青风手把手教你学 stm32f030 系列教程

---------- 库函数操作版本

出品论坛： www.qfv8.com 青风电子社区

作者： 青风

QQ： 157736409

淘宝店：http://qfv5.taobao.com

邮箱： wanqin_002@126.com

硬件平台：QF-STM32F030 开发板

2.9 ADC 采样/DMA 通道

模数转换器（ADC）外设用于将连续的模拟电压转换成离散的数字量。Stm32f030 包含一个分辨率为 12 位的 ADC 模块，同时具有 19 个 ADC 通道，其中 16 个外部采样通道和 3 个内部信号源。ADC 采样通道源和 ADC 管脚如下表所示：

Table 29. ADC internal signals

Internal signal name	Signal type	Description
TRGx	Input	ADC conversion triggers
V_{SENSE}	Input	Internal temperature sensor output voltage
V_{REFINT}	Input	Internal voltage reference output voltage
V_{BAT}	Input	V_{BAT} pin input voltage divided by 2

Table 30. ADC pins

Name	Signal type	Remarks
V_{DDA}	Input, analog power supply	Analog power supply and positive reference voltage for the ADC, $V_{DDA} \geq V_{DD}$
V_{SSA}	Input, analog supply ground	Ground for analog power supply. Must be at V_{SS} poteential
ADC_IN[15:0]	Analog input signals	16 analog input channels

本实验我们采样 ADC_IN 选取其中一个管脚作为输入引脚，需要对 ADC 进行配置。需要配置的几个参数：ADC 转化分辨率 ，配置采样的采样方式和扫描方向。这几个参数的配置在 stm32f0xx_ad.H 中使用一个结构体进行了说：

```
01. typedef struct
02. {
```

```
03.     uint32_t ADC_Resolution;   //     配置 ADC 的转化分辨率
04.     FunctionalState ADC_ContinuousConvMode; // 配置选择连续采样或单次采样
05.     uint32_t ADC_ExternalTrigConvEdge;     // ADC 内部边缘触发
06.     uint32_t ADC_ExternalTrigConv;         //ADC  内部触发
07.     uint32_t ADC_DataAlign;                //     设置 ADC 是左对齐或者右对齐
08.     uint32_t    ADC_ScanDirection;         //    设置 ADC 扫描方向
09.  }ADC_InitTypeDef;
```

使用中给出一组操作，基于直接存储器访问 DMA 的控制方式：称为为直接存储器访问模式：DMA。这种方式是为了更为有效地利用处理器和总线可用带宽而设计的。他不需要 CPU 的情况完成操作。下大大提高了运行效率。

DMA 控制器依赖于处理器内核，但 DMA 不影响总线传输，因为 DMA 控制器总是在系统总线空闲的时候使用总线。该总线实现处理器和 DMA 控制器之间最优化设计，使两者之间的冲突降到最低，因此传输性能得到提高。

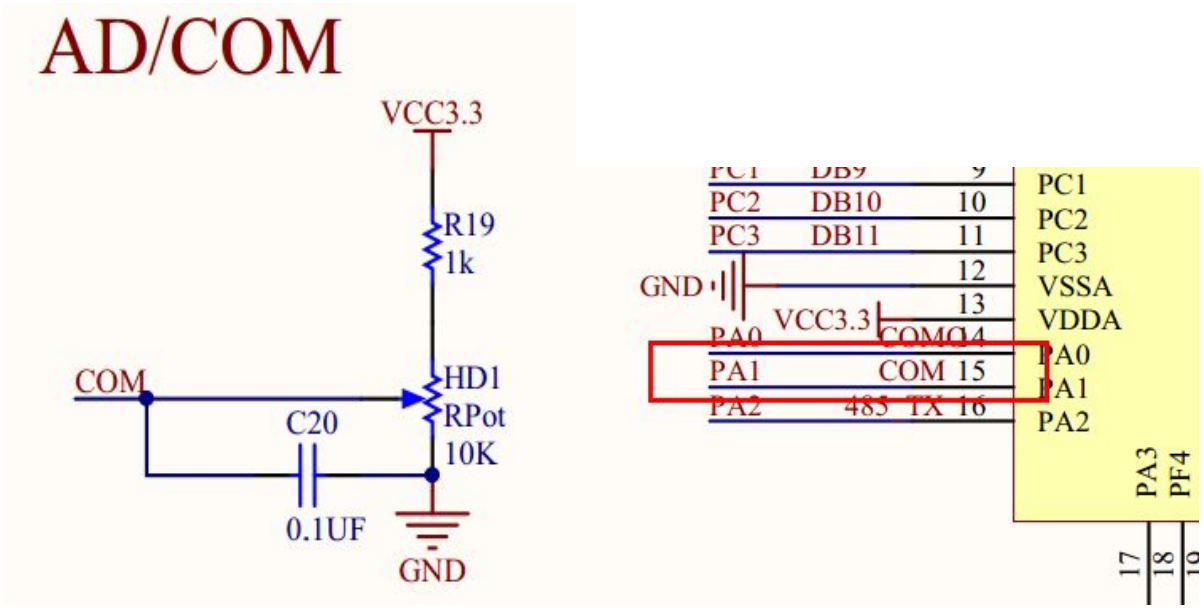
提供了存储单元到存储单元，外设到存储单元，存储单元到外设等转换模式。DMA 为每种支持的外设功能提供专用通道，可以各自独立进行配置。其配置模式多种多样，时候于各自不同的设置要求。

