介绍

本应用手册适用于需要 ACM32G103 系列芯片模拟比较器（COMP）模块的使用者。它描述了与模拟比较器模块相关的设置和功能使用方法，以便在应用程序中进行优化设计。

本应用说明应与相关的用户手册、数据表一同阅读。

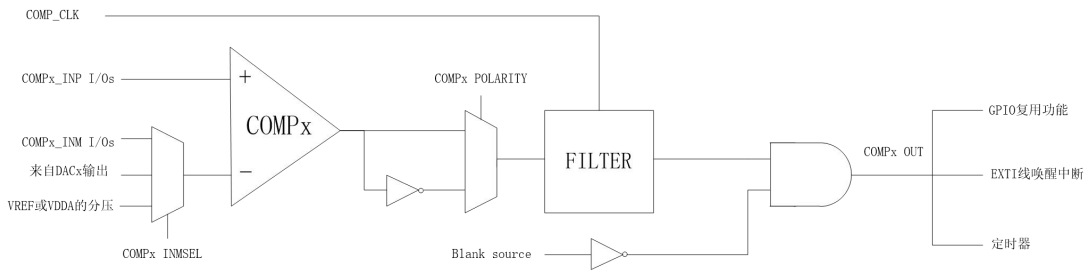
1. 概述

模拟电压比较器用于比较两个输入模拟电压的大小，并根据比较结果输出高/低电平。当比较器正端输入电压高于负端输入电压时，电压比较器输出高电平；当比较器正端输入电压低于负端输入电压时，电压比较器输出低电平。

具有以下特性：

- 支持电压比较功能；
- 比较器负端输入可配置：
  - 芯片管脚；
  - DAC 的输出；
  - 内部基准电压 VREF 或 VDDA 的分压；
- 可编程的迟滞窗口；
- 比较器输出到管脚；
- 比较器输出可作为定时器的刹车输入或捕获输入；
- 比较器输出可通过定时器切断；
- 比较器输出可作为 EXTI 控制器输入，支持 Sleep 和 Stop 模式下的唤醒功能；
- 两个比较器级联可以组合构成窗口比较器；
- 提供软件可配置的滤波时间以增强芯片的抗干扰能力。

图 1-1 比较器窗口模式级联



2. 功能描述

2.1.1. 负端输入

负端输入可以选择来自芯片管脚、DAC 输出、或者来自内部基准电压 VREF 或 VDDA 的分压。通过配置 COMP_CR 的 INMSEL 位域选择。

选择内部基准电压分压时，可以通过配置 CRV_SEL 位域选择基准分压来源为 VDDA 或 VREF，分压系数通过配置 CRV_CFG 来实现内部输入不同负端电压。

2.1.2. 滤波

滤波功能可以滤除来自比较器输入端的尖峰毛刺，防止应用电路的误触发。比较器滤波电路可以滤除因毛刺输入产生的窄脉冲输出，可以通过配置控制寄存器的 FLTEN 位使能滤波功能，配置 FLTTIME 位域可以更改滤除毛刺的最大宽度。

比较器滤波时钟可通过设置 RCC 模块中的 CCR2 寄存器，通过设置其中的 FLTCLKSEL 位来选择，设置 0 时滤波时钟为 PCLK 的 32 分频，设置 1 时为 RC32K。

