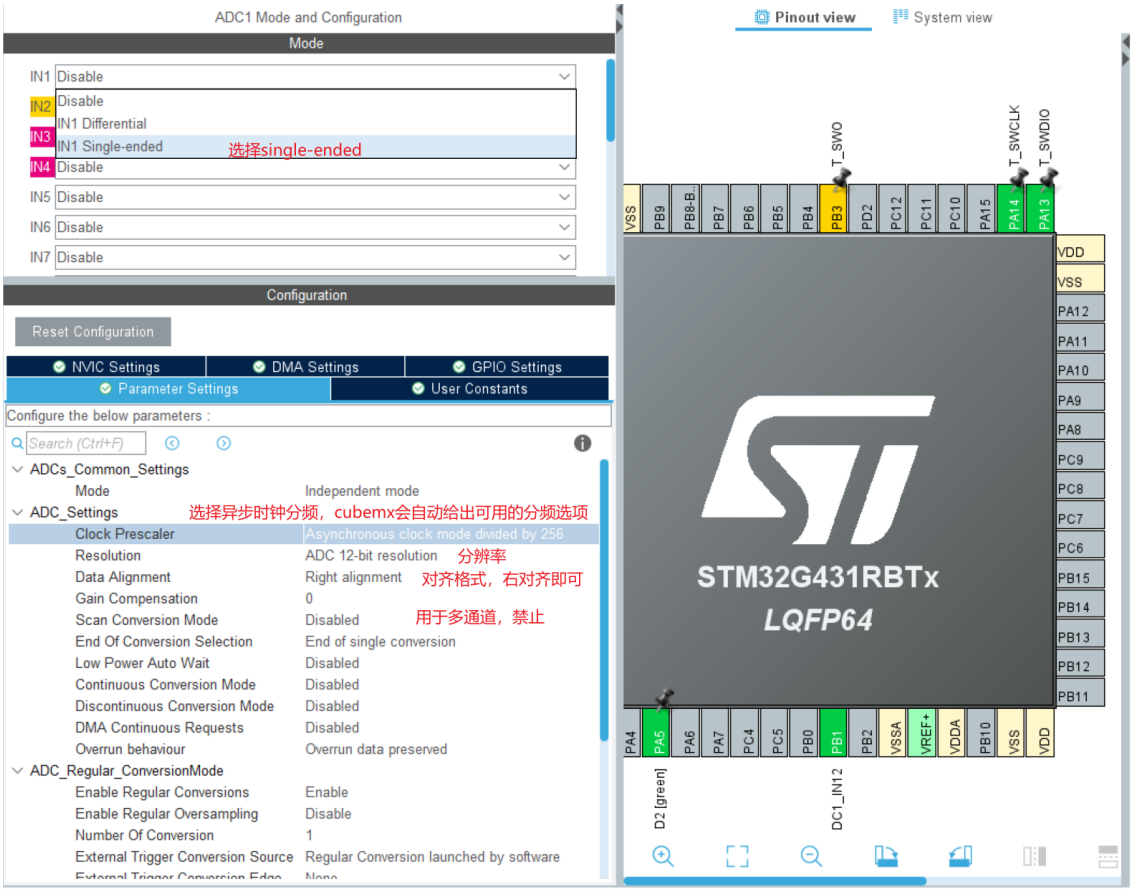
ADC和DAC

ADC

1. CubeMX配置



2. 可能使用的库函数

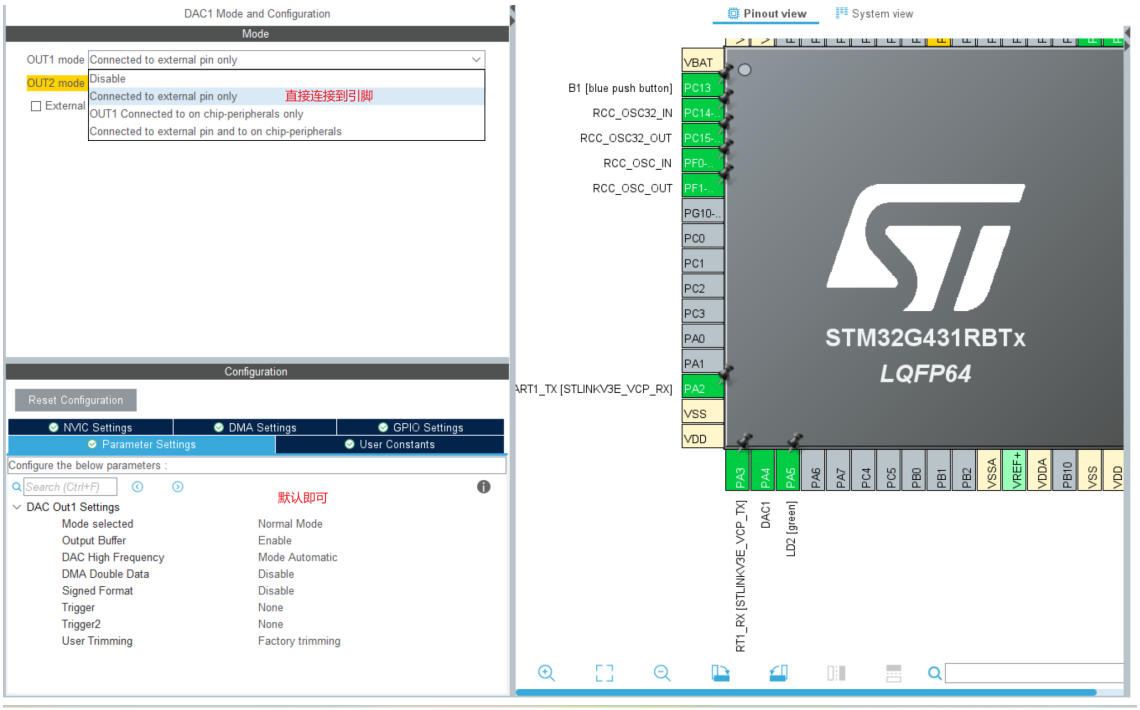
    1. **初始化和配置函数**

        - `HAL_ADC_MspInit()`：初始化ADC的底层硬件（如GPIO、时钟等）。

2. **启动和停止转换函数**
    - `HAL_ADC_Start()`：启动ADC转换（轮询模式）。
    - `HAL_ADC_Stop()`：停止ADC转换（轮询模式）。
    - `HAL_ADC_Start_IT()`：启动ADC转换（中断模式）。
    - `HAL_ADC_Stop_IT()`：停止ADC转换（中断模式）。
    - `HAL_ADC_Start_DMA()`：启动ADC转换（DMA模式）。
    - `HAL_ADC_Stop_DMA()`：停止ADC转换（DMA模式）。
3. **读取转换结果函数**
    - `HAL_ADC_GetValue()`：获取ADC转换结果。
    - `HAL_ADCEx_Calibration_Start(ADC_HandleTypeDef *hadc)`：用于启动ADC的校准过程
4. **回调函数**
    - `HAL_ADC_ConvCpltCallback()`：ADC转换完成回调函数（中断模式）。
    - `HAL_ADC_ConvHalfCpltCallback()`：ADC转换一半完成回调函数（DMA模式）。

DAC

1. CubeMX配置



2. 可能使用的库函数
    1. **初始化和配置函数**
        - `HAL_DAC_MspInit(DAC_HandleTypeDef *hdac)`
    2. **启动和停止转换函数**
        - `HAL_DAC_Start(DAC_HandleTypeDef *hdac, uint32_t Channel)`
            - `hdac`：指向DAC外设的句柄。
            - `Channel`：指定DAC通道（例如，`DAC_CHANNEL_1`或`DAC_CHANNEL_2`）。
        - `HAL_DAC_Stop(DAC_HandleTypeDef *hdac, uint32_t Channel)`

- `hdac`：指向DAC外设的句柄。
