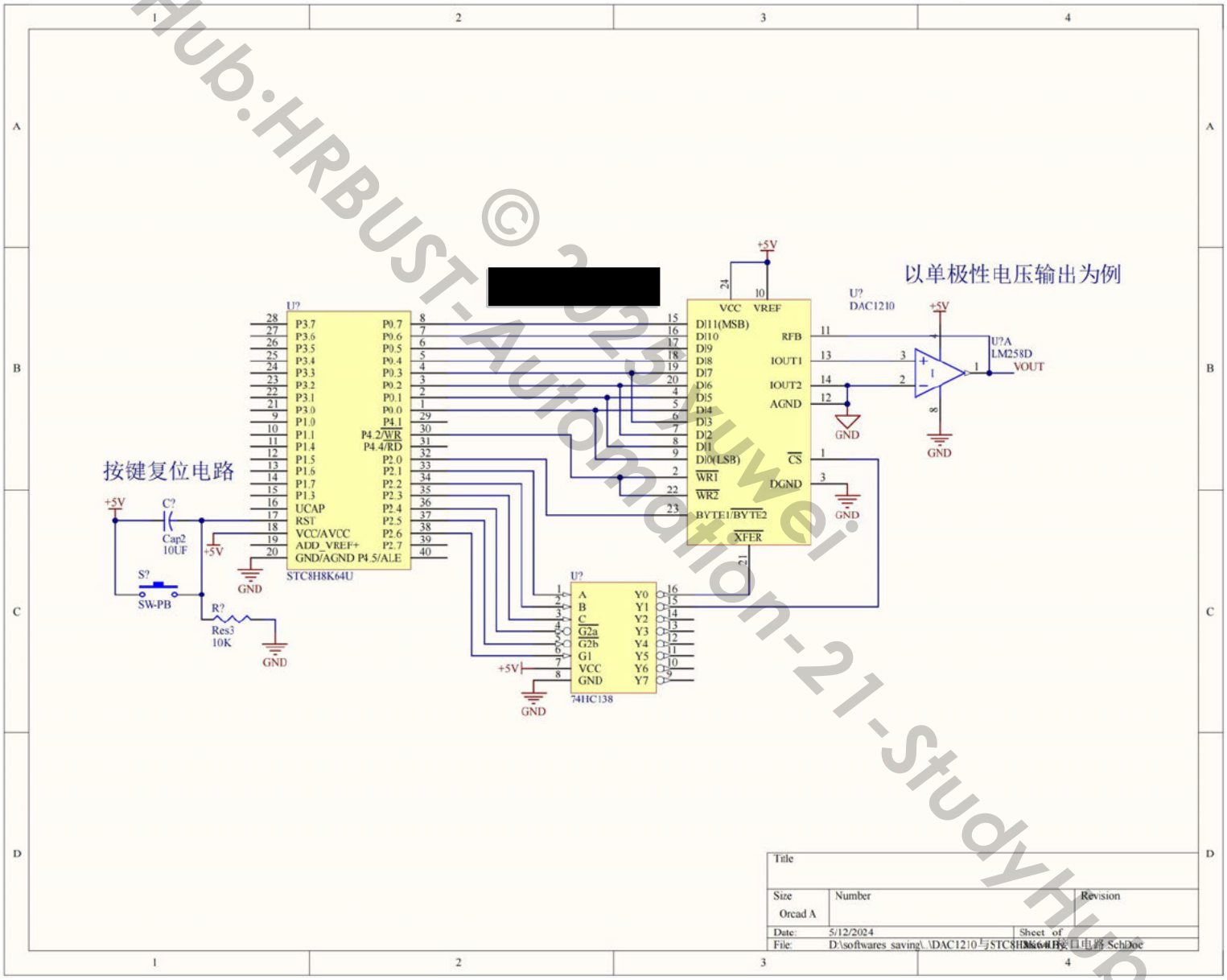
设计 DAC1210 接口电路作业

电路图



电路设计说明

①关于 DAC1210 的输入引脚接线设计

DAC1210 是 12 位 D/A 转换器，所以拥有两个输入寄存器：一个用于存放高 8 位，一个用于存放低 4 位的。题目要求的 STC8H8K64U 单片机是基于 8051 架构的，其数据总线是 8 位的，故在与 DAC1210 构成接口电路时，需要对数据总线复用。

所以我认为可以有两种连接方式：首先高八位（DI11~DI4）一一对应接到 P0 数据总线端口上，然后一种是将低 4 位(DI3~DI0)接到 P0 口的高四位(P07~P04)，另一种是将低 4 位（DI3~DI0）接到 P0 口的低四位（P03~P00）。两种方式的差别在于第二次向 DAC1210 写入低 4 位数据时，取的是 P0 的高 4 位还是低 4 位，其他地方没什么区别，这里就采用第二种方式。

②关于 DAC1210 的输出方式设计

参考课堂上老师讲过的 DAC0832，其输出方式有电压/电流输出，进一步细分可以分类成单极性输出和双极性输出。由于，单极性输出更简单和直观，所以这里以单极性输出为例来完成任务。

