

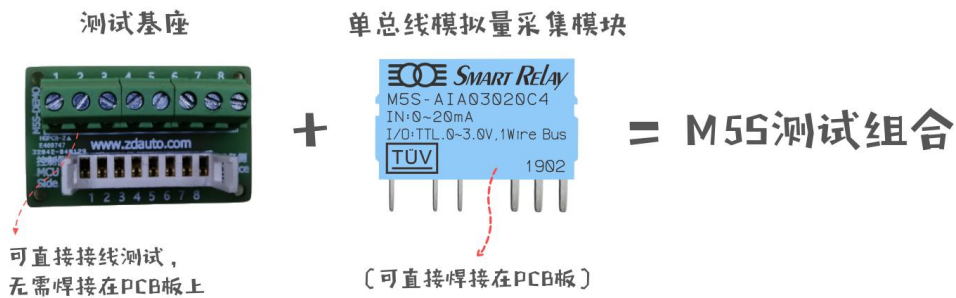
单总线模拟量信号采集测试组合

M5S-AIA03020C4 测试组合

模拟量信号 0~20mA → TTL 数据 (精度 2%)

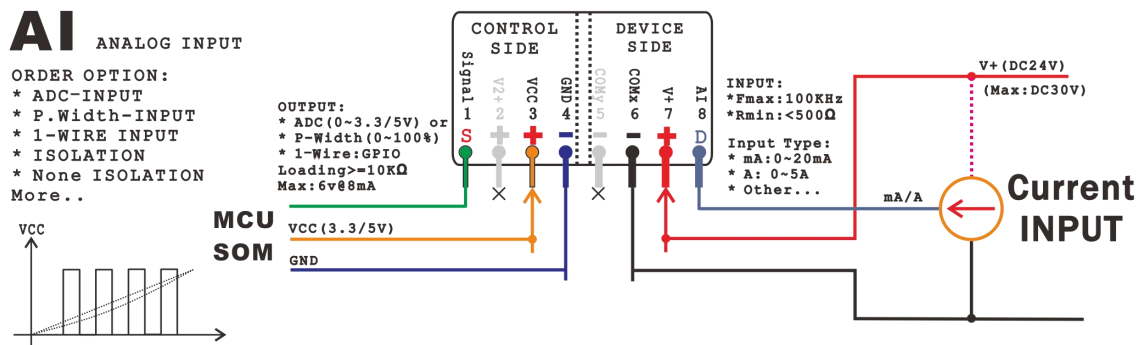
这是一款模拟量信号采集模块，插入测试基座可以直接接线使用，无需焊接。

更方便使用者连接树莓派，增加树莓派的信号采集功能。



应用电路图

单总线光隔电流模拟量输入(DC 电流, 6-7-8 脚)，控制侧(TTL 电平数据, 1-3-4 脚)

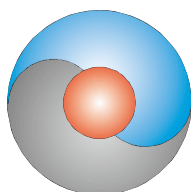


型号参数表

单总线光隔电流模拟量输入(DC 电流, 6-7-8 脚)，控制侧(TTL 电平数据, 1-3-4 脚)

型号	Control Side (控制侧, 输出)				曲线	隔离	Device Side (设备侧, 输入)				电路索引
	电压(1)	电流(1)	电源	精度			电压(8)	电流(8)	电源	频响	
M5S-AIA03020C4	TTL 数据	<5mA	3.3V	2%	线性	●	0~20mA	24V	0.2KHz	C4	

*更多型号可在商城内选购，需要更多信息请联系客服



ZDAUTO®

智达自动化

M5S-AIA03020C4

单总线模拟量模块

使用手册

www.zdauto.com

中山智达自动化科技有限公司

2019-10

目录

1. 概述	1
1.1 一般说明	1
1.2 特性	2
1.3 引脚排列	3
2. 命令及寄存器读写	4
2.1 区分命令和寄存器的定义	4
2.2 命令一览	4
2.3 寄存器一览	5
2.4 数据校验方法	5
2.5 单点通信时写命令	6
2.6 单点通信时写寄存器数据	8
2.7 单点通信时读寄存器数据	10
3. 通信时序	12
3.1 复位信号	12
3.2 主机发写时隙	15
3.3 主机发读时隙	17
4. 电气特性及封装	20
4.1 实物封装及 PCB 布局	20
4.2 接线图	20
5. 应用案例	21
5.1 Raspberry	21
6. 固件更新	23
6.1 安装软件	23
6.2 STlink 接线	23
6.3 配置软件	24