注：在比较器用于唤醒 STOP 模式 MCU，且需要使用比较器滤波功能时，滤波时钟只能选择 RC32K。

2.1.3. 迟滞比较

迟滞比较功能可以防止在比较电压附近时，输出产生振荡。通过配置控制寄存器的 HYS 位域开启或更改迟滞窗口的电压范围值。

2.1.4. 切断

切断功能可以通过其他输入来切断比较器输出。通过配置控制寄存器的 BLANKSEL 位域开启或更改切断源，配置 BLANKTIME 位域更改切断窗口时间宽度。

2.1.5. STOP 模式唤醒

比较器输出支持将 MCU 从 STOP 模式唤醒。比较器输出在内部连接到 EXTI 模块的触发输入，详情见 EXTI 模块的“触发源”章节。

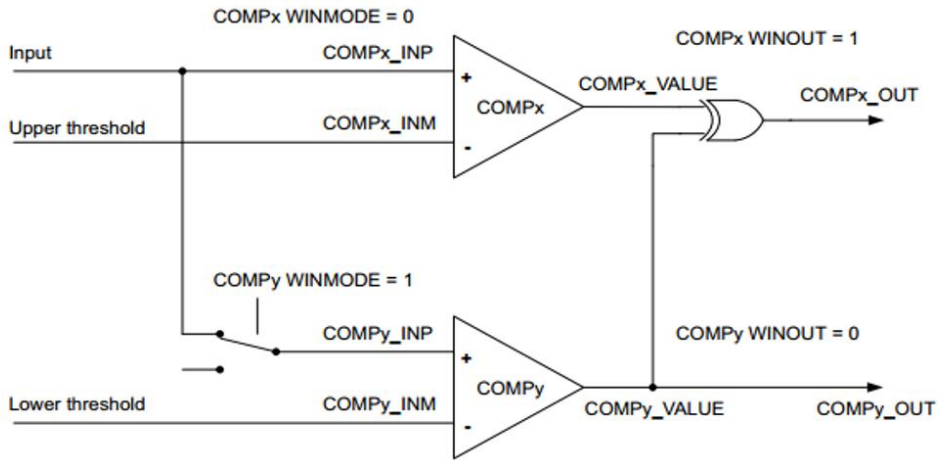
2.1.6. 窗口模式级联

可以将两个比较器连接成窗口模式，设置两个负端为不同输入电压，其中一个为上门限电压，一个为下门限电压，这样输入在门限电压之间时产生比较输出。

设置为窗口模式级联时，如下图所示：

- 需将 COMPx 的输出设置 WINOUT 设为 1，此时输出为 COMPx_VALUE 异或 COMPy_VALUE 的结果；
- 需将 COMPy 的正端输入设置 WINMODE 设为 1，此时 COMPy 的正端输入连接到 COMPx 的正端输入。

图 2-1 比较器窗口模式级联



–

3. 应用例程说明

3.1. 比较器基本功能演示例程

正负端输入电压比较，结果输出到外部管脚。正端输入高于负端则输出结果为高；正端输入低于负端则输出结果为低。

3.1.1. 硬件层配置

在 HAL_COMP_MspInit 函数中，将需要用到的管脚设置为模拟端口，例程中包含了基本功能模式的设置方法，这里代码需要用户根据具体应用情况自行设置修改。

例程中为 COMP1 的管脚设置。

例：

```
if(hcomp->Init.COMPx == COMP1)
    {
        //VINP:PA1, VINM:PA4, VOUT:PA0(AF6)

        __HAL_RCC_GPIOA_CLK_ENABLE();

        GPIO_Handle.Pin        = GPIO_PIN_1;
        GPIO_Handle.Mode       = GPIO_MODE_ANALOG;
        GPIO_Handle.Pull       = GPIO_NOPULL;
        GPIO_Handle.Drive      = GPIO_DRIVE_LEVEL3;
        HAL_GPIO_Init(GPIOA, &GPIO_Handle);

        GPIO_Handle.Pin        = GPIO_PIN_4;
        GPIO_Handle.Mode       = GPIO_MODE_ANALOG;
        GPIO_Handle.Pull       = GPIO_NOPULL;
        GPIO_Handle.Drive      = GPIO_DRIVE_LEVEL3;
        HAL_GPIO_Init(GPIOA, &GPIO_Handle);

        GPIO_Handle.Pin                 = GPIO_PIN_0;
        GPIO_Handle.Mode                = GPIO_MODE_AF_PP;
        GPIO_Handle.Pull                = GPIO_NOPULL;
        GPIO_Handle.Drive      = GPIO_DRIVE_LEVEL3;
        GPIO_Handle.Alternate          = GPIO_FUNCTION_6;
        HAL_GPIO_Init(GPIOA, &GPIO_Handle);
    }
```

