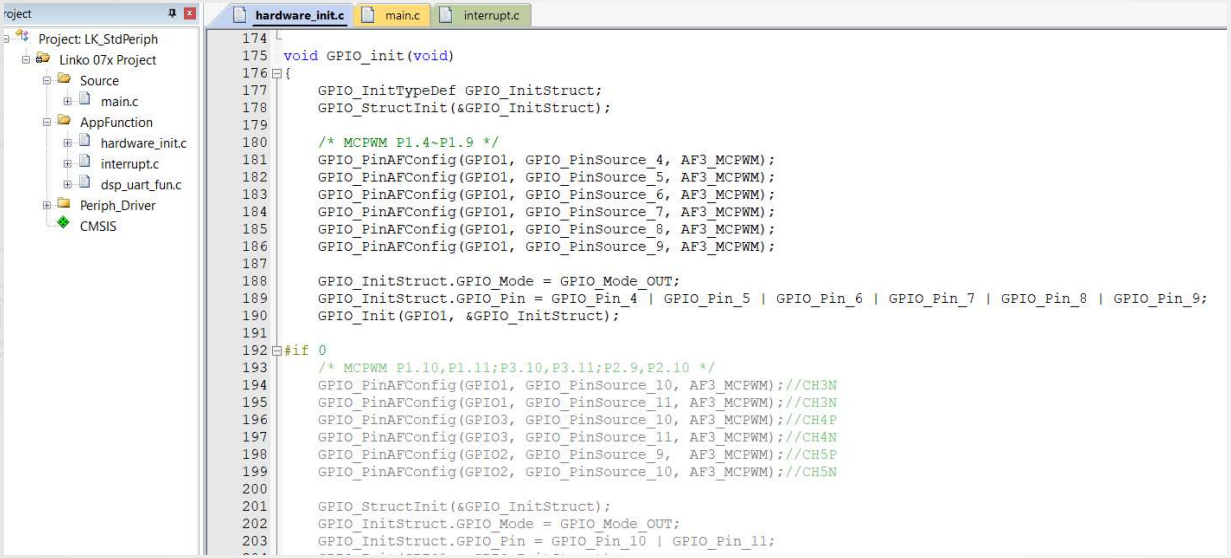
洗地机主控方案程序介绍 - 1

- 看门狗模块初始化，在200ms时间间隔内喂狗，防止程序跑飞。

```
162  void IWDG_init(void)// 200ms看门狗复位
163 {
164
165      IWDG_InitTypeDef IWDG_InitStruct;
166      IWDG_StrutInit(&IWDG_InitStruct);
167      IWDG_InitStruct.WDG_EN  = ENABLE;       //使能看门狗
168      IWDG_InitStruct.RTH     = 6400;//设置看门狗
169      IWDG_InitStruct.DWK_EN  = DISABLE;      // 深度休眠定时唤醒关闭
170      IWDG_InitStruct.WTH     = 0;            // 看门狗定时唤醒时间（21位计数器，但低12恒位0）
171      IWDG_Init(&IWDG_InitStruct);
172      IWDG_Feed();//看门狗喂狗
173 }
```

洗地机主控方案程序介绍 - 2

- GPIO模块初始化，GPIO外部中断配置。




```
230 void EXTI_init(void)
231 {
232     GPIO_InitTypeDef GPIO_InitStruct;
233     GPIO_StructInit(&GPIO_InitStruct);
234
235     GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
236     GPIO_InitStruct.GPIO_Pin = GPIO_Pin_14;
237     GPIO_Init(GPIO0, &GPIO_InitStruct);
238     EXTI_Trigger_Config(GPIO0, GPIO_PinSource_14, EXTI_Trigger_Edge);// GPIO外部中断
239 }
```

洗地机主控方案程序介绍 - 3

- 定时器模块初始化，嘀嗒定时器做1ms时基用，TIMER模块输出PWM控制水泵电机。

```
154  void SysTick_init(void)
155 {
156
157      SysTick->LOAD  = 96000;
158      SysTick->VAL   = 0;
159      SysTick->CTRL = 0x07;
160  }
161
```

