



# 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

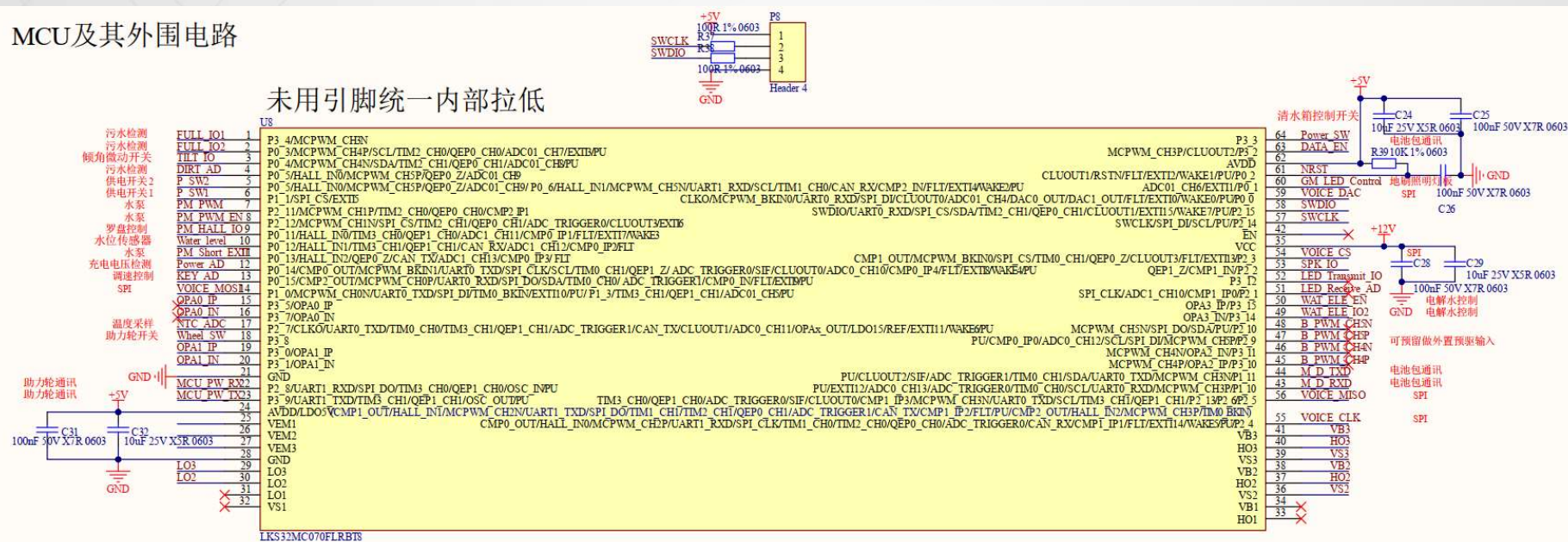
凌鸥创芯

围绕产业链 · 打造创新链

## ➤ MCU使用LKS32MC070FLRBT8

### MCU及其外围电路

未用引脚统一内部拉低





### 地刷电机功率部分





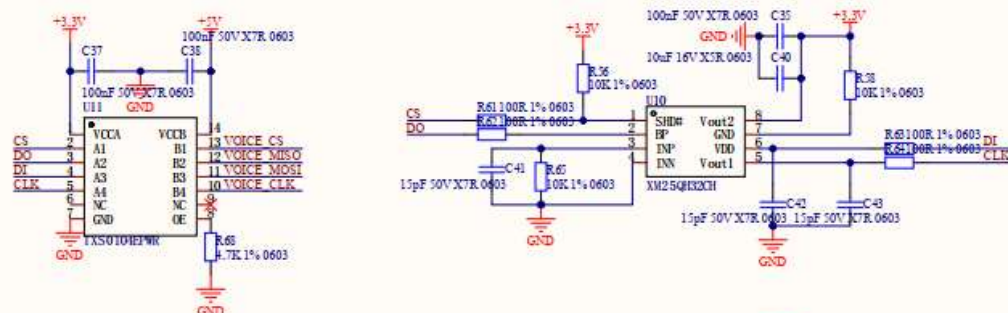
## 洗地机主控硬件电路设计 - 3

凌鸥创芯

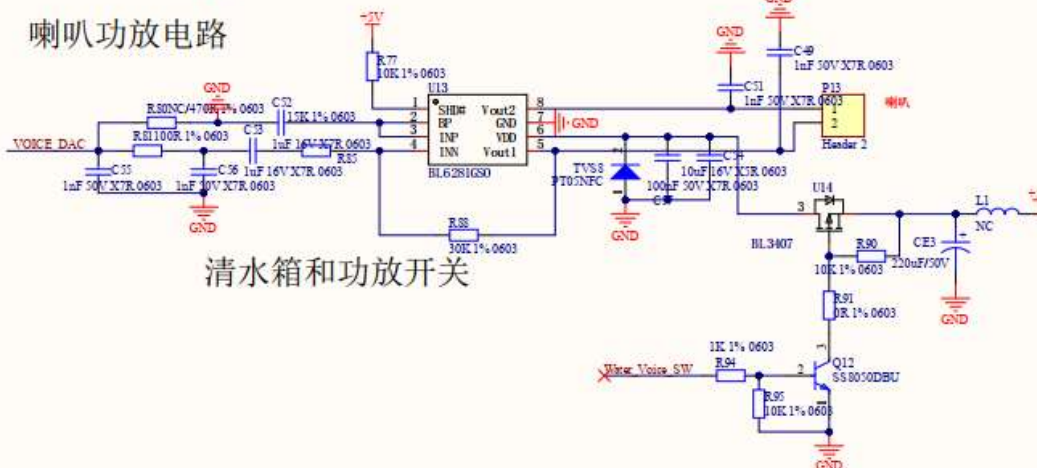
围绕产业链 · 打造创新链

- SPI通讯电路，喇叭功放电路，清水箱和功放开关。

SPI及其转换电路



喇叭功放电路



清水箱和功放开关



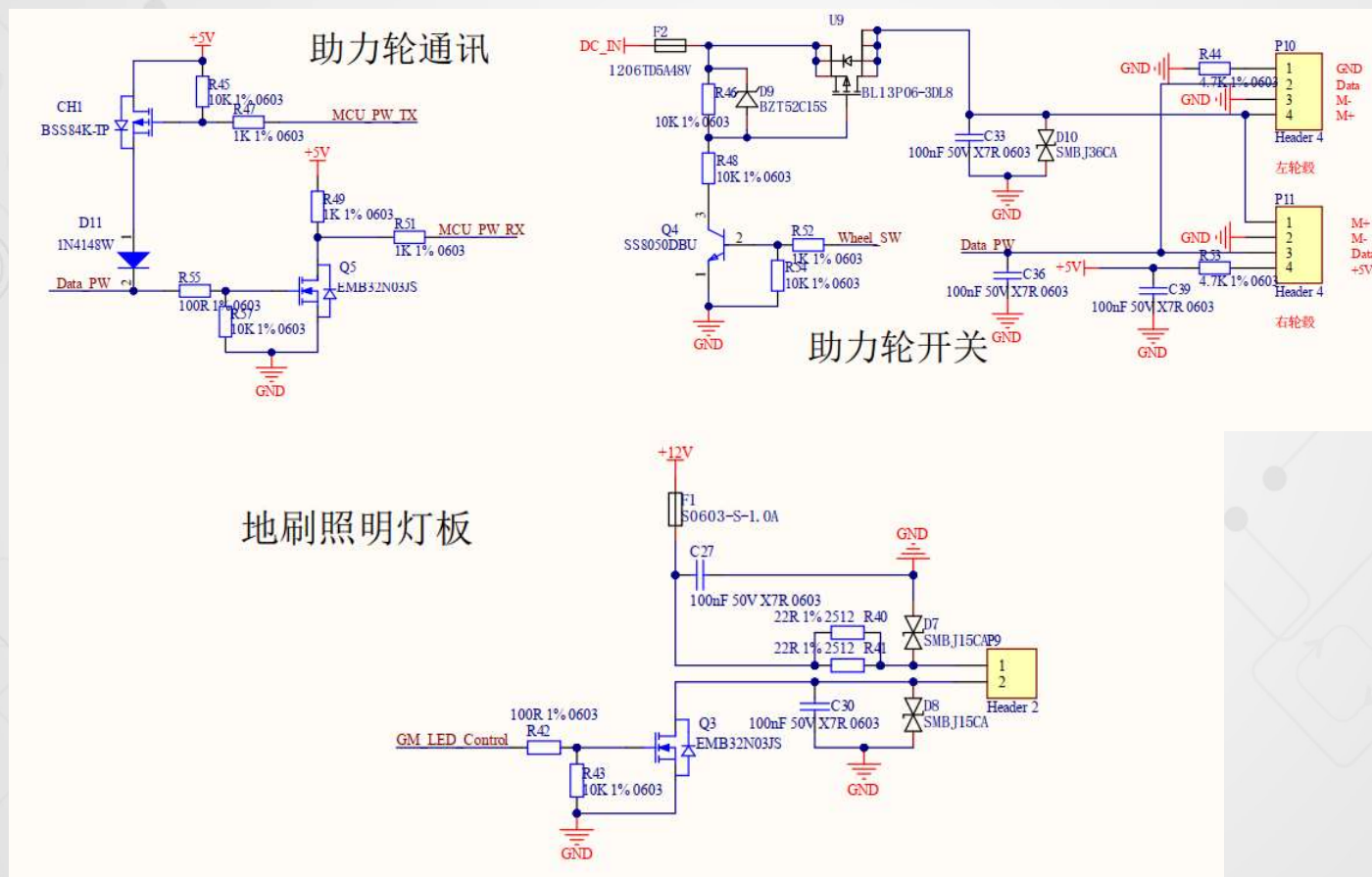


## 洗地机主控硬件电路设计 - 4

凌鸥创芯

围绕产业链 · 打造创新链

- 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

凌鸥创芯

围绕产业链 · 打造创新链

- 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。

```
project
├── Project: LK_StdPeriph
│   ├── Linko 07x Project
│   │   ├── Source
│   │   │   ├── main.c
│   │   │   ├── AppFunction
│   │   │   │   ├── hardware_init.c
│   │   │   │   ├── interrupt.c
│   │   │   │   ├── dsp_uart_fun.c
│   │   │   └── Periph_Driver
│   │   └── CMSIS
└── hardware_init.c
    72 void Hardware_init(void)
    73 {
    74     __disable_irq();          /* 关闭中断 中断总开关 */
    75     SYS_WR_PROTECT = 0x7a83;  /* 关闭看门狗 */
    76     IWDG_DISABLE();          /* enable preetch */
    77     FLASH_CFG |= 0x00080000;
    78
    79     SysTick_init();           // 系统滴答定时器配置
    80 #if 0
    81     IWDG_init();              // 看门狗功能配置，配置后需要喂狗
    82 #endif
    83     GPIO_init();              // GPIO初始化