3.1.2. 初始化设置

- 首先定义 COMP 的结构体变量

  例如：

  ```
  COMP_HandleTypeDef COMP1_handle;
  ```

- 然后给所需的结构体成员变量赋值

```
void COMP1_Init(void)
{
    HAL_RCC_FLTClockSourceConfig(RCC_FLTCLKSOURCE_PCLK_DIV32);

    COMP1_handle.Instance = COMP;//赋值 COMP 基地址
    COMP1_handle.Init.COMPx = COMP1;//指定需要配置的比较号
    COMP1_handle.Init.InPSel = COMP1_INPSEL_PA1;//指定正端输入
    COMP1_handle.Init.InMSel = COMP1_INMSEL_PA4;//指定负端输入
    COMP1_handle.Init.Polarity = COMP_POLARITY_NOINVERT;//输出极性为同相
    COMP1_handle.Init.HYS = COMP_HYS_7;//迟滞窗口选择 48mv
    COMP1_handle.Init.BlankSel = COMP_BLANKSEL_NONE;//不选择切断源
    COMP1_handle.Init.WinMode = COMP_WINMODE_SELF_INP;//正端输入选择本通道
    COMP1_handle.Init.WinOut = COMP_WINOUT_DIRECT;//输出 VCOUT1
    COMP1_handle.Init.FltEn = COMP_FLT_ENABLE;//滤波使能
    COMP1_handle.Init.FltTime = COMP_FLTTIME_4CLK;//滤波周期为 4 个周期

    HAL_COMP_Init(&COMP1_handle);//执行初始化
}
```

3.1.3. 运行操作

```
void APP_Test(void)
{
    COMP1_Init();
    HAL_COMP_Start(&COMP1_handle);
}
```

3.2. 比较器窗口功能演示例程

COMP1 正端输入电平高于或 COMP1 及 COMP2 的负端输入时，输出到管脚的比较结果为低；当 COMP1 正端输入电平介于或 COMP1 及 COMP2 的负端输入时，输出到管脚的比较结果为高。

3.2.1. 硬件层配置

COMP1 同 3.1.1.章节。

```
else if(hcomp->Init.COMPx == COMP2)
{
    //VINP:PA7, VINM:PA5, VOUT:PA12(AF6)

    __HAL_RCC_GPIOA_CLK_ENABLE();

    GPIO_Handle.Pin        = GPIO_PIN_7;
    GPIO_Handle.Mode       = GPIO_MODE_ANALOG;
    GPIO_Handle.Pull       = GPIO_NOPULL;
    GPIO_Handle.Drive      = GPIO_DRIVE_LEVEL3;
    HAL_GPIO_Init(GPIOA, &GPIO_Handle);

    GPIO_Handle.Pin        = GPIO_PIN_5;
    GPIO_Handle.Mode       = GPIO_MODE_ANALOG;
    GPIO_Handle.Pull       = GPIO_NOPULL;
    GPIO_Handle.Drive      = GPIO_DRIVE_LEVEL3;
    HAL_GPIO_Init(GPIOA, &GPIO_Handle);

    GPIO_Handle.Pin                = GPIO_PIN_12;
    GPIO_Handle.Mode               = GPIO_MODE_AF_PP;
    GPIO_Handle.Pull               = GPIO_NOPULL;
    GPIO_Handle.Drive      = GPIO_DRIVE_LEVEL3;
    GPIO_Handle.Alternate          = GPIO_FUNCTION_6;
    HAL_GPIO_Init(GPIOA, &GPIO_Handle);
}
```

3.2.2. 初始化设置

- 首先定义 COMP 的结构体变量

  例如：

  ```
  COMP_HandleTypeDef COMP1_handle;
  COMP_HandleTypeDef COMP2_handle;
  ```