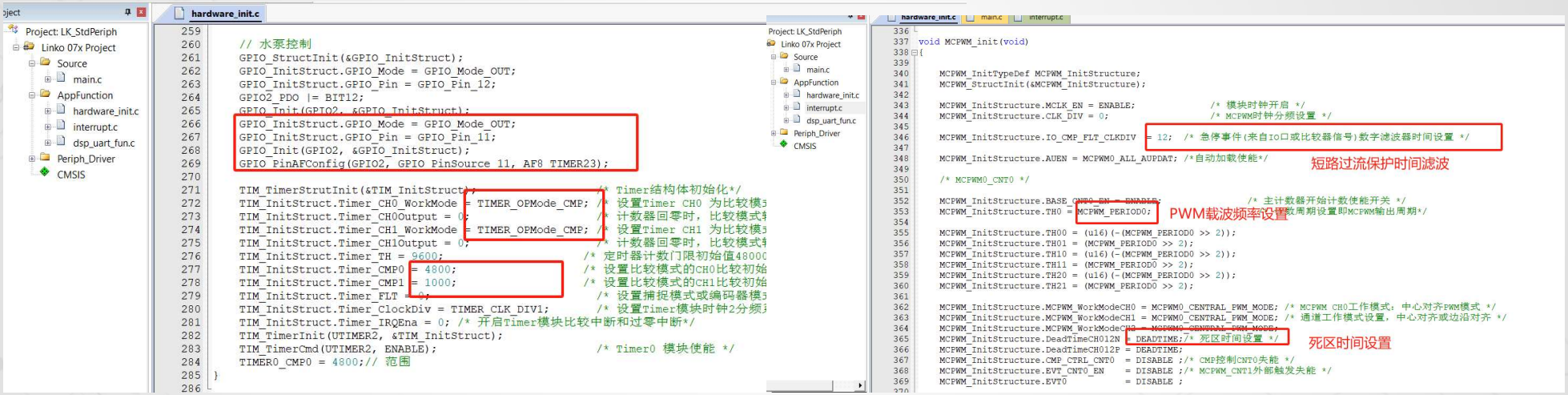
```
241  void UTimer_init(void)
242 {
243      TIM_TimerInitTypeDef TIM_InitStruct;
244      GPIO_InitTypeDef GPIO_InitStruct;
245
246      TIM_TimerStrutInit(&TIM_InitStruct);                    /* Timer结构体初始化*/
247      TIM_InitStruct.Timer_CH0_WorkMode = TIMER_OPMode_CMP; /* 设置Timer CH0 为比较模式 */
248      TIM_InitStruct.Timer_CH0Output = 0;                     /* 计数器回零时，比较模式输出极性控制 */
249      TIM_InitStruct.Timer_CH1_WorkMode = TIMER_OPMode_CMP; /* 设置Timer CH1 为比较模式 */
250      TIM_InitStruct.Timer_CH1Output = 0;                     /* 计数器回零时，比较模式输出极性控制 */
251      TIM_InitStruct.Timer_TH = 12000;                        /* 定时器计数门限初始值48000*/
252      TIM_InitStruct.Timer_CMP0 = 1000;                     /* 设置比较模式的CH0比较初始值24000 */
253      TIM_InitStruct.Timer_CMP1 = 1000;                     /* 设置比较模式的CH1比较初始值24000 */
254      TIM_InitStruct.Timer_FLT = 0;                           /* 设置捕捉模式或编码器模式下对应通道的数字滤波值 */
255      TIM_InitStruct.Timer_ClockDiv = TIMER_CLK_DIV1;         /* 设置Timer模块时钟2分频系数 */
256      TIM_InitStruct.Timer_IRQEna = Timer_IRQEna_ZC; /* 开启Timer模块比较中断和过零中断*/
257      TIM_TimerInit(UTIMER0, &TIM_InitStruct);
258      TIM_TimerCmd(UTIMER0, ENABLE);                          /* Timer0 模块使能 */
259
260      // 水泵控制
261      GPIO_StructInit(&GPIO_InitStruct);
262      GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
263      GPIO_InitStruct.GPIO_Pin = GPIO_Pin_12;
264      GPIO2_PDO |= BIT12;
265      GPIO_Init(GPIO2, &GPIO_InitStruct);
266      GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
267      GPIO_InitStruct.GPIO_Pin = GPIO_Pin_11;
268      GPIO_Init(GPIO2, &GPIO_InitStruct);
269      GPIO_PinAFConfig(GPIO2, GPIO_PinSource_11, AF8_TIMER23);
270
271      TIM_TimerStrutInit(&TIM_InitStruct);                    /* Timer结构体初始化*/
```