    84     EXTI_init();              // GPIO外部中断配置
    85     UTimer_init();            // UTimer初始化
    86     ADC0_init();              // ADC初始化
    87     MCPWM_init();             // PWM初始化
    88     PGA_init();               // 运放初始化
    89     DAC_init();               // DAC初始化
    90     CMP_init();               // 比较器初始化
    91     UART_init();              // 串口初始化
    92     SPI_init();               // SPI初始化
    93 #if 0
    102     SoftDelay(100);           // 延时等待硬件初始化稳定
    103
    104     NVIC_SetPriority(GPIO_IRQn, 1);
    105     NVIC_EnableIRQ(GPIO_IRQn);
    106     NVIC_SetPriority(TIMERO_IRQn, 1);
    107     NVIC_EnableIRQ(TIMERO_IRQn);
    108     NVIC_SetPriority(ADC0_IRQn, 1);
    109     NVIC_EnableIRQ(ADC0_IRQn);
    110     NVIC_SetPriority(UART0_IRQn, 1);
    111     NVIC_EnableIRQ(UART0_IRQn);
    112     NVIC_SetPriority(UART1_IRQn, 1);
    113     NVIC_EnableIRQ(UART1_IRQn);
    114     NVIC_SetPriority(DMA0_IRQn, 1);
    115     NVIC_EnableIRQ(DMA0_IRQn);
    116
    117     SYS_WR_PROTECT = 0x0;      /* 关闭系统寄存器写操作 */
    118 #if POWER_MODE
    119     SYS_VolSelModuleEnableIRQ(MCU_POWER_5v0); /* MCU电源中断使能函数 */
    120 #endif
    121     CurrentOffsetCalibration(); // 采样运放电流偏置
    122     __enable_irq();
    123
    124
    125
    126
    127
    128     while(1)
    129     {
    130     }
    131 }
    132
```