- `Channel`：指定DAC通道（例如，`DAC_CHANNEL_1`或`DAC_CHANNEL_2`）。

- `HAL_DAC_Start_DMA(DAC_HandleTypeDef *hdac, uint32_t Channel, uint32_t *pData, uint32_t Length, uint32_t Alignment)`
  - `hdac`：指向DAC外设的句柄。
  - `Channel`：指定DAC通道（例如，`DAC_CHANNEL_1`或`DAC_CHANNEL_2`）。
  - `pData`：指向要传输的数据缓冲区的指针。
  - `Length`：要传输的数据长度。
  - `Alignment`：数据对齐方式（例如，`DAC_ALIGN_12B_R`或`DAC_ALIGN_12B_L`）。
- `HAL_DAC_Stop_DMA(DAC_HandleTypeDef *hdac, uint32_t Channel)`
  - `hdac`：指向DAC外设的句柄。
  - `Channel`：指定DAC通道（例如，`DAC_CHANNEL_1`或`DAC_CHANNEL_2`）。

3. **设置和获取值函数**
   - `HAL_DAC_SetValue(DAC_HandleTypeDef *hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)`
     - `hdac`：指向DAC外设的句柄。
     - `Channel`：指定DAC通道（例如，`DAC_CHANNEL_1`或`DAC_CHANNEL_2`）。
     - `Alignment`：数据对齐方式（例如，`DAC_ALIGN_12B_R`或`DAC_ALIGN_12B_L`）。
     - `Data`：要设置的DAC值。
   - `HAL_DAC_GetValue(DAC_HandleTypeDef *hdac, uint32_t Channel)`
     - `hdac`：指向DAC外设的句柄。
     - `Channel`：指定DAC通道（例如，`DAC_CHANNEL_1`或`DAC_CHANNEL_2`）。
4. **回调函数**
   - `HAL_DAC_ConvCpltCallback(DAC_HandleTypeDef *hdac)`
     - `hdac`：指向DAC外设的句柄。
   - `HAL_DAC_ErrorCallback(DAC_HandleTypeDef *hdac)`
     - `hdac`：指向DAC外设的句柄。