洗地机主控方案程序介绍 - 4

- ADC模块初始化：用作ADC采样，配置一段触发模式，电压/电流的采样，开启ADC中断，在中断里读取ADC采样值，读取清水箱电压值做有水无水判断，地刷电流做过流保护。

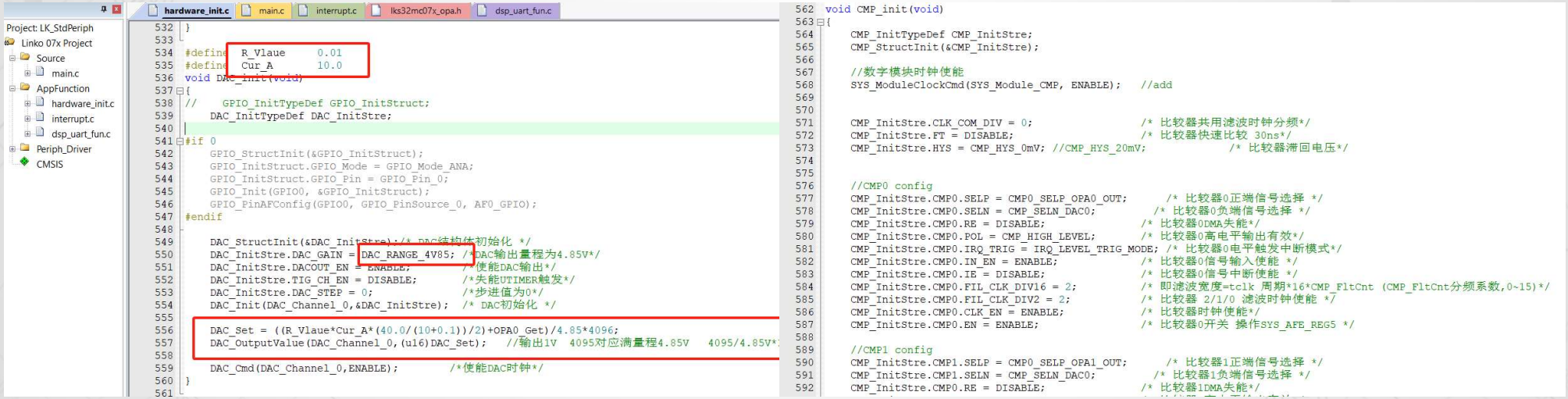
洗地机主控方案程序介绍 - 5

- MCPWM模块初始化：输出中心对齐 MCPWM 来驱动地刷电机， 用 H 桥驱动地刷电机换向控制。



洗地机主控方案程序介绍 - 6

- 硬件过流保护：
- DAC模块初始化， DAC使能，配置DAC满量程为4.85V，采样电阻为0.01欧姆，保护电流设置10A，公式计算后给定DAC值。
- 比较器模块初始化，打开比较器，配置比较器0负端接DAC0，正端接OPA0的输出，实现硬件过流保护功能。



洗地机主控方案程序介绍－7

- UART串口通讯模块初始化：实现串口DMA发送，中断接收。用于与助力轮通讯，与主电机通讯。
- 如果两路串口不够用，可以用DSP模拟一路串口。

```c
void UART_init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    DMA_InitTypeDef DMA_InitStruct;
    UART_InitTypeDef UART_InitStruct;

    GPIO_StructInit(&GPIO_InitStruct);
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_15;
    GPIO_Init(GPIO0, &GPIO_InitStruct);
    GPIO_StructInit(&GPIO_InitStruct);
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0;
    GPIO_Init(GPIO1, &GPIO_InitStruct);
    GPIO_PinAFConfig(GPIO0, GPIO_PinSource_15, AF4_UART);
    GPIO_PinAFConfig(GPIO1, GPIO_PinSource_0, AF4_UART);

    UART_StructInit(&UART_InitStruct);
    UART_InitStruct.BaudRate = 9600;                    /* 设置波特率38400 */
    UART_InitStruct.WordLength = UART_WORDLENGTH_8b;  /* 发送数据长度8位 */
    UART_InitStruct.StopBits = UART_STOPBITS_1b;
    UART_InitStruct.FirstSend = UART_FIRSTSEND_LSB;    /* 先发送LSB */
    UART_InitStruct.ParityMode = UART_Parity_NO;       /* 无奇偶校验 */
    UART_InitStruct.IRQEna = UART_IRQEna_RcvOver;// 接受中断
    UART_Init(UART0, &UART_InitStruct);

    // 一轮由DMA串口发送10个字节数据
    DMA_StructInit(&DMA_InitStruct);
    DMA_InitStruct.EN = ENABLE;                                 // DMA通道使能
    DMA_InitStruct.IRQ_EN = ENABLE;                             // DMA 传输完成中断使能
    DMA_InitStruct.SINC = ENABLE;                               // 源地址递增，  高有效,地址按照 SBTW 对应大小递增 1/2/4
```