## 洗地机主控方案程序介绍 - 1

凌鸥创芯

围绕产业链 · 打造创新链

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



## 洗地机主控方案程序介绍 - 2

凌鸥创芯

围绕产业链 · 打造创新链

### ➤ GPIO模块初始化，GPIO外部中断配置。

```
Project: LK_StdPeriph
Linko 07x Project
├── Source
│   ├── main.c
│   ├── AppFunction
│   │   ├── hardware_init.c
│   │   ├── interrupt.c
│   │   └── dsp_uart_func.c
│   └── Periph_Driver
│       └── CMSIS
└── hardware_init.c
    174
    175 void GPIO_init(void)
    176 {
    177     GPIO_InitTypeDef GPIO_InitStructure;
    178     GPIO_StructInit(&GPIO_InitStructure);
    179
    180     /* MCPWM P1.4~P1.9 */
    181     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_4, AF3_MCPWM);
    182     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_5, AF3_MCPWM);
    183     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_6, AF3_MCPWM);
    184     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_7, AF3_MCPWM);
    185     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_8, AF3_MCPWM);
    186     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_9, AF3_MCPWM);
    187
    188     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    189     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8 | GPIO_Pin_9;
    190     GPIO_Init(GPIO1, &GPIO_InitStructure);
    191
    192     #if 0
    193     /* MCPWM P1.10,P1.11;P3.10,P3.11;P2.9,P2.10 */
    194     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_10, AF3_MCPWM); //CH3N
    195     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_11, AF3_MCPWM); //CH3N
    196     GPIO_PinAFConfig(GPIO3, GPIO_PinSource_10, AF3_MCPWM); //CH4P
    197     GPIO_PinAFConfig(GPIO3, GPIO_PinSource_11, AF3_MCPWM); //CH4N
    198     GPIO_PinAFConfig(GPIO2, GPIO_PinSource_9, AF3_MCPWM); //CH5P
    199     GPIO_PinAFConfig(GPIO2, GPIO_PinSource_10, AF3_MCPWM); //CH5N
    200
    201     GPIO_StructInit(&GPIO_InitStructure);
    202     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    203     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_11;
    204
    230 void EXTI_init(void)
    231 {
    232     GPIO_InitTypeDef GPIO_InitStructure;
    233     GPIO_StructInit(&GPIO_InitStructure);
    234
    235     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
    236     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_14;
    237     GPIO_Init(GPIO0, &GPIO_InitStructure);
    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

凌鸥创芯

围绕产业链 · 打造创新链

- 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     SysTick->LOAD = 96000;
157     SysTick->VAL = 0;
158     SysTick->CTRL = 0x07;
159 }
160
161
241 void UTimer_init(void)
242 {
243     TIM_TimrInitTypeDef TIM_InitStruct;
244     GPIO_InitTypeDef GPIO_InitStruct;
245
246     TIM_TimrStrutInit(&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_2C; /* 开启Timer模块比较中断和过零中断 */
257     TIM_TimrInit(UTIMER0, &TIM_InitStruct);
258     TIM_TimrCmd(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_TimrStrutInit(&TIM_InitStruct); /* Timer结构体初始化*/
```