- 然后给所需的结构体成员变量赋值

  ```
  void COMP1_Init(void)
  {
      HAL_RCC_FLTClockSourceConfig(RCC_FLTCLKSOURCE_PCLK_DIV32);

      COMP1_handle.Instance = COMP;//赋值 COMP 基地址
  ```

```
    COMP1_handle.Init.COMPx = COMP1;//指定需要配置的比较号
    COMP1_handle.Init.InPSel = COMP1_INPSEL_PA1;//指定正端输入
    COMP1_handle.Init.InMSel = COMP1_INMSEL_PA4;//指定负端输入
    COMP1_handle.Init.Polarity = COMP_POLARITY_NOINVERT;//输出极性为同相
    COMP1_handle.Init.HYS = COMP_HYS_7;//迟滞窗口选择 48mv
    COMP1_handle.Init.BlankSel = COMP_BLANKSEL_NONE;//不选择切断源
    COMP1_handle.Init.WinMode = COMP_WINMODE_SELF_INP;//正端输入选择本通道
    COMP1_handle.Init.WinOut =COMP_WINOUT_XOR;//输出 VCOUT1 XOR VCOUT2
    COMP1_handle.Init.FltEn = COMP_FLT_ENABLE;//滤波使能
    COMP1_handle.Init.FltTime = COMP_FLTTIME_4CLK;//滤波周期为 4 个周期

    HAL_COMP_Init(&COMP1_handle);//执行初始化
}
void COMP2_Init(void)
{
    HAL_RCC_FLTClockSourceConfig(RCC_FLTCLKSOURCE_PCLK_DIV32);

    COMP2_handle.Instance = COMP;//赋值 COMP 基地址
    COMP2_handle.Init.COMPx = COMP2;//指定需要配置的比较号
    COMP2_handle.Init.InPSel = COMP2_INPSEL_PA7;//指定正端输入
    COMP2_handle.Init.InMSel = COMP2_INMSEL_PA5;//指定负端输入
    COMP2_handle.Init.Polarity = COMP_POLARITY_NOINVERT;//输出极性为同相
    COMP2_handle.Init.HYS = COMP_HYS_7;//迟滞窗口选择 48mv
    COMP2_handle.Init.BlankSel = COMP_BLANKSEL_NONE;//不选择切断源
    COMP2_handle.Init.WinMode = COMP_WINMODE_ANOTHER_INP;//正端输入选择 COMP1 的 INPSEL
    COMP2_handle.Init.WinOut = COMP_WINOUT_DIRECT;//输出 VCOUT2
    COMP2_handle.Init.FltEn = COMP_FLT_ENABLE;//滤波使能
    COMP2_handle.Init.FltTime = COMP_FLTTIME_4CLK;//滤波周期为 4 个周期

    HAL_COMP_Init(&COMP2_handle);//执行初始化
}
```

3.2.3. 运行操作

```
void APP_Test(void)
{
    COMP1_Init();
    COMP2_Init();
    HAL_COMP_Start(&COMP1_handle);
    HAL_COMP_Start(&COMP2_handle);
}
```