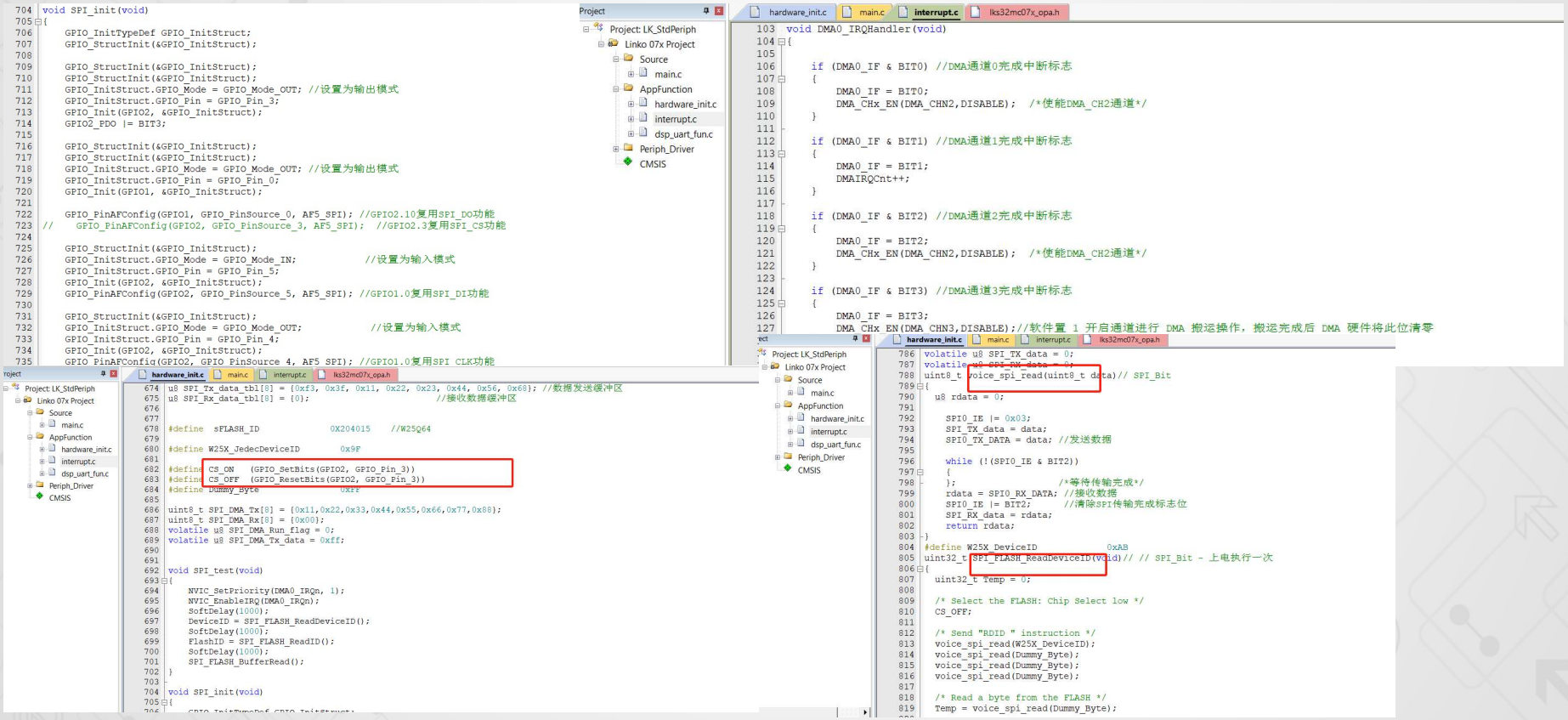
```c
u8 Uart_Buff = 0;
void UART0_IRQHandler(void)
{

    if (UART_GetIRQFlag(UART0, UART_IF_SendOver))     // 发送完成中断
    {
        UART_ClearIRQFlag(UART0, UART_IF_SendOver);  // 清除发送完成标志位
    }
    if (UART_GetIRQFlag(UART0, UART_IF_RcvOver))      // 接收完成中断
    {
        UART_ClearIRQFlag(UART0, UART_IF_RcvOver);   // 清除接收完成标志位
        Uart_Buff = UART_ReadData(UART0);       // 接收 1 Byte数据
    }
}

void UART1_IRQHandler(void)
{

    if (UART_GetIRQFlag(UART1, UART_IF_SendOver))     // 发送完成中断
    {
        UART_ClearIRQFlag(UART1, UART_IF_SendOver);  // 清除发送完成标志位
    }
    if (UART_GetIRQFlag(UART1, UART_IF_RcvOver))      // 接收完成中断
    {
        UART_ClearIRQFlag(UART1, UART_IF_RcvOver);   // 清除接收完成标志位
        Uart_Buff = UART_ReadData(UART1);       // 接收 1 Byte数据
    }
}
```

```c
void dsp_uart_tx(int8_t data)
{
    DSP0_SC |= BIT1;
    DSP0_SC |= BIT3;

    DSP_MEM0->BRR_OP = data;
    DSP0_PC = 0x30;
    DSP_MEM0->DATA[0x30] = 0; // clear uart0 rx done flag
    DSP_MEM0->DATA[0x31] = 0; // clear uart0 rx done flag
    DSP0_SC = BIT0 | BIT8;
}
uint8_t rx_data[32];
volatile uint8_t uart_flag = 0;
void dsp_uart_rx(int8_t data)
{
    static u16 p = 0;