## 洗地机主控方案程序介绍 - 4

凌鸥创芯

围绕产业链 · 打造创新链

- 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。

```
hardware_init.c
286
287 u8 Adc_buff[6]={0x11,0x22,0x33,0x44,0x55,0x66};
288 u8 Adc_buff2[6]={0x00};
289 void Adc0_init(void)
290 {
291     ADC_InitTypeDef ADC_InitStructure;
292     DMA_InitTypeDef DMA_InitStructure;
293
294     ADC_StructInit(&ADC_InitStructure);
295     ADC_InitStructure.IE      = ADC_SF1_IE;           // 中断使能
296     ADC_InitStructure.RE      = 1;                   // DMA请求使能
297     ADC_InitStructure.NSMP    = DISABLE;             // 两段采样使能
298     ADC_InitStructure.DATA_ALIGN = DISABLE;          // DAT右对齐使能
299     ADC_InitStructure.CSMP    = DISABLE;             // 连续采样使能
300     ADC_InitStructure.TCMT    = 0;                   // 触发一次采样所需的事件数
301     ADC_InitStructure.TROVS   = DISABLE;             // 手动触发过采样使能，开启后一次采样需要多次触发
302     ADC_InitStructure.OVSR    = 0;                   // 过采样率
303     ADC_InitStructure.TRIG    = ADC_TRIG_MCPWM0_T0;   // 触发信号
304     ADC_InitStructure.S1      = 6;                   // 第一段常规采样次数
305     ADC_InitStructure.S2      = 0;                   // 第二段常规采样次数
306     ADC_InitStructure.IS1     = 1;                   // 空闲采样次数
307     ADC_InitStructure.LTH     = 0;                   // ADC 模拟看门狗 0 下阈值
308     ADC_InitStructure.HTH     = 0;                   // ADC 模拟看门狗 0 上阈值
309     ADC_InitStructure.GEN     = 0;                   // ADC 模拟看门狗 0 对应使能位
310     ADC_Init(ADC0, &ADC_InitStructure);
311     if (0)
312     {
313         ADC_ClearIRQFlag(ADC0, ADC_ALL_IF); //清除所有中断标志位
314         ADC_ClearIRQFlag(ADC1, ADC_ALL_IF);
315     }
316     ADC_CHN_GAIN_CFG(ADC0, CHN0, ADC_CHANNEL_0, ADC_GAIN3V6);
317     ADC_CHN_GAIN_CFG(ADC0, CHN1, ADC_CHANNEL_1, ADC_GAIN3V6);
318 }

interrupt.c
43 volatile s16 OPA0_ADC;
44 volatile s16 OPA1_ADC;
45 extern volatile s16 OPA0_adc_offset;
46 extern volatile s16 OPA1_adc_offset;
47 extern volatile s32 OPA0_adc_offset_sum;
48 extern volatile s32 OPA1_adc_offset_sum;
49 volatile u16 OPA_Offset_Cnt;
50 void ADC0_IRQHandler(void) // 10KHz的时基
51 {
52     ADC0_IF = BIT1 | BIT0;
53     if((MCPWM0_EIF & BIT4) || (MCPWM0_EIF & BIT5)) // MCPWM_FALL事件 硬件过流判断
54     {
55         MCPWM0_EIF = BIT4 | BIT5;
56     }
57
58     OPA0_ADC = (ADC0_DAT0 - OPA0_adc_offset)>>3; // 采样电阻*母线电流*运放倍数=ADC的电压，ADC电压/3.6*4096=ADC值
59     OPA1_ADC = (ADC0_DAT1 - OPA1_adc_offset)>>3; // 运放倍数与运放配置的电阻和外部电阻有关，现在配置的是40K/10K，如果外部电阻是1K，运放倍率=40.0K/1K=40
```



## 洗地机主控方案程序介绍 - 5

凌鸥创芯

围绕产业链 · 打造创新链

- 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。

```
Project LK_StdPeriph
Linko 07x Project
Source
main.c
AppFunction
hardware_init.c
interrupt.c
dsp_uart_func.c
Periph_Driver
CMSIS

259 // 水泵控制
260 GPIO_StructInit(&GPIO_InitStructure);
261 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
262 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
263 GPIO2_PDO |= BIT12;
264 GPIO_Init(GPIO2, &GPIO_InitStructure);
265
266 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
267 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
268 GPIO_Init(GPIO2, &GPIO_InitStructure);
269 GPIO_PinAFConfig(GPIO2, GPIO_PinSource_11, AF8_TIMER23);
270
271 TIM_TimStructInit(&TIM_InitStructure); // Timer结构体初始化*/
272 TIM_InitStructure.Tim_CH0_WorkMode = TIMER_OpMode_CMP; // 设置Timer CH0 为比较模式
273 TIM_InitStructure.Tim_CH0Output = 0; // 计数器回零时，比较模式
274 TIM_InitStructure.Tim_CH1_WorkMode = TIMER_OpMode_CMP; // 设置Timer CH1 为比较模式
275 TIM_InitStructure.Tim_CH1Output = 0; // 计数器回零时，比较模式
276 TIM_InitStructure.Tim_TH = 9600; // 定时器计数门限初始值48000
277 TIM_InitStructure.Tim_CMP0 = 4800; // 设置比较模式的CH0比较初始
278 TIM_InitStructure.Tim_CMP1 = 1000; // 设置比较模式的CH1比较初始
279 TIM_InitStructure.Tim_FLT = 0; // 设置捕捉模式或编码器模式
280 TIM_InitStructure.Tim_ClockDiv = TIMER_CLK_DIV1; // 设置Timer模块时钟2分频
281 TIM_InitStructure.Tim_IRQEna = 0; // 开启Timer模块比较中断和过零中断*/
282 TIM_TimInit(TIM2, &TIM_InitStructure);
283 TIM_Cmd(TIM2, ENABLE); // Timer0 模块使能 */
284 TIM2_CMP0 = 4800; // 范围
285
286
336 void MCPWM_init(void)
337 {
338     MCPWM_InitTypeDef MCPWM_InitStructure;
339     MCPWM_StructInit(&MCPWM_InitStructure);
340
341     MCPWM_InitStructure.MCLK_EN = ENABLE; // 模块时钟开启 */
342     MCPWM_InitStructure.CLK_DIV = 0; // MCPWM时钟分频设置 */
343
344     MCPWM_InitStructure.IO_CMP_FLT_CLKDIV = 12; // 急停事件(来自IO口或比较器信号)数字滤波器时间设置 */
345
346     MCPWM_InitStructure.AUEN = MCPWM0_ALL_AUEN; // 自动加载使能 */
347
348     /* MCPWM0_CNT0 */
349
350     MCPWM_InitStructure.BASE_CNT0_EN = ENABLE; // 主计数器开始计数使能开关 */
351     MCPWM_InitStructure.TH0 = MCPWM_PERIOD0; // PWM载波频率设置 载波周期设置即MCPWM输出周期 */
352
353     MCPWM_InitStructure.TH00 = (u16)((MCPWM_PERIOD0 >> 2));
354     MCPWM_InitStructure.TH01 = (u16)((MCPWM_PERIOD0 >> 2));
355     MCPWM_InitStructure.TH10 = (u16)((MCPWM_PERIOD0 >> 2));
356     MCPWM_InitStructure.TH11 = (u16)((MCPWM_PERIOD0 >> 2));
357     MCPWM_InitStructure.TH20 = (u16)((MCPWM_PERIOD0 >> 2));
358     MCPWM_InitStructure.TH21 = (u16)((MCPWM_PERIOD0 >> 2));
359
360     MCPWM_InitStructure.MCPWM_WorkModeCH0 = MCPWM0_CENTRAL_PWM_MODE; // MCPWM CH0工作模式：中心对齐PWM模式 */
361     MCPWM_InitStructure.MCPWM_WorkModeCH1 = MCPWM0_CENTRAL_PWM_MODE; // 通道工作模式设置，中心对齐或边沿对齐 */
362     MCPWM_InitStructure.MCPWM_WorkModeCH2 = MCPWM0_CENTRAL_PWM_MODE;
363     MCPWM_InitStructure.DeadTimeCH012N = DEADTIME; // 死区时间设置 */
364     MCPWM_InitStructure.DeadTimeCH012P = DEADTIME; // 死区时间设置
365     MCPWM_InitStructure.CMP_CTRL_CNT0 = DISABLE; // CMP控制CNT0失能 */
366     MCPWM_InitStructure.EVT_CNT0_EN = DISABLE; // MCPWM_CNT0外部触发失能 */
367     MCPWM_InitStructure.EVT0 = DISABLE;
368 }
```