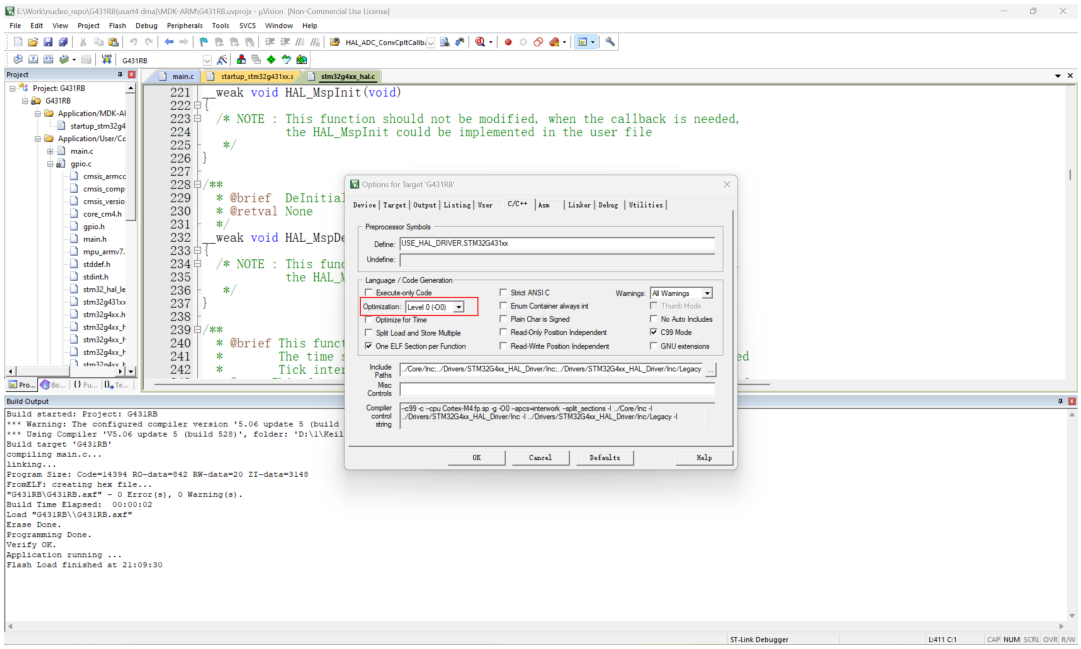
3. 正弦波采样点数组

```
uint32_t SinData[256]={2048, 2098, 2149, 2199, 2250, 2300, 2350, 2399,
2449, 2498, 2548, 2596, 2645, 2693, 2741, 2788, 2835, 2881, 2927, 2972,
3017, 3061, 3105, 3148, 3190, 3231, 3272, 3312, 3351, 3390, 3428, 3465,
3501, 3536, 3570, 3603, 3636, 3667, 3697, 3727, 3755, 3783, 3809, 3834,
3858, 3881, 3903, 3924, 3944, 3962, 3980, 3996, 4011, 4025, 4037, 4049,
4059, 4068, 4075, 4082, 4087, 4091, 4094, 4095, 4095, 4095, 4093, 4089,
4085, 4079, 4072, 4063, 4054, 4043, 4031, 4018, 4003, 3988, 3971, 3953,
3934, 3914, 3892, 3870, 3846, 3822, 3796, 3769, 3741, 3712, 3682, 3651,
3620, 3587, 3553, 3518, 3483, 3446, 3409, 3371, 3332, 3292, 3252, 3211,
3169, 3126, 3083, 3039, 2995, 2950, 2904, 2858, 2811, 2764, 2717, 2669,
2621, 2572, 2523, 2474, 2424,2375, 2325, 2275, 2224, 2174, 2124, 2073,
2023, 1972, 1922, 1872, 1821, 1771, 1721, 1672, 1622, 1573, 1524, 1475,
1427, 1379, 1332, 1285, 1238, 1192, 1146, 1101, 1057, 1013, 970, 927,
885, 844, 804, 764, 725, 687, 650, 613, 578, 543, 509, 476, 445, 414,
384, 355, 327, 300, 274, 250, 226, 204, 182, 162, 143, 125, 108, 93, 78,
65, 53, 42, 33, 24, 17, 11, 7, 3, 1, 0, 0, 2, 5, 9, 14, 21, 28, 37, 47,
59, 71, 85, 100, 116, 134, 152, 172, 193, 215, 238, 262, 287, 313, 341,
369, 399, 429, 460, 493, 526, 560, 595, 631, 668, 706, 745, 784, 824,
865, 906, 948, 991, 1035, 1079, 1124, 1169, 1215, 1261, 1308, 1355, 1403,
1451, 1500, 1548, 1598, 1647, 1697, 1746, 1796, 1846, 1897, 1947, 1998,
2048};
```

关于大作业的一些提示

1. 一些关于串口的问题
    1. **取消编译器优化！！！！**

        > sb编译器优化

        现在两块板子或许能跑了（

2. 关于编辑用户按键功能，用于更改发送的数据内容
    1. 可以在在状态机中增加一个计数器来记录按下的次数。不同次数对应不同模式，如此循环。在结构体里多增加几个状态。
    2. 可以试试自己写一个长按状态

3. 整体架构可以根据adc输入做成一个状态机试试