    rx_data[p] = data;
    p++;
    if (p >= 32)
    {
        p = 0;
    }
    DSP0_SC |= BIT1;
    DSP0_SC |= BIT3;
//    DSP_MEM0->DATA[0] = (DSP_MEM0->DATA[0]& 0xffff0000) | BIT9;
    DSP0_PC = 0x0;
    DSP_MEM0->DATA[0x30] = 0; // clear uart0 rx done flag
    DSP_MEM0->DATA[0x31] = 0; // clear uart0 rx done flag
    DSP0_SC = BIT0 | BIT8;
}
char t[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
u8 p = 0;
```

```c
void dsp_fun_init(void)
{
    UINT32 i;
    SYS_ModuleClockCmd(SYS_Module_DSP0, ENABLE);
    SYS_ModuleClockCmd(SYS_Module_GPIO, ENABLE);

    DSP0_SC |= BIT1;
    DSP0_SC |= BIT3;
    // dsp data mem flush
    for (i = 0; i < 60; i += 1)
    {
        DSP_MEM0->DATA[i] = 0;
    }
    // dsp code mem flush
    for (i = 0; i < 512; i += 1)
    {
        DSP_MEM0->CODE[i] = 0;
    }
    // dsp data mem init
    for (i = 0; i < (sizeof(test_data_array) / 4); i += 1)
    {
        DSP_MEM0->DATA[i] = test_data_array[i];
    }
    for (i = 0; i < (sizeof(test_code_array) / 2); i += 1)
    {
        DSP_MEM0->CODE[i] = test_code_array[i];
    }

//    GPIO0_PDO = BIT14;
    GPIO0_POE = BIT14; // DSP TX
    GPIO1_PIE = BIT0;  // DSP RX
    NVIC_EnableIRQ(DSP0_IRQn);
    __enable_irq();
```