3.3. DAC 电压输出到比较器负端功能演示例程

DAC_OUT 通过内部通道连接到 COMP 负端输入。正负端输入电平进行比较。

3.3.1. 硬件层配置

同 3.1.1.章节。

3.3.2. 初始化设置

- 首先定义 COMP、DAC 的结构体变量

  例如：

```
COMP_HandleTypeDef COMP1_handle;
DAC_HandleTypeDef hdac1;
```

- 然后给所需的结构体成员变量赋值

```
void COMP1_Init(void)
{
    HAL_RCC_FLTClockSourceConfig(RCC_FLTCLKSOURCE_PCLK_DIV32);

    COMP1_handle.Instance = COMP;//赋值 COMP 基地址
    COMP1_handle.Init.COMPx = COMP1;//指定需要配置的比较号
    COMP1_handle.Init.InPSel = COMP1_INPSEL_PA1;//指定正端输入
    COMP1_handle.Init.InMSel =COMP1_INMSEL_DAC1;//指定负端输入
    COMP1_handle.Init.Polarity = COMP_POLARITY_NOINVERT;//输出极性为同相
    COMP1_handle.Init.HYS = COMP_HYS_7;//迟滞窗口选择 48mv
    COMP1_handle.Init.BlankSel = COMP_BLANKSEL_NONE;//不选择切断源
    COMP1_handle.Init.WinMode = COMP_WINMODE_SELF_INP;//正端输入选择本通道
    COMP1_handle.Init.WinOut = COMP_WINOUT_DIRECT;//输出 VCOUT1
    COMP1_handle.Init.FltEn = COMP_FLT_ENABLE;//滤波使能
    COMP1_handle.Init.FltTime = COMP_FLTTIME_4CLK;//滤波周期为 4 个周期

    HAL_COMP_Init(&COMP1_handle);//执行初始化
}
/*DAC 初始化*/
void DAC_Config_OutPut_Voltage()
{
    hdac1.Instance = DAC;
    HAL_DAC_Init(&hdac1);

    DAC_ChannelConfTypeDef sConfig;
    sConfig.DAC_Trigger = DAC_TRIGGER_SOFTWARE;
```

```
    sConfig.DAC_Trigger2 = DAC_TRIGGER_SOFTWARE;
    sConfig.DAC_SampleAndHold = DAC_SAMPLEANDHOLD_DISABLE;
    sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_DISABLE;
    sConfig.DAC_ConnectOnChipPeripheral = DAC_CHIPCONNECT_INTERNAL;//DAC 输出到内部通道
    sConfig.DAC_UserTrimming = DAC_TRIMMING_FACTORY;

    HAL_DAC_ConfigChannel(&hdac1, &sConfig,DAC_CHANNEL_2);

    /* 自动校准，可根据需要开启/关闭 */
    HAL_DACEx_SelfCalibrate(&hdac1, &sConfig,DAC_CHANNEL_2);
    HAL_SimpleDelay(500000);
}
/*设置 DAC 输出电压*/
void DAC_OutPutVoltage(float voltage)
{
    if( voltage > 3.3)
    {
        return;
    }
    uint16_t data=((voltage / 3.3) * 4095);
    HAL_DACEx_DualSetValue(&hdac1,DAC_ALIGN_12B_R,data,data);
    HAL_DAC_Start(&hdac1, DAC_CHANNEL_2);
}
```

3.3.3. 运行操作

```
void APP_Test(void)
{
    DAC_Config_OutPut_Voltage();
    DAC_OutPutVoltage(2.5);
    COMP1_Init();
    HAL_COMP_Start(&COMP1_handle);
}
```