③关于 DAC1210 的控制字接线设计

DAC1210 的缓冲方式有三种：直通、单缓冲、双缓冲。为了更好理解 DAC1210 的工作原理，这里采用双缓冲方式，片选 nCS 作为第一级缓冲，连接端口地址译码器 74HC138 的 nY1，传输控制信号 nXFER 作为第二级缓冲，连接译码器的 nY0。

字节控制信号 BYTE1/2 连接地址线 P2.0，通过控制其高低电平来控制高低数据的输入。两个写信号 nWR1 和 nWR2，接到一起同时选通，并且连接到 STC8H8K64U 单片机上的 nWR，这样当单片机写入数据到 DAC1210 时，两个写信号是同时有效的。

电路工作原理解释

整个电路的数模转换过程是这样的：首先要选中 DAC1210，即让片选信号 nCS 有效，也就是译码器 nY1 有效。当单片机要向 DAC1210 输入被转换数据时，nWR 信号会过来，为了实现 8 位数据线传送 12 位数据，BYTE1/2 先为高电平，让其高 8 位数据被写入第一级的 8 位输入寄存器和低 4 位输入寄存器。然后再将低 4 位数据写入，此时 nWR 再次有效，BYTE1/2 为低电平，只将低 4 位数据存入低 4 位输入寄存器。然后再通过控制译码器 nY0 有效，打开内部的 12 位 DAC 寄存器，把数据送进去转换。当输出指令执行完后，DAC 寄存器又自动处于锁存状态以保持 D/A 转换的输出不变。最后，经过运算放大器将信号组合成连续模拟输出，输出端口可以直接连接到外设。

假设 12 位被转换数的高 8 位存放在 DATA 单元里，则低 4 位就存放在 DATA+1 单元里。由此可以得到三种地址分配：

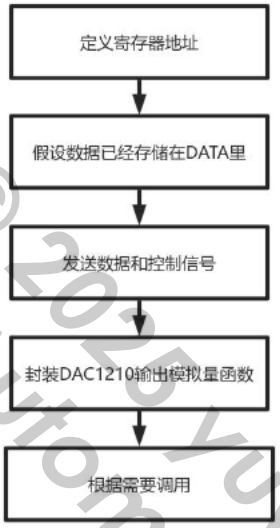
①写高 8 位数据时的地址——>选中 DAC1210，译码器 Y1 有效，且 BYTE1/2=1，表示写高 8 位数据，则地址为 4900H

②写低 4 位数据时的地址——>选中 DAC1210，译码器 Y1 有效，且 BYTE1/2=0，表示写低 4 位数据，则地址为 4800H

③将 12 位数据送进 DAC 转换器里，开始转换——>选中内部 DAC 寄存器，译码器 Y0 有效，则地址为 4000H

程序设计思路

首先在头文件里就定义 DAC1210 的寄存器地址，方便后续修改。然后需要准备好要发送的数据。这里假设数据已经存储在名为 DATA 的数组中，并且该数组在函数外部定义和初始化。函数内部通过指针访问该数组，并提取高 8 位和低 4 位有效数据。再通过使用 XBYTE 来访问外部数据存储器中的 DAC1210 寄存器，并发送数据和控制信号。首先发送高 8 位数据，然后发送低 4 位有效数据，最后发送控制信号以启动转换。根据需要，可以在发送完数据和控制信号后添加延时或其他后处理操作，以确保 DAC1210 正确完成转换。下图是以流程图的形式来说明。



程序源代码

头文件（dac1210.h）

```
#ifndef  DAC1210_H
#define  DAC1210_H

#include <STC8H8K64U.h>        //  包含 STC8H8K64U 寄存器定义的头文件

#define  DAC1210_HIGH_BYTE_ADDR  0x4900    //  写高 8 位数据寄存器地址
#define  DAC1210_LOW_BYTE_ADDR  0x4800      //  写低 4 位数据寄存器地址
#define  DAC1210_CONTROL_ADDR  0x4000       //  控制开启 DAC 寄存器地址，打开第二级缓冲

void  DAC1210_Out_A(unsigned  char  *data)      //  声明开启 DAC1210 输出模拟量的函数

#endif
```

源文件（dac1210.c）

```c
#include "dac1210.h"

//假设 12 位被转换数的高 8 位存放在 DATA 单元里，则低 4 位就存放在 DATA+1 单元里
//假设 data 指向 DATA 内存区

//  开启 DAC1210 输出模拟量的函数
void DAC1210_Out_A(unsigned char *data) {

        unsigned char highByte = *data;
//  读取数据总线中的高 8 位数据
        unsigned char lowByte = *(data + 1) << 4;
//  读取低 4 位作为有效数据

        //  发送高 8 位数据到 DAC1210
        XBYTE[DAC1210_HIGH_BYTE_ADDR] = highByte;

        //  发送低 4 位数据到 DAC1210
        XBYTE[DAC1210_LOW_BYTE_ADDR] = lowByte;

        //  打开第二级缓冲，将数据送到 DAC 寄存器里开始转换
        XBYTE[DAC1210_CONTROL_ADDR] = 0x01;
//假设写入 0x01 来启动缓冲，实际理论上应该是写入任意值进去都可以启动
}
```