采样库函数配置了 DMA 的几个参数：

```
10.  typedef struct
11.  {
12.     uint32_t DMA_PeripheralBaseAddr; //配置外设地址
13.     uint32_t DMA_MemoryBaseAddr;    // 配置内存映射地址
14.     uint32_t DMA_DIR;                  // 制定外设的源或者目的地
15.     uint32_t DMA_BufferSize;            // DMA 缓冲设置
16.     uint32_t DMA_PeripheralInc;       //设置外设地址是否增加或不增加
17.     uint32_t DMA_MemoryInc;            // 设置内存地址是否增加或不增加
18.     uint32_t DMA_PeripheralDataSize; //外设数据宽度设置
19.     uint32_t DMA_MemoryDataSize;     //内存数据宽度设置
20.     uint32_t DMA_Mode;              //     DMA 模式
21.     uint32_t DMA_Priority;           //     DMA 优先级
22.     uint32_t DMA_M2M;               //    memory-to-memory  传输模式设置
23.  }DMA_InitTypeDef;
```

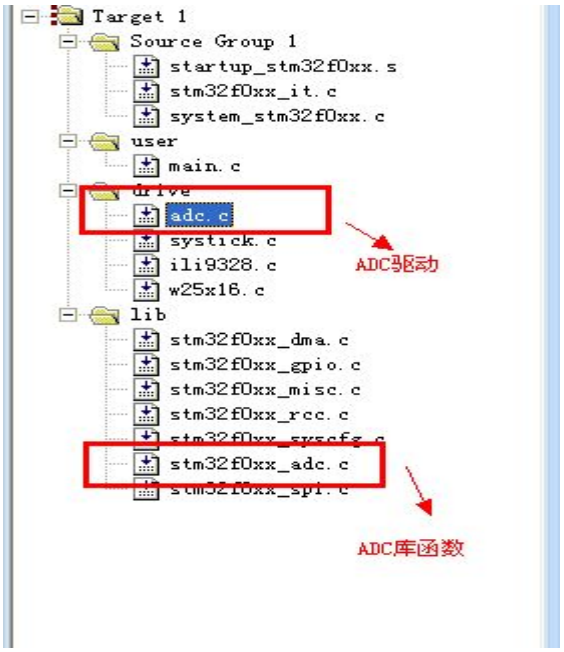
硬件准备：

硬件配置入下图所示，采用 **PA1** 作为引脚，对变阻器输入的信号进行转换：



软件准备：

软件采用库函数进行配置，用户需要配置编写 adc.c 驱动函数，工程目录如下图所示：



首先我们需要初始化 AD 配置，在 adc.c 文件中，我们编写 ADC1_DMA_Init()函数，通过配置 DMA 通道和 ADC 外设，首先 DMA 配置如下：

```
24.  /* DMA1 Channel1 Config */
25.      DMA_DeInit(DMA1_Channel1);//选择频道
26.      DMA_InitStruct.DMA_PeripheralBaseAddr = (uint32_t)ADC1_DR_Address;//设置外设地址
```

```
27.     DMA_InitStruct.DMA_MemoryBaseAddr = (uint32_t)&RegularConvData_Tab;//设置内存映射
    地址
28.     DMA_InitStruct.DMA_DIR = DMA_DIR_PeripheralSRC;
29.     DMA_InitStruct.DMA_BufferSize =4;//缓冲为 4
30.     DMA_InitStruct.DMA_PeripheralInc = DMA_PeripheralInc_Disable;//关外设地址计数
31.     DMA_InitStruct.DMA_MemoryInc = DMA_MemoryInc_Enable;//关内存地址计数
32.     DMA_InitStruct.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
33.     DMA_InitStruct.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;//半字节
34.     DMA_InitStruct.DMA_Mode = DMA_Mode_Circular;//循环模式
35.     DMA_InitStruct.DMA_Priority = DMA_Priority_High;//高优先级
36.     DMA_InitStruct.DMA_M2M = DMA_M2M_Disable;//关内存到内存
37.     DMA_Init(DMA1_Channel1, &DMA_InitStruct);
38.
39.     /* DMA1 Channel1 enable */
40.     DMA_Cmd(DMA1_Channel1, ENABLE);//频道使能
41.  ADC_DMARequestModeConfig(ADC1, ADC_DMAMode_Circular);//配置 DMA 循环模式
42.     /* Enable ADC_DMA */
43.     ADC_DMACmd(ADC1, ENABLE);