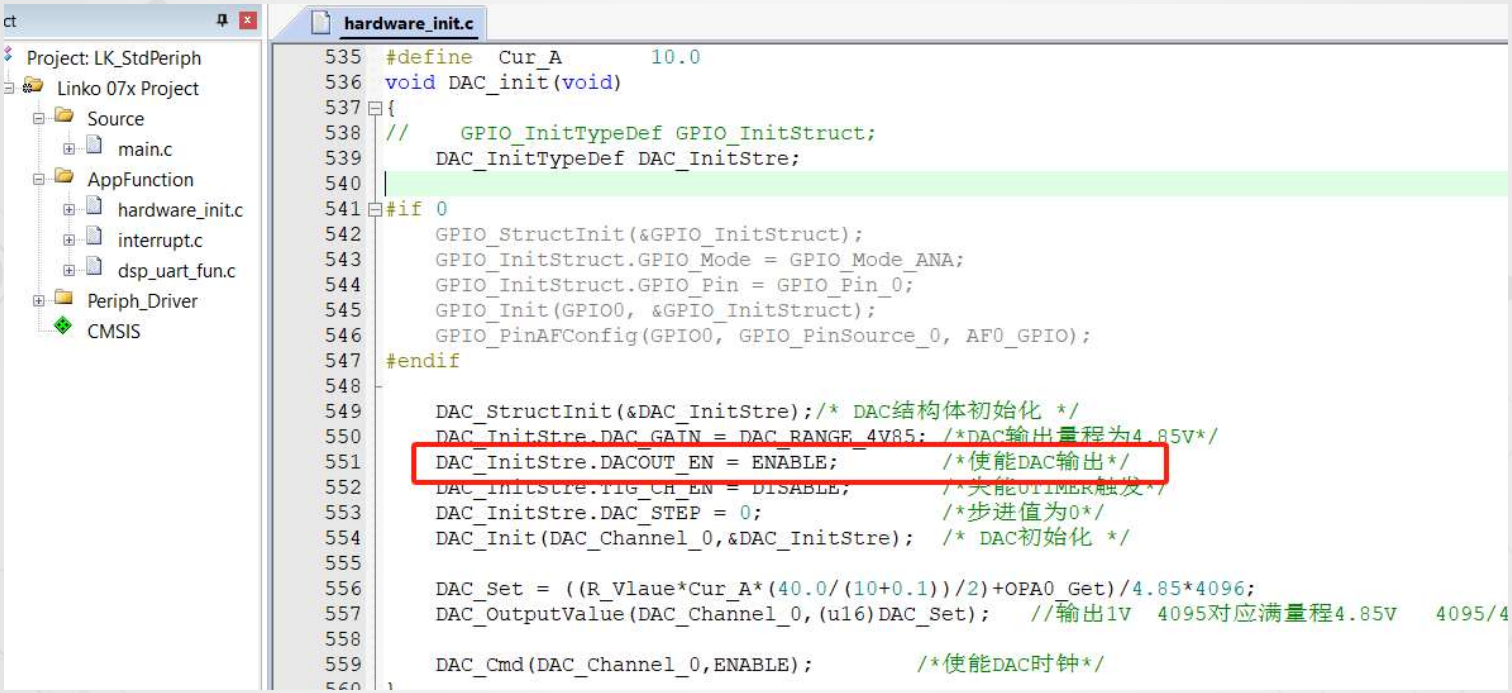
洗地机主控方案程序介绍－8

- 外置语音模块SPI通讯，实现MCU与语音外设模块的SPI通讯，DMA发送和接收。
- 通讯测试程序，读取SPI_FLASH外设的ID，判断有没通讯成功。



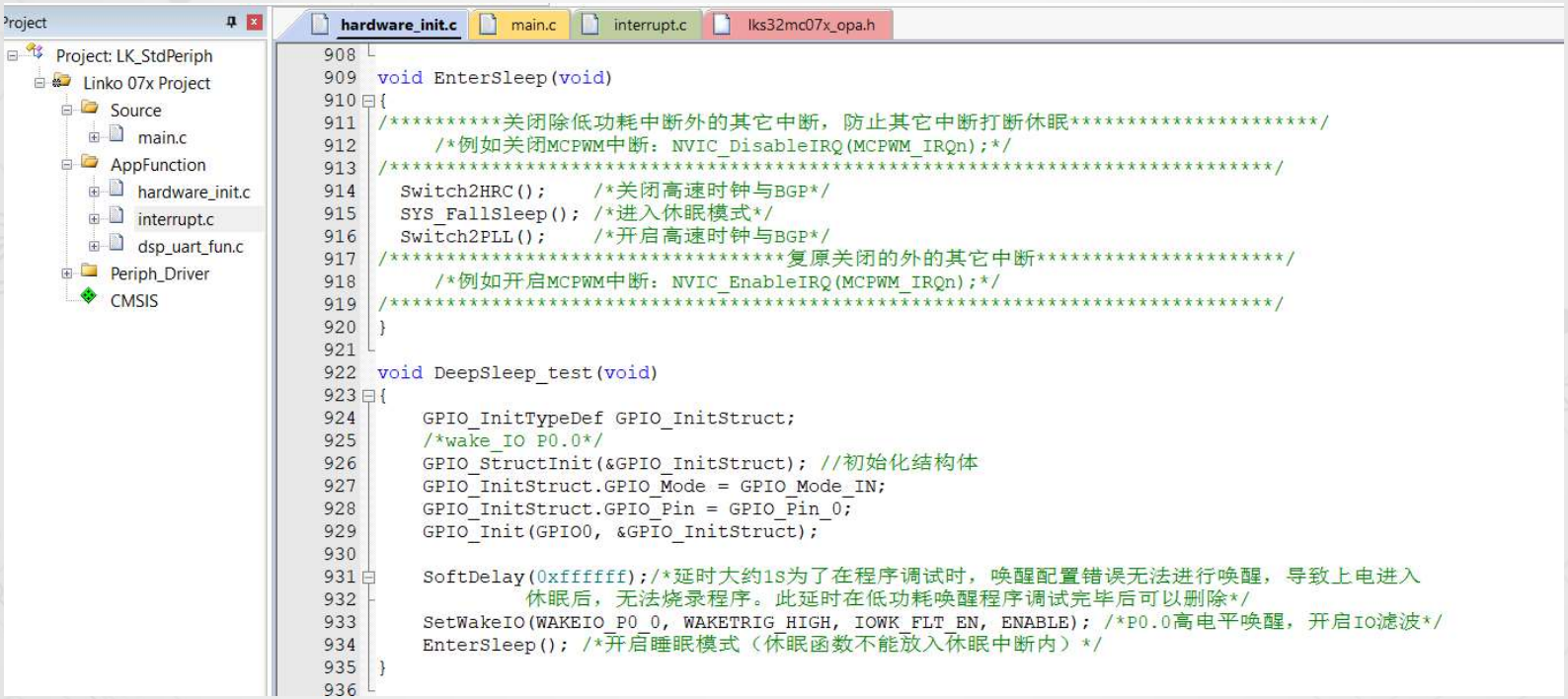
洗地机主控方案程序介绍 – 9

- DAC输出语音信号



洗地机主控方案程序介绍 – 10

- 低功耗模式配置，通过外部IO高电平唤醒实现退出低功耗模式。




```
908
909 void EnterSleep(void)
910 {
911 /**********关闭除低功耗中断外的其它中断，防止其它中断打断休眠**********************/
912     /*例如关闭MCPWM中断：NVIC_DisableIRQ(MCPWM_IRQn);*/
913 /*******************************************************************************/
914   Switch2HRC();    /*关闭高速时钟与BGP*/
915   SYS_FallSleep(); /*进入休眠模式*/
916   Switch2PLL();    /*开启高速时钟与BGP*/
917 /**********************************复原关闭的外的其它中断**********************/
918     /*例如开启MCPWM中断：NVIC_EnableIRQ(MCPWM_IRQn);*/
919 /*******************************************************************************/
920 }
921
922 void DeepSleep_test(void)
923 {
924     GPIO_InitTypeDef GPIO_InitStruct;
925     /*wake_IO P0.0*/
926     GPIO_StructInit(&GPIO_InitStruct); //初始化结构体
927     GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
928     GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0;
929     GPIO_Init(GPIO0, &GPIO_InitStruct);
930
931     SoftDelay(0xffffff);/*延时大约1S为了在程序调试时，唤醒配置错误无法进行唤醒，导致上电进入
932             休眠后，无法烧录程序。此延时在低功耗唤醒程序调试完毕后可以删除*/
933     SetWakeIO(WAKEIO_P0_0, WAKETRIG_HIGH, IOWK_FLT_EN, ENABLE); /*P0.0高电平唤醒，开启IO滤波*/
934     EnterSleep(); /*开启睡眠模式（休眠函数不能放入休眠中断内）*/
935 }
936
```