## 洗地机主控方案程序介绍 - 6

凌鸥创芯

围绕产业链 · 打造创新链

- 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。

```
Project: LK_StdPeriph
Linko 07x Project
Source
main.c
AppFunction
hardware_init.c
interrupt.c
dsp_uart_func.c
Periph_Driver
CMSIS

532 }
533
534 #define R_Vlaue 0.01
535 #define Cur_A 10.0
536 void DAC_init(void)
537 {
538     // GPIO_InitTypeDef GPIO_InitStructure;
539     DAC_InitTypeDef DAC_InitStructure;
540
541     if 0
542     {
543         GPIO_StructInit(&GPIO_InitStructure);
544         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_ANA;
545         GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
546         GPIO_Init(GPIO0, &GPIO_InitStructure);
547         GPIO_PinAFConfig(GPIO0, GPIO_PinSource_0, AF0_GPIO);
548     }
549     DAC_StructInit(&DAC_InitStructure); /* DAC结构体初始化 */
550     DAC_InitStructure.DAC_GAIN = DAC_RANGE_4V85; /* DAC输出量程为4.85V */
551     DAC_InitStructure.DACOUT_EN = ENABLE; /* 使能DAC输出 */
552     DAC_InitStructure.TIG_CH_EN = DISABLE; /* 失能UTIMER触发 */
553     DAC_InitStructure.DAC_STEP = 0; /* 步进值为0 */
554     DAC_Init(DAC_Channel_0, &DAC_InitStructure); /* DAC初始化 */
555
556     DAC_Set = ((R_Vlaue*Cur_A*(40.0/(10+0.1))/2)+OPA0_Get)/4.85*4096;
557     DAC_OutputValue(DAC_Channel_0, (u16)DAC_Set); /* 输出1V 4095对应满量程4.85V 4095/4.85V */
558
559     DAC_Cmd(DAC_Channel_0, ENABLE); /* 使能DAC时钟 */
560 }
561
```

```
562 void CMP_init(void)
563 {
564     CMP_InitTypeDef CMP_InitStructure;
565     CMP_StructInit(&CMP_InitStructure);
566
567     //数字模块时钟使能
568     SYS_ModuleClockCmd(SYS_Module_CMP, ENABLE); //add
569
570
571     CMP_InitStructure.CLK_COM_DIV = 0; /* 比较器共用滤波时钟分频 */
572     CMP_InitStructure.FT = DISABLE; /* 比较器快速比较 30ns */
573     CMP_InitStructure.HYS = CMP_HYS_0mV; //CMP_HYS_20mV; /* 比较器滞回电压 */
574
575
576     //CMP0 config
577     CMP_InitStructure.CMP0_SELP = CMP0_SELP_OPA0_OUT; /* 比较器0正端信号选择 */
578     CMP_InitStructure.CMP0_SELN = CMP_SELN_DAC0; /* 比较器0负端信号选择 */
579     CMP_InitStructure.CMP0_RE = DISABLE; /* 比较器0DMA失能 */
580     CMP_InitStructure.CMP0_POL = CMP_HIGH_LEVEL; /* 比较器0高电平输出有效 */
581     CMP_InitStructure.CMP0_IRQ_TRIG = IRQ_LEVEL_TRIG_MODE; /* 比较器0电平触发中断模式 */
582     CMP_InitStructure.CMP0_IN_EN = ENABLE; /* 比较器0信号输入使能 */
583     CMP_InitStructure.CMP0_IE = DISABLE; /* 比较器0信号中断使能 */
584     CMP_InitStructure.CMP0_FIL_CLK_DIV16 = 2; /* 即滤波宽度=tclk 周期*16*CMP_FltCnt (CMP_FltCnt分频系数,0~15) */
585     CMP_InitStructure.CMP0_FIL_CLK_DIV2 = 2; /* 比较器 2/1/0 滤波时钟使能 */
586     CMP_InitStructure.CMP0_CLK_EN = ENABLE; /* 比较器时钟使能 */
587     CMP_InitStructure.CMP0_EN = ENABLE; /* 比较器0开关 操作SYS_AFE_REG5 */
588
589     //CMP1 config
590     CMP_InitStructure.CMP1_SELP = CMP0_SELP_OPA1_OUT; /* 比较器1正端信号选择 */
591     CMP_InitStructure.CMP1_SELN = CMP_SELN_DAC0; /* 比较器1负端信号选择 */
592     CMP_InitStructure.CMP0_RE = DISABLE; /* 比较器1DMA失能 */
593 }
```



