



ADC (DMA)

目录

| | |
|--|---|
| 使用 STM32CubeMX 配置 ADC 以 DMA 方式读取..... | 2 |
| 初始化代码..... | 3 |
| ADC 初始化..... | 3 |
| DMA 初始化 | 4 |
| 常用 ADC 相关操作函数 stm32f4xx_hal_adc.c..... | 5 |
| 1.1 启动 ADC(以 DMA 方式)HAL_ADC_Start_DMA..... | 5 |
| 卡尔曼滤波算法..... | 5 |



使用 STM32CubeMX 配置 ADC 以 DMA 方式读取

STM32CubeMX Project_TestIoc*: STM32F405RGTx

File Window Help

Home STM32F405RGTx Project_TestIoc - Pinout & Configuration GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

Categories A-Z

System Core

Analog

ADC1

ADC2

ADC3

DAC

Timers

Connectivity

Multimedia

Security

Computing

Middleware

ADC1 Mode and Configuration

Mode

IN0

IN1

IN2

IN3

IN4

IN5

IN6

IN7

IN8

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

Search Ctrl+F

ADCs_Common_Settings

Mode

Independent mode

ADC_Settings

Clock Prescaler

Resolution

Data Alignment

Scan Conversion Mode

Continuous Conversion Mode

Discontinuous Conversion Mode

DMA Continuous Requests

End Of Conversion Selection

ADC_Regular_ConversionMode

Number Of Conversion

External Trigger Conversion Source

External Trigger Conversion

Rank

Channel

Sampling Time

ADC_Injected_ConversionMode

Number Of Conversion

模式-独立模式

时钟分频-4

转换位数-12

数据对齐方式-右对齐

使能扫描模式

使能连续转换模式

失能不连续转换模式

使能DMA请求

单通道转换结束EOC标志位

转换个数-1

外部触发源-软件逻辑转换

外部触发电平-无

优先级1:

通道-4

采样间隔-480个时钟周期

选择PA4引脚的功能为 ADC1_IN4

STM32F405RGTx LQFP64

PC1

PC2

PC3

VSSA

VDDA

PA0...

PA1

PA2

PA3

VSS

VDD

PA4

PA5

PA6

PA7

PC4

PC5

PB0

PB1

PB2

PB10

PB11

VCAP...

ADC1_IN4

LED1

USER

INPUT_IT

STM32CubeMX Project_TestIoc*: STM32F405RGTx

File Window Help

Home STM32F405RGTx Project_TestIoc - Pinout & Configuration GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

Categories A-Z

System Core

Analog

ADC1

ADC2

ADC3

DAC

Timers

Connectivity

Multimedia

Security

Computing

Middleware

ADC1 Mode and Configuration

Mode

IN0

IN1

IN2

IN3

IN4

IN5

IN6

IN7

IN8

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

DMA Request Stream Direction Priority

ADC1 DMA2 Stream 0 Peripheral To Memory High

DMA请求 通道 方向 优先级

ADC1 DMA2通道0 外设-内存 高

Add Delete

DMA Request Settings

Mode Circular

Increment Address

Use Fifo

Threshold

Data Width

Burst Size

Peripheral Memory

Continuous mode

数据长度-字节

STM32F405RGTx LQFP64

PC1

PC2

PC3

VSSA

VDDA

PA0...

PA1

PA2

PA3

VSS

VDD

PA4

PA5

PA6

PA7

PC4

PC5

PB0

PB1

PB2

PB10

PB11

VCAP...

ADC1_IN4

LED1

USER

INPUT_IT



初始化代码

在 adc.c 源文件中，有如下配置代码

ADC 初始化

```
1. void MX_ADC1_Init(void)
2. {
3.
4.     /* USER CODE BEGIN ADC1_Init 0 */
5.
6.     /* USER CODE END ADC1_Init 0 */
7.
8.     ADC_ChannelConfTypeDef sConfig = {0};
9.
10.    /* USER CODE BEGIN ADC1_Init 1 */
11.
12.    /* USER CODE END ADC1_Init 1 */
13.    /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)
14.    */
15.    hadc1.Instance = ADC1;
16.    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
17.    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
18.    hadc1.Init.ScanConvMode = ENABLE;
19.    hadc1.Init.ContinuousConvMode = ENABLE;
20.    hadc1.Init.DiscontinuousConvMode = DISABLE;
21.    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
22.    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
23.    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
24.    hadc1.Init.NbrOfConversion = 1;
25.    hadc1.Init.DMAContinuousRequests = ENABLE;
26.    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
27.    if (HAL_ADC_Init(&hadc1) != HAL_OK)
28.    {
29.        Error_Handler();
30.    }
31.    /** Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
32.    */
33.    sConfig.Channel = ADC_CHANNEL_4;
```



```
34. sConfig.Rank = 1;
35. sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;
36. if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
37. {
38.     Error_Handler();
39. }
40. /* USER CODE BEGIN ADC1_Init 2 */
41.
42. /* USER CODE END ADC1_Init 2 */
43.
44. }
```