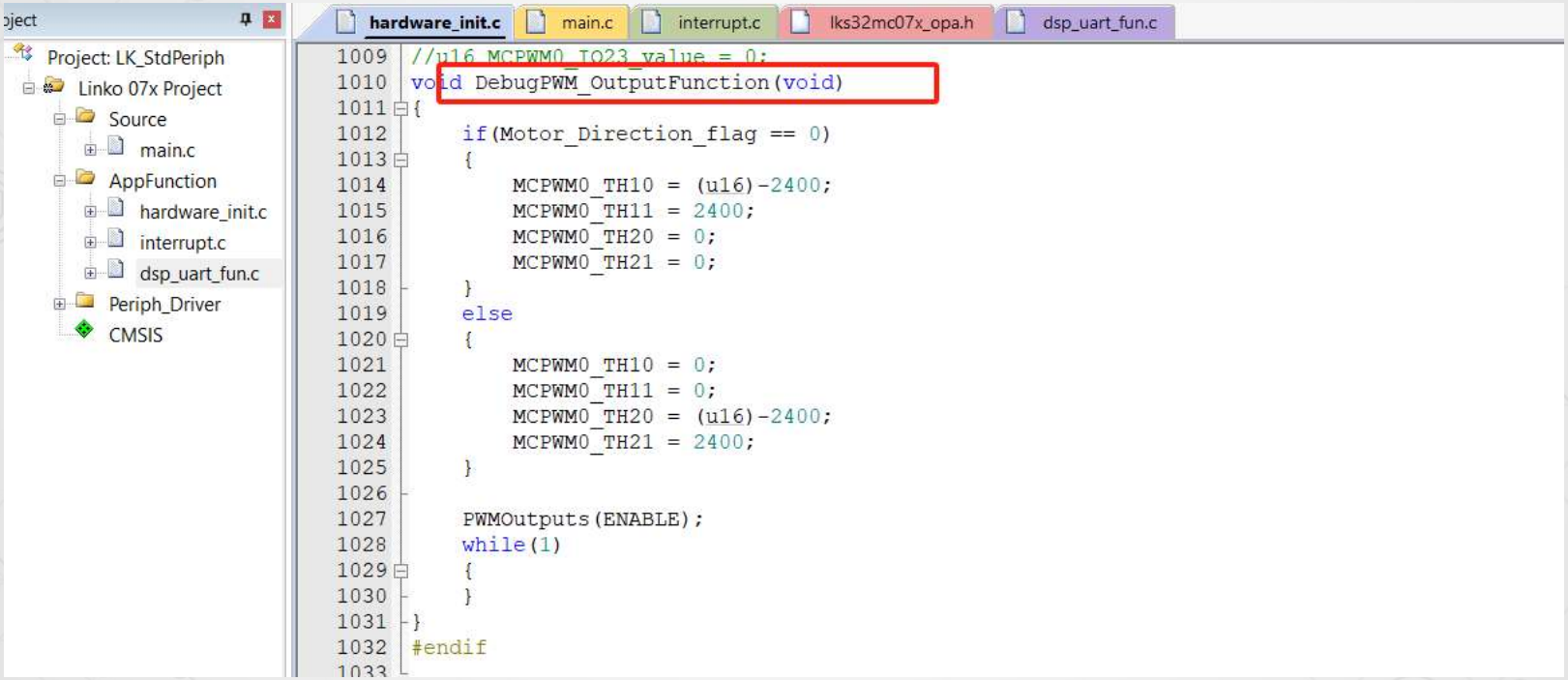
洗地机主控方案程序介绍 – 11

- FLASH掉电存储功能

```
u8 Flash_Write_Buff[256] = {0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88};
u32 Flash_Read_Buff[64] = {0x00};
void Flash_Write(void)
{
#if 1
    erase_flag = 0x9A0D361F;//擦除解锁
    EraseSector(0x00000000 , Flash_NVR);//擦除flash的main区域即存储程序区域
    erase_flag = 0x00;//擦除上锁

    progm_flag = 0x9AFDA40C;//编程解锁
    ProgramPage(0x00000000 ,256, Flash_Write_Buff, Flash_NVR);//编程flash的main区域即存储程序区域
    erase_flag = 0x00;      //编程上锁
#else

    erase_flag = 0x9A0D361F;//擦除解锁
    EraseSector(0x0001FC00 , Flash_MAIN);//擦除flash的main区域即存储程序区域
    erase_flag = 0x00;//擦除上锁

    progm_flag = 0x9AFDA40C;//编程解锁
    ProgramPage(0x0001FC00 ,256, Flash_Write_Buff, Flash_MAIN);//编程flash的main区域即存储程序区域
    erase_flag = 0x00;      //编程上锁
#endif
}

void Flash_Read(void)
{

    Read_More_Flash(0x0,2,Flash_Read_Buff,Flash_NVR);
}
```

洗地机主控方案程序介绍 – 12

- 测试硬件驱动电路模式，Debug模式，通过输出25%占空比，检验硬件PWM通路是否正常。



创芯驱动，领航电控未来！

正直诚信！利他共赢！成长超越！

江苏省南京市经济技术开发区兴智科技园B栋15层
http://www.linkosemi.com