## 洗地机主控方案程序介绍 – 7

凌鸥创芯

围绕产业链 · 打造创新链

- 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。

```
607 void UART_init(void)
608 {
609     GPIO_InitTypeDef GPIO_InitStructure;
610     DMA_InitTypeDef DMA_InitStructure;
611     UART_InitTypeDef UART_InitStructure;
612
613     GPIO_StructInit(&GPIO_InitStructure);
614     DMA_InitTypeDef DMA_InitStructure;
615     UART_InitTypeDef UART_InitStructure;
616     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_15;
617     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
618     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
619     GPIO_Init(GPIO15, &GPIO_InitStructure);
620     DMA_InitTypeDef DMA_InitStructure;
621     DMA_InitStructure.DMA_Channel = DMA_Channel_4;
622     DMA_InitStructure.DMA_Memory0BaseAddress = (uint32_t)0x08000000;
623     DMA_InitStructure.DMA_PeripheralBaseAddress = ((uint32_t)0x40000000);
624     DMA_InitStructure.DMA_Streams = 1;
625     DMA_InitStructure.DMA_MemToMem = 0;
626     DMA_Init(DMA1, &DMA_InitStructure);
627     UART_InitStructure.UART_BaudRate = 9600;
628     UART_InitStructure.UART_WordLength = UART_WordLength_8b;
629     UART_InitStructure.UART_StopBits = UART_StopBits_1;
630     UART_InitStructure.UART_Parity = UART_Parity_No;
631     UART_InitStructure.UART_Mode = UART_Mode_Rx;
632     UART_Init(UART0, &UART_InitStructure);
633
634     // 一轮由DMA串口发送10个字节数据
635     DMA_StructInit(&DMA_InitStructure);
636     DMA_InitStructure.DMA_Channel = DMA_Channel_4;
637     DMA_InitStructure.DMA_Memory0BaseAddress = (uint32_t)0x08000000;
638     DMA_InitStructure.DMA_PeripheralBaseAddress = ((uint32_t)0x40000000);
639     DMA_InitStructure.DMA_Streams = 1;
640     DMA_InitStructure.DMA_MemToMem = 0;
641     DMA_Init(DMA1, &DMA_InitStructure);
642
643     // DMA通道使能
644     // DMA 传输完成中断使能
645     // 源地址递增，高有效，地址按照 SBTW 对应大小递增 1/2/4
646     DMA_Cmd(DMA1_Channel4, ENABLE);
647     DMA_ITConfig(DMA1_Channel4, DMA_IT_TC, ENABLE);
648 }
649
650 void UART0_IRQHandler(void)
651 {
652     if (UART_GetITStatus(UART0, UART_IT_TXE)) // 发送完成中断
653     {
654         UART_ClearITStatus(UART0, UART_IT_TXE); // 清除发送完成标志位
655     }
656     if (UART_GetITStatus(UART0, UART_IT_RXNE)) // 接收完成中断
657     {
658         UART_ClearITStatus(UART0, UART_IT_RXNE); // 清除接收完成标志位
659         Uart_Buff = UART_ReadData(UART0); // 接收 1 Byte数据
660     }
661 }
662
663 void UART1_IRQHandler(void)
664 {
665     if (UART_GetITStatus(UART1, UART_IT_TXE)) // 发送完成中断
666     {
667         UART_ClearITStatus(UART1, UART_IT_TXE); // 清除发送完成标志位
668     }
669     if (UART_GetITStatus(UART1, UART_IT_RXNE)) // 接收完成中断
670     {
671         UART_ClearITStatus(UART1, UART_IT_RXNE); // 清除接收完成标志位
672         Uart_Buff = UART_ReadData(UART1); // 接收 1 Byte数据
673     }
674 }
675
676 void dsp_uart_tx(int8_t data)
677 {
678     DSP0_SC |= BIT11;
679     DSP0_SC |= BIT3;
680     DSP_MEM0->BR0_OP = data;
681     DSP0_FC = 0x30;
682     DSP_MEM0->DATA[0x30] = 0; // clear uart0 rx done flag
683     DSP_MEM0->DATA[0x31] = 0; // clear uart0 rx done flag
684     DSP0_SC = BIT0 | BIT8;
685 }
686
687 void dsp_uart_rx(int8_t data)
688 {
689     uint8_t rx_data[32];
690     volatile uint8_t uart_flag = 0;
691     while (uart_flag == 0)
692     {
693         rx_data[p] = data;
694         p++;
695         if (p >= 32)
696         {
697             p = 0;
698         }
699     }
700     DSP0_SC |= BIT11;
701     DSP0_SC |= BIT3;
702     DSP_MEM0->DATA[0] = (DSP_MEM0->DATA[0] & 0xffff0000) | BIT9;
703     DSP0_FC = 0x30;
704     DSP_MEM0->DATA[0x30] = 0; // clear uart0 rx done flag
705     DSP_MEM0->DATA[0x31] = 0; // clear uart0 rx done flag
706     DSP0_SC = BIT0 | BIT8;
707 }
708
709 char t[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
710
711 void dsp_fun_init(void)
712 {
713     UINIT32 i;
714     SYS_ModuleClockCmd(SYS_Module_DSP0, ENABLE);
715     SYS_ModuleClockCmd(SYS_Module_GPIO, ENABLE);
716     DSP0_SC |= BIT11;
717     DSP0_SC |= BIT3;
718     // dsp data mem flush
719     for (i = 0; i < 60; i++)
720     {
721         DSP_MEM0->DATA[i] = 0;
722     }
723     // dsp code mem flush
724     for (i = 0; i < 512; i++)
725     {
726         DSP_MEM0->CODE[i] = 0;
727     }
728     // dsp data mem init
729     for (i = 0; i < (sizeof(test_data_array) / 4); i++)
730     {
731         DSP_MEM0->DATA[i] = test_data_array[i];
732     }
733     // dsp code mem init
734     for (i = 0; i < (sizeof(test_code_array) / 2); i++)
735     {
736         DSP_MEM0->CODE[i] = test_code_array[i];
737     }
738     // GPIO0_PIO = BIT14;
739     // GPIO0_PIO = BIT14; // DSP TX
740     // GPIO0_PIO = BIT0; // DSP RX
741     NVIC_EnableIRQ(DSP0_IRQn);
742     _enable_irq();
743 }
```



