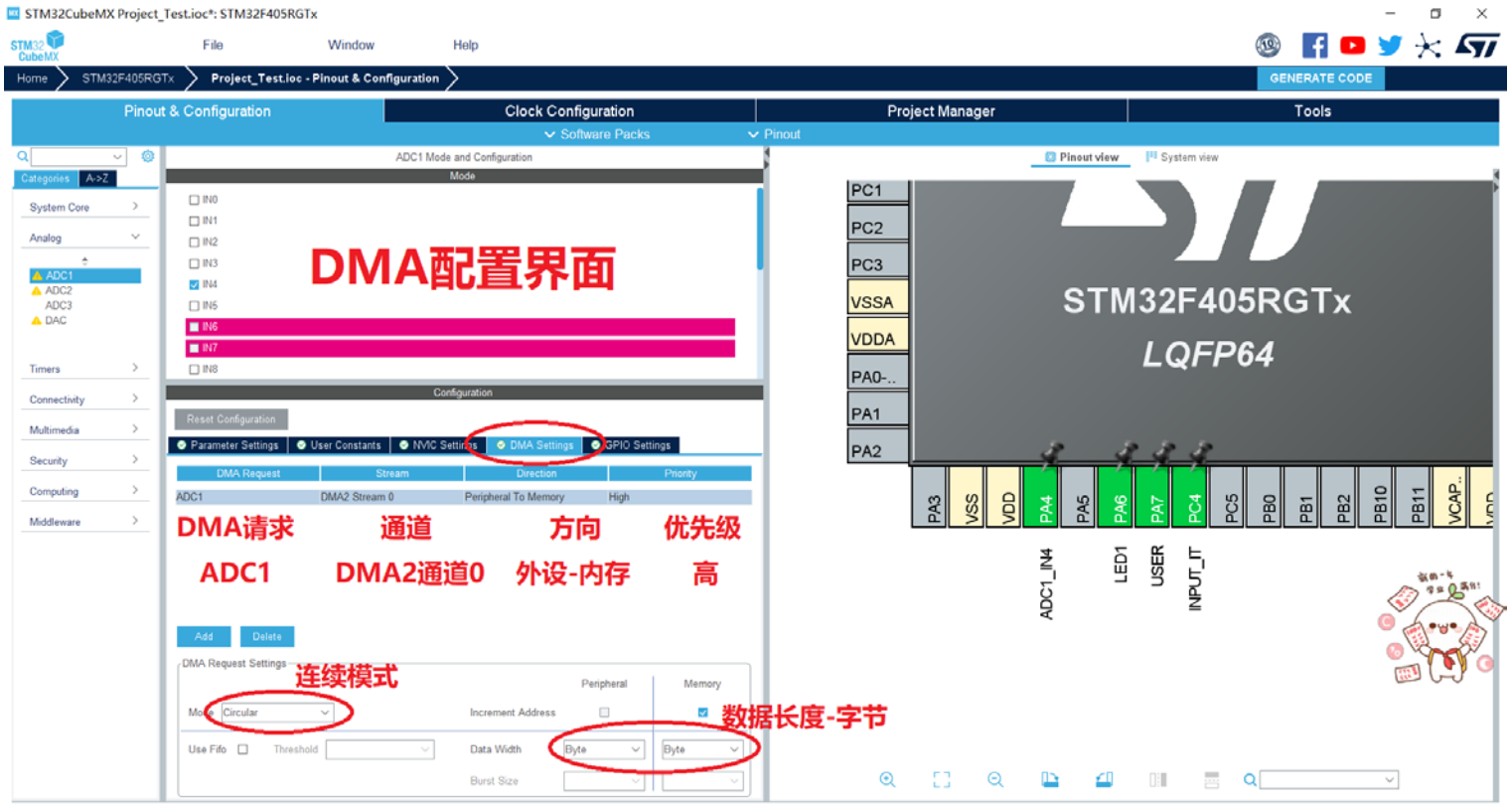
目录

使用 STM32CubeMX 配置 ADC 以 DMA 方式读取



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 Alig
    nment 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;
```

```
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}
```

DMA 初始化

```
void HAL_ADC_MspInit(ADC_HandleTypeDef* adcHandle)


GPIO_InitTypeDef GPIO_InitStruct = {0};
if(adcHandle->Instance==ADC1)
{
/* USER CODE BEGIN ADC1_MspInit 0 */

/* USER CODE END ADC1_MspInit 0 */
  /* ADC1 clock enable */
  __HAL_RCC_ADC1_CLK_ENABLE();

  __HAL_RCC_GPIOA_CLK_ENABLE();
  /**ADC1 GPIO Configuration
  PA4     ------> ADC1_IN4
  */
  GPIO_InitStruct.Pin = GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* ADC1 DMA Init */
  /* ADC1 Init */
  hdma_adc1.Instance = DMA2_Stream0;
  hdma_adc1.Init.Channel = DMA_CHANNEL_0;
  hdma_adc1.Init.Direction = DMA_PERIPH_TO_MEMORY;
  hdma_adc1.Init.PeriphInc = DMA_PINC_DISABLE;
  hdma_adc1.Init.MemInc = DMA_MINC_ENABLE;
  hdma_adc1.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
  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(adcHandle,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：

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

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。

```
/**
  * @name   kalmanCreate
  * @brief  创建一个卡尔曼滤波器
  * @param  p:  滤波器
  *         T_Q:系统噪声协方差
  *         T_R:测量噪声协方差
  *
  * @retval none
  */
void kalmanCreate(kalman *p,float T_Q,float T_R)
{
    //kalman* p = ( kalman*)malloc(sizeof( kalman));
    p->X_last = (float)0;
    p->P_last = 0;
    p->Q = T_Q;
    p->R = T_R;
    p->A = 1;
    p->H = 1;
    p->X_mid = p->X_last;
    //return p;
}

/**
  * @name   KalmanFilter
  * @brief  卡尔曼滤波器
  * @param  p:  滤波器
  *         dat:待滤波数据
  * @retval 滤波后的数据
  */

float KalmanFilter(kalman* p,float dat)
{
    p->X_mid =p->A*p->X_last;                     //x(k|k-1) = AX(k-1|k-1)+B
U(k)
    p->P_mid = p->A*p->P_last+p->Q;               //p(k|k-1) = Ap(k-1|k-1)A'
+Q
    p->kg = p->P_mid/(p->P_mid+p->R);             //kg(k) = p(k|k-1)H'/(Hp(k
|k-1)'+R)
    p->X_now = p->X_mid+p->kg*(dat-p->X_mid);     //x(k|k) = X(k|k-1)+kg(k)(
Z(k)-HX(k|k-1))
    p->P_now = (1-p->kg)*p->P_mid;                //p(k|k) = (I-kg(k)H)P(k|k
-1)
    p->P_last = p->P_now;                         //状态更新
    p->X_last = p->X_now;
```

```
    return p->X_now;
}
```

```
typedef struct {
    float X_last; //上一时刻的最优结果
    float X_mid;  //当前时刻的预测结果
    float X_now;  //当前时刻的最优结果
    float P_mid;  //当前时刻预测结果的协方差
    float P_now;  //当前时刻最优结果的协方差
    float P_last; //上一时刻最优结果的协方差
    float kg;     //kalman 增益
    float A;      //系统参数
    float Q;
    float R;
    float H;
}kalman;

void kalmanCreate(kalman *p,float T_Q,float T_R);
float KalmanFilter(kalman* p,float dat);
```

使用示例：

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

25-FEB-2021

厦大嘉庚 TCR 嵌入式