DMA 初始化

```
1. void HAL_ADC_MspInit(ADC_HandleTypeDef* adcHandle)
2.
3.
4. GPIO_InitTypeDef GPIO_InitStruct = {0};
5. if(adcHandle->Instance==ADC1)
6. {
7.     /* USER CODE BEGIN ADC1_MspInit 0 */
8.
9.     /* USER CODE END ADC1_MspInit 0 */
10.    /* ADC1 clock enable */
11.    __HAL_RCC_ADC1_CLK_ENABLE();
12.
13.    __HAL_RCC_GPIOA_CLK_ENABLE();
14.    /**ADC1 GPIO Configuration
15.    PA4      -----> ADC1_IN4
16.    */
17.    GPIO_InitStruct.Pin = GPIO_PIN_4;
18.    GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
19.    GPIO_InitStruct.Pull = GPIO_NOPULL;
20.    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
21.
22.    /* ADC1 DMA Init */
23.    /* ADC1 Init */
24.    hdma_adc1.Instance = DMA2_Stream0;
25.    hdma_adc1.Init.Channel = DMA_CHANNEL_0;
26.    hdma_adc1.Init.Direction = DMA_PERIPH_TO_MEMORY;
27.    hdma_adc1.Init.PeriphInc = DMA_PINC_DISABLE;
28.    hdma_adc1.Init.MemInc = DMA_MINC_ENABLE;
29.    hdma_adc1.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
30.    hdma_adc1.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
```



```
31. hdma_adc1.Init.Mode = DMA_NORMAL;
32. hdma_adc1.Init.Priority = DMA_PRIORITY_HIGH;
33. hdma_adc1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
34. if (HAL_DMA_Init(&hdma_adc1) != HAL_OK)
35. {
36.     Error_Handler();
37. }
38.
39. __HAL_LINKDMA(adHandle,DMA_Handle,hdma_adc1);
40.
41. /* USER CODE BEGIN ADC1_MspInit 1 */
42.
43. /* USER CODE END ADC1_MspInit 1 */
44. }
```

常用 ADC 相关操作函数 stm32f4xx_hal_adc.c

1.1 启动 ADC(以 DMA 方式)HAL_ADC_Start_DMA

```
1. HAL_StatusTypeDef HAL_ADC_Start_DMA(ADC_HandleTypeDef* hadc, uint32_t* pData
    , uint32_t Length)
```

入口参数: ADC 句柄、数据存放地址指针、数据长度

返回值: HAL_OK(成功)

使用示例:

```
1. uint32_t ADC_Value[1];
2. float Voltage;
3. HAL_ADC_Start_DMA(&hadc1,ADC_Value,1);
4. Voltage = ADC_Value[0]*3.3f/4096.0f;
```

实际电压计算公式:

$$\text{电压}(V) = \frac{\text{ADC 数据} \times \text{基准电压}}{\text{ADC 位数的二进制值}}$$

卡尔曼滤波算法

ADC 读取的电压值在小范围内会产生高频跳变,除了在电路中使用各种滤波电路外,在软件上使用卡尔曼滤波算法是一个很好的方案。



```
1. /**
2.  * @name    kalmanCreate
3.  * @brief   创建一个卡尔曼滤波器
4.  * @param   p: 滤波器
5.  *          T_Q:系统噪声协方差
6.  *          T_R:测量噪声协方差
7.  *
8.  * @retval  none
9.  */
10. void kalmanCreate(kalman *p, float T_Q, float T_R)
11. {
12.     //kalman* p = ( kalman*)malloc(sizeof( kalman));
13.     p->X_last = (float)0;
14.     p->P_last = 0;
15.     p->Q = T_Q;
16.     p->R = T_R;
17.     p->A = 1;
18.     p->H = 1;
19.     p->X_mid = p->X_last;
20.     //return p;
21. }
22.
23. /**
24.  * @name    KalmanFilter
25.  * @brief   卡尔曼滤波器
26.  * @param   p: 滤波器
27.  *          dat:待滤波数据
28.  * @retval  滤波后的数据
29.  */
30.
31. float KalmanFilter(kalman* p, float dat)
32. {
33.     p->X_mid = p->A*p->X_last;                //x(k|k-1) = AX(k-1|k-1)+B
34.     U(k)                                       //p(k|k-1) = Ap(k-1|k-1)A'
35.     p->P_mid = p->A*p->P_last+p->Q;            +Q
36.     p->kg = p->P_mid/(p->P_mid+p->R);           //kg(k) = p(k|k-1)H'/(Hp(k
37.     |k-1)'+R)
38.     p->X_now = p->X_mid+p->kg*(dat-p->X_mid);   //x(k|k) = X(k|k-1)+kg(k)(
39.     Z(k)-HX(k|k-1))
40.     p->P_now = (1-p->kg)*p->P_mid;             //p(k|k) = (I-kg(k)H)P(k|k
41.     -1)
42.     p->P_last = p->P_now;                     //状态更新
43.     p->X_last = p->X_now;
```



```
40.     return p->X_now;
41. }

1.  typedef struct {
2.     float X_last; //上一时刻的最优结果
3.     float X_mid;  //当前时刻的预测结果
4.     float X_now;  //当前时刻的最优结果
5.     float P_mid;  //当前时刻预测结果的协方差
6.     float P_now;  //当前时刻最优结果的协方差
7.     float P_last; //上一时刻最优结果的协方差
8.     float kg;     //kalman 增益
9.     float A;      //系统参数
10.    float Q;
11.    float R;
12.    float H;
13. }kalman;
14.
15. void kalmanCreate(kalman *p,float T_Q,float T_R);
16. float KalmanFilter(kalman* p,float dat);
```

使用示例：

```
1. kalmanCreate(&p,0.5,0.5);
2. Voltage = ADC_Value[0]*3.3f/4096.0f;
3. Voltage = KalmanFilter(&p,Voltage);
```

25-FEB-2021

厦大嘉庚 TCR 嵌入式