## 洗地机主控方案程序介绍 – 8

凌鸥创芯

围绕产业链 · 打造创新链

- 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，判断有没有通讯成功。

```
704 void SPI_init(void)
705 {
706     GPIO_InitTypeDef GPIO_InitStructure;
707     GPIO_StructInit(&GPIO_InitStructure);
708
709     GPIO_StructInit(&GPIO_InitStructure);
710     GPIO_StructInit(&GPIO_InitStructure);
711     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT; //设置为输出模式
712     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
713     GPIO_Init(GPIO2, &GPIO_InitStructure);
714     GPIO2_PDO |= BIT3;
715
716     GPIO_StructInit(&GPIO_InitStructure);
717     GPIO_StructInit(&GPIO_InitStructure);
718     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT; //设置为输出模式
719     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
720     GPIO_Init(GPIO1, &GPIO_InitStructure);
721
722     GPIO_PinAFConfig(GPIO1, GPIO_PinSource_0, AF5_SPI); //GPIO1.0复用SPI_D0功能
723     // GPIO_PinAFConfig(GPIO2, GPIO_PinSource_3, AF5_SPI); //GPIO2.3复用SPI_CS功能
724
725     GPIO_StructInit(&GPIO_InitStructure);
726     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN; //设置为输入模式
727     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
728     GPIO_Init(GPIO2, &GPIO_InitStructure);
729     GPIO_PinAFConfig(GPIO2, GPIO_PinSource_5, AF5_SPI); //GPIO1.0复用SPI_DI功能
730
731     GPIO_StructInit(&GPIO_InitStructure);
732     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT; //设置为输入模式
733     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
734     GPIO_Init(GPIO2, &GPIO_InitStructure);
735     GPIO_PinAFConfig(GPIO2, GPIO_PinSource_4, AF5_SPI); //GPIO1.0复用SPI_CLK功能
736
737     u8 SPI_Tx_data_tbl[8] = {0xf3, 0x3f, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66}; //数据发送缓冲区
738     u8 SPI_Rx_data_tbl[8] = {0}; //接收数据缓冲区
739
740     #define FLASH_ID 0x204015 //W25Q64
741     #define W25X_JedecDeviceID 0x9F
742     #define CS_ON (GPIO_SetBits(GPIO2, GPIO_Pin_3))
743     #define CS_OFF (GPIO_ResetBits(GPIO2, GPIO_Pin_3))
744     #define Dummy_byte 0xff
745
746     uint8_t SPI_DMA_Tx[8] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88};
747     uint8_t SPI_DMA_Rx[8] = {0x00};
748     volatile u8 SPI_DMA_Run_flag = 0;
749     volatile u8 SPI_DMA_Tx_data = 0xff;
750
751     void SPI_test(void)
752     {
753         NVIC_SetPriority(DMA_IRQn, 1);
754         NVIC_EnableIRQ(DMA_IRQn);
755         SoftDelay(1000);
756         DeviceID = SPI_FLASH_ReadDeviceID();
757         SoftDelay(1000);
758         FlashID = SPI_FLASH_ReadID();
759         SoftDelay(1000);
760         SPI_FLASH_BufferRead();
761     }
762
763     void SPI_init(void)
764     {
765         //
766     }
767 }
```

```
103 void DMA_IRQHandler(void)
104 {
105     if (DMA0_IF & BIT0) //DMA通道0完成中断标志
106     {
107         DMA0_IF = BIT0;
108         DMA_CHX_EN(DMA_CHN2, DISABLE); /*使能DMA_CH2通道*/
109     }
110
111     if (DMA0_IF & BIT1) //DMA通道1完成中断标志
112     {
113         DMA0_IF = BIT1;
114         DMAIRQcnt++;
115     }
116
117     if (DMA0_IF & BIT2) //DMA通道2完成中断标志
118     {
119         DMA0_IF = BIT2;
120         DMA_CHX_EN(DMA_CHN2, DISABLE); /*使能DMA_CH2通道*/
121     }
122
123     if (DMA0_IF & BIT3) //DMA通道3完成中断标志
124     {
125         DMA0_IF = BIT3;
126         DMA_CHX_EN(DMA_CHN3, DISABLE); //软件置1 开启通道进行 DMA 搬运操作，搬运完成后 DMA 硬件将此位清零
127     }
128 }
```

```
786 volatile u8 SPI_TX_data = 0;
787 volatile u8 SPI_RX_data = 0;
788 uint8_t voice_spi_read(uint8_t data) // SPI_Bit
789 {
790     u8 rdata = 0;
791
792     SPI0_IE |= 0x03;
793     SPI_TX_data = data; //发送数据
794     SPI0_TX_DATA = data; //发送数据
795
796     while (!(SPI0_IE & BIT2))
797     {
798     }; //等待传输完成*/
799     rdata = SPI0_RX_DATA; //接收数据
800     SPI0_IE |= BIT2; //清除SPI传输完成标志位
801     SPI_RX_data = rdata;
802     return rdata;
803 }
804 #define W25X_DeviceID 0x9F
805 uint32_t SPI_FLASH_ReadDeviceID(void) // SPI_Bit - 上电执行一次
806 {
807     uint32_t Temp = 0;
808
809     /* Select the FLASH: Chip Select low */
810     CS_OFF;
811
812     /* Send "RDID" instruction */
813     voice_spi_read(W25X_DeviceID);
814     voice_spi_read(Dummy_byte);
815     voice_spi_read(Dummy_byte);
816     voice_spi_read(Dummy_byte);
817
818     /* Read a byte from the FLASH */
819     Temp = voice_spi_read(Dummy_byte);
820 }
```