```

然后对 ADC 参数进行配置：

```
44.     /*  初始 ADC  配置  */
45.     ADC_StructInit(&ADC_InitStruct);
46.     /*  配置 ADC1  在连续模式下分辨率为 12 bits    */
47.     ADC_InitStruct.ADC_Resolution = ADC_Resolution_12b;
48.     ADC_InitStruct.ADC_ContinuousConvMode = ENABLE;
49.     ADC_InitStruct.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;
50.     ADC_InitStruct.ADC_DataAlign = ADC_DataAlign_Right;
51.     ADC_InitStruct.ADC_ScanDirection = ADC_ScanDirection_Backward;
52.     ADC_Init(ADC1, &ADC_InitStruct);
53.    /* Convert the ADC1 Vref    with 55.5 Cycles as sampling time */
54.     ADC_ChannelConfig(ADC1, ADC_Channel_1    , ADC_SampleTime_55_5Cycles);
55.     /* ADC  刻度  */
56.     ADC_GetCalibrationFactor(ADC1);
57.       ADC_DMACmd(ADC1, ENABLE);
58.     /*  使能 ADC1 */
59.     ADC_Cmd(ADC1, ENABLE);
60.     /*  等待 ADCEN  标志  */
61.     while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_ADEN));
62.
63.     /* ADC1  定期变换  */
64.     ADC_StartOfConversion(ADC1);
```

写好初始化函数后，在子函数内就可以直接调用：

```
65.    #include "stm32f0xx.h"
66.   #include "adc.h"
```

```
#include "systick.h"
#include "w25x16.h"
#include   "ili9328.h"
// ADC1 转换的电压值通过 MDA 方式传到 flash
extern __IO uint16_t   RegularConvData_Tab;

// 局部变量，用于存从 flash 读到的电压值
__IO uint16_t ADC_ConvertedValueLocal;

void delay(__IO uint32_t nCount)
{
   for(; nCount != 0; nCount--);
}

int main(void)
{
        SystemInit();
        ADC1_DMA_Init();
        LCD_init();         // 液晶显示器初始化
      SPI_FLASH_Init();
          LCD_Clear(WHITE);     // 全屏显示白色
          POINT_COLOR = BLACK; // 定义笔的颜色为黑色
          BACK_COLOR = WHITE;    // 定义笔的背景色为白色
      LCD_DrawRectage(0, 0, 320, 20, DARKBLUE);   // 画一个深蓝色边框的矩形
        LCD_ShowString(2,2,"实验十");
        LCD_ShowString(100,2,"adc 采样实验");
        while(1)
          {
              ADC_ConvertedValueLocal= RegularConvData_Tab;
              delay(0xffffee);
              LCD_ShowString(20,40,"adc 采样值:");
              LCD_ShowNum(100,40,ADC_ConvertedValueLocal,4);
              }
}
```

实验现象：

调节滑动变阻器，AD 转换后成不同的值,结果如下图所示：