3.4. 比较器切断源功能演示例程

定时器比较输出信号切断比较器输出信号。

3.4.1. 硬件层配置

同 3.1.1.章节。

3.4.2. 初始化设置

- 首先定义 COMP、TIM 的结构体变量

  例如：

```
COMP_HandleTypeDef COMP1_handle;
TIM_HandleTypeDef TIM1_Handler;
```

- 然后给所需的结构体成员变量赋值

```
void COMP1_Init(void)
{
    HAL_RCC_FLTClockSourceConfig(RCC_FLTCLKSOURCE_PCLK_DIV32);

    COMP1_handle.Instance = COMP;//赋值 COMP 基地址
    COMP1_handle.Init.COMPx = COMP1;//指定需要配置的比较号
    COMP1_handle.Init.InPSel = COMP1_INPSEL_PA1;//指定正端输入
    COMP1_handle.Init.InMSel = COMP1_INMSEL_PA4;//指定负端输入
    COMP1_handle.Init.Polarity = COMP_POLARITY_NOINVERT;//输出极性为同相
    COMP1_handle.Init.HYS = COMP_HYS_7;//迟滞窗口选择 48mv
    COMP1_handle.Init.BlankSel = COMP_BLANKSEL_NONE;//不选择切断源
    COMP1_handle.Init.WinMode = COMP_WINMODE_SELF_INP;//正端输入选择本通道
    COMP1_handle.Init.WinOut = COMP_WINOUT_DIRECT;//输出 VCOUT1
    COMP1_handle.Init.FltEn = COMP_FLT_ENABLE;//滤波使能
    COMP1_handle.Init.FltTime = COMP_FLTTIME_4CLK;//滤波周期为 4 个周期

    HAL_COMP_Init(&COMP1_handle);//执行初始化
}
#define PWM_OUTPUT_CHANNEL      TIM_CHANNEL_5
#define TIM_CLOCK_FREQ                (8000000U)
void TIM1_Init(void)
{
   TIM_OC_InitTypeDef Tim_OC_Init_Para;
    uint32_t timer_clock;

    timer_clock = HAL_RCC_GetPCLK2Freq();   // TIM1 is on APB2

    if (HAL_RCC_GetHCLKFreq() != timer_clock)   // if hclk/pclk != 1, then timer clk = pclk * 2
    {
        timer_clock =   timer_clock << 1;
```

```
    }

    TIM1_Handler.Instance = TIM1;
    TIM1_Handler.Init.ARRPreLoadEn = TIM_ARR_PRELOAD_ENABLE;
    TIM1_Handler.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    TIM1_Handler.Init.CounterMode = TIM_COUNTERMODE_UP;
    TIM1_Handler.Init.RepetitionCounter = 0;
    TIM1_Handler.Init.Prescaler = (timer_clock/TIM_CLOCK_FREQ) - 1;
    if (timer_clock%TIM_CLOCK_FREQ > TIM_CLOCK_FREQ/2)
    {
        TIM1_Handler.Init.Prescaler = TIM1_Handler.Init.Prescaler + 1;
    }
    TIM1_Handler.Init.Period = (TIM_CLOCK_FREQ/1000)*1 - 1;    // period = 1ms
     TIM1_MSP_Pre_Init(&TIM1_Handler);

    HAL_TIMER_Base_Init(&TIM1_Handler);

    Tim_OC_Init_Para.OCMode = OUTPUT_MODE_PWM1;
    Tim_OC_Init_Para.OCIdleState = OUTPUT_IDLE_STATE_0;
    Tim_OC_Init_Para.OCNIdleState = OUTPUT_IDLE_STATE_0;
    Tim_OC_Init_Para.OCPolarity = OUTPUT_POL_ACTIVE_HIGH;
    Tim_OC_Init_Para.OCNPolarity = OUTPUT_POL_ACTIVE_HIGH;
    Tim_OC_Init_Para.OCFastMode =    OUTPUT_FAST_MODE_DISABLE;
    Tim_OC_Init_Para.Pulse = (TIM1_Handler.Init.Period + 1)/2;      // 50% duty cycle
    HAL_TIMER_Output_Config(TIM1_Handler.Instance, &Tim_OC_Init_Para, PWM_OUTPUT_CHANNEL);

}
```

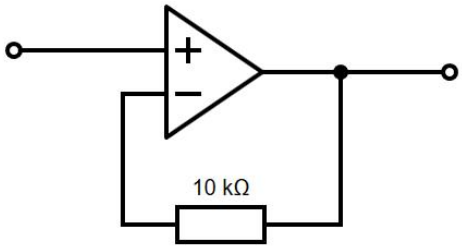
3.4.3. 运行操作

```
void APP_Test(void)
{
    TIM1_Init();
    HAL_TIM_PWM_Output_Start(TIM1_Handler.Instance, PWM_OUTPUT_CHANNEL);
    COMP1_Init();
    HAL_COMP_Start(&COMP1_handle);
}
```