## 洗地机主控方案程序介绍 – 9

凌鸥创芯

围绕产业链 · 打造创新链

### ➤ DAC输出语音信号

```
535 #define Cur_A 10.0
536 void DAC_init(void)
537 {
538     // GPIO_InitTypeDef GPIO_InitStructure;
539     DAC_InitTypeDef DAC_InitStructure;
540
541     #if 0
542     GPIO_StructInit(&GPIO_InitStructure);
543     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_ANA;
544     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
545     GPIO_Init(GPIO0, &GPIO_InitStructure);
546     GPIO_PinAFConfig(GPIO0, GPIO_PinSource_0, AF0_GPIO);
547     #endif
548
549     DAC_StructInit(&DAC_InitStructure); /* DAC结构体初始化 */
550     DAC_InitStructure.DAC_RANGE = DAC_RANGE_4V85; /* DAC输出量程为4.85V */
551     DAC_InitStructure.DACOUT_EN = ENABLE; /* 使能DAC输出 */
552     DAC_InitStructure.TIG_CH_EN = DISABLE; /* 失能TIMER触发 */
553     DAC_InitStructure.DAC_STEP = 0; /* 步进值为0 */
554     DAC_Init(DAC_Channel_0, &DAC_InitStructure); /* DAC初始化 */
555
556     DAC_Set = ((R_Vlaue*Cur_A*(40.0/(10+0.1))/2)+OPA0_Get)/4.85*4096;
557     DAC_OutputValue(DAC_Channel_0, (u16)DAC_Set); //输出1V 4095对应满量程4.85V 4095/4
558
559     DAC_Cmd(DAC_Channel_0, ENABLE); /* 使能DAC时钟 */
560 }
```



## 洗地机主控方案程序介绍 – 10

凌鸥创芯

围绕产业链 · 打造创新链

- 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。

```
Project
├── Project: LK_StdPeriph
│   └── Linko 07x Project
│       ├── Source
│       │   ├── main.c
│       │   └── AppFunction
│       │       ├── hardware_init.c
│       │       ├── interrupt.c
│       │       └── dsp_uart_fun.c
│       ├── Periph_Driver
│       └── CMSIS
└── hardware_init.c
    ├── main.c
    ├── interrupt.c
    └── lks32mc07x_opa.h

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_InitStructure;
925     /*wake IO P0.0*/
926     GPIO_StructInit(&GPIO_InitStructure); //初始化结构体
927     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
928     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
929     GPIO_Init(GPIO0, &GPIO_InitStructure);
930
931     SoftDelay(0xfffff); /*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并可以删除*/
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

凌鸥创芯

围绕产业链 · 打造创新链

### ➤ FLASH掉电存储功能

```
937 u8 Flash_Write_Buff[256] = {0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88};
938 u32 Flash_Read_Buff[64] = {0x00};
939 void Flash_Write(void)
940 {
941     #if 1
942         erase_flag = 0x9A0D361F; //擦除解锁
943         EraseSector(0x00000000, Flash_NVR); //擦除flash的main区域即存储程序区域
944         erase_flag = 0x00; //擦除上锁
945
946         prog_flag = 0x9AFDA40C; //编程解锁
947         ProgramPage(0x00000000, 256, Flash_Write_Buff, Flash_NVR); //编程flash的main区域即存储程序区域
948         erase_flag = 0x00; //编程上锁
949     #else
950
951         erase_flag = 0x9A0D361F; //擦除解锁
952         EraseSector(0x0001FC00, Flash_MAIN); //擦除flash的main区域即存储程序区域
953         erase_flag = 0x00; //擦除上锁
954
955         prog_flag = 0x9AFDA40C; //编程解锁
956         ProgramPage(0x0001FC00, 256, Flash_Write_Buff, Flash_MAIN); //编程flash的main区域即存储程序区域
957         erase_flag = 0x00; //编程上锁
958     #endif
959 }
960
961 void Flash_Read(void)
962 {
963
964     Read_More_Flash(0x0, 2, Flash_Read_Buff, Flash_NVR);
965 }
```



## 洗地机主控方案程序介绍 – 12

凌鸥创芯

围绕产业链 · 打造创新链

- 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。

```
1009 //u16 MCPWM0_IQ23_value = 0;
1010 void DebugPWM_OutputFunction(void)
1011 {
1012     if(Motor_Direction_flag == 0)
1013     {
1014         MCPWM0_TH10 = (u16)-2400;
1015         MCPWM0_TH11 = 2400;
1016         MCPWM0_TH20 = 0;
1017         MCPWM0_TH21 = 0;
1018     }
1019     else
1020     {
1021         MCPWM0_TH10 = 0;
1022         MCPWM0_TH11 = 0;
1023         MCPWM0_TH20 = (u16)-2400;
1024         MCPWM0_TH21 = 2400;
1025     }
1026
1027     PWMOutputs(ENABLE);
1028     while(1)
1029     {
1030     }
1031 }
1032 #endif
1033
```



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>

為天地立心  
為控制塑魂

**创芯驱动，领航电控未来！**

**正直诚信！ 利他共赢！ 成长超越！**