4. 外置模式（SA）运放应用电路举例

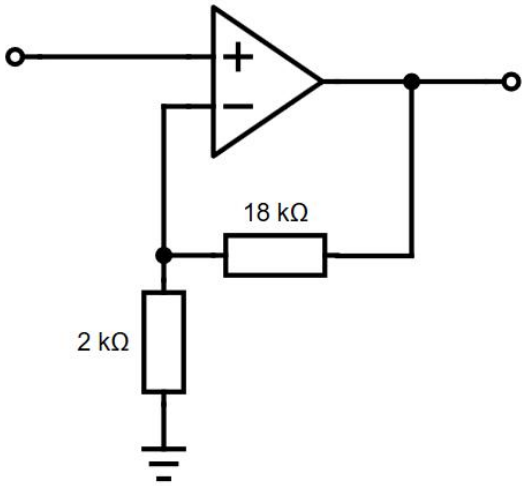
4.1. 单位增益缓冲电路

单位增益缓冲电路用于阻抗匹配，高输入阻抗，低输出阻抗，增益为 1 。典型应用电路如下图所示：



4.2. 直流信号放大电路

下图是一个典型同相 10 倍直流放大电路的应用电路图。根据负反馈电阻和负反馈落地电阻计算电路的直流增益为：A = 1 + Rf / R; 其中 Rf = 18KΩ，R = 2KΩ，A = 10;

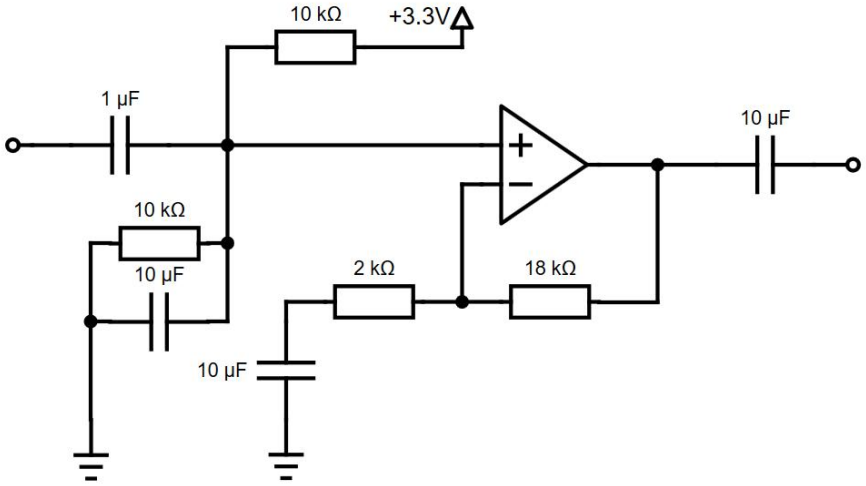


4.3. 交流信号放大电路

下图是一个典型同相 10 倍交流放大电路的应用电路图。电路的直流增益为 1，交流增益为：A = 1 + Rf / R; 其中 Rf = 18KΩ，R = 2KΩ，A = 10。

需要注意的是，由于运放采用 MCU 的 VDDA 单电源供电，交流信号放大电路与直流放大电路有很大不同。主要有如下几点：

- 需要输入、输出隔直电容；
- 需要在正端输入提供 VDDA / 2 的中点偏置电压；
- 需要反馈落地隔直电容；
- 需要注意电容和阻抗形成的一阶 RC 滤波效应影响电路的带宽，一般-3dB 带宽截止频率点为 Fc = 1 / (2*π*R*C)。



联系我们

公司：上海航芯电子科技股份有限公司
地址：上海市闵行区合川路 2570 号科技绿洲三期 2 号楼 702 室
邮编：200241
电话：+86-21-6125 9080
传真：+86-21-6125 9080-830
Email： Service@AisinoChip.com
Website： www.aisinochip.com

版本维护

版本	日期	作者	描述
V1.0	2023-05-09	Aisinochip	初始版