1. 概述

1.1 一般说明

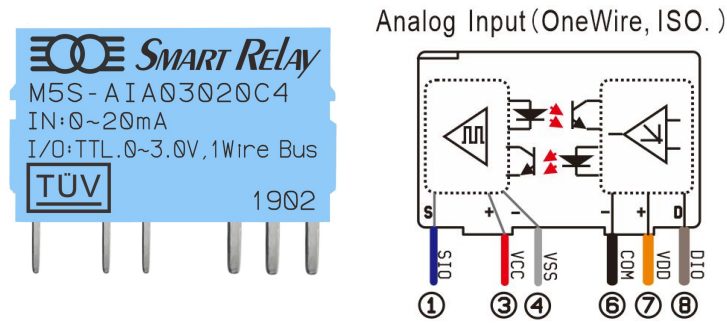


图 1 M5S-AIA03020C4 实物

M5S-AIA03020C4 是一款 10 位分辨率的模拟量采集模块，0-20mA 的电流采集。

信息经过单线接口送入 M5S 或从 M5S 送出，因此从中央处理器到 M5S 通信仅需连接一条线。单线读写和完成模拟量转换所需的电源是分开的，在控制侧需要供给 3.3V 电源 (PIN3, VCC) 用于单总线通信，在设备侧需要供给 24V (PIN7, V+) 用于模块正常工作。

每一个 M5S 的序列号 (silicon serial number) 可由用户自定义 (出厂时为 0x00, 可定制)，因此多个 M5S 可以存在于同一条单线总线上。在 IO 引脚数量紧张的情况下 (如树莓派)，可由一路引脚通过寻序列号的方式，对多个 M5S 模块进行读写。

1.2 特性

1. 独特的单线接口，只需 1 个通信接口即可通信
2. 用户可自定义模块序列号
3. 多点 (multi-drop) 能力使分布式模拟量检测应用得以简化，并解放某些场合 IO 数量紧张的情况
4. 不需要外部元件
5. 控制侧与设备侧光电隔离，符合工业电气安全标准
6. 简单 PCB 布线
7. M5S-AIA03020C4 测量范围从 0 至 20mA，增量值为 0.02mA
8. 以 10 位数字值方式读出模拟值
9. 在 1 微秒 (典型值) 内把电压或电流变换为数字
10. 具有滤波器算法或其他扩展功能，用户可通过读写寄存器的方式操作
11. 用户可定义的，非易失性的模拟阈值告警设置
12. 告警搜索命令识别和寻址模拟值在编定的极限之外的器件 (模拟值告警情况)
13. 应用范围包括恒温控制，工业系统，消费类产品，家电类产品或任何电控系统

1.3 引脚排列

Analog Input (OneWire, ISO.)

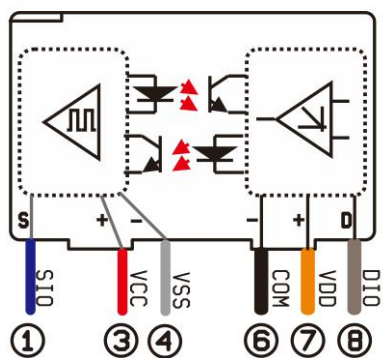


表 1 M5S-AIV03010C4,-AIA03020C4 引脚表

引脚号	名称	描述
1	SIO	单总线 GPIO 脚
3	VCC	控制侧提供 3.0-5.5V 电压 (与 2 号脚连通)
4	VSS	控制侧 GND
5	SWIM	内核更新引脚 (特定要求可选此引脚)
6	COM	设备侧 GND
7	VDD	模块供电: 额定 12VDC
8	DIO	设备侧接入输入电压或者输入电流

2. 命令及寄存器读写

2.1 区分命令和寄存器的定义

M5S-AIA03020C4 通信分写时隙和读时隙，写时隙分别有写命令、写寄存器、读寄存器，具体区别如表 2。

表 2 M5S-AIA03020C4 写时隙区别表

高 2 位	低 6 位	类别	描述
00	命令号	写命令	低 6 位为对应的命令号，直接发命令号，再读回 8bit 校验码（命令号），模块作相应的响应。
01	寄存器号	读寄存器	低 6 位为要读取数据的寄存器地址，发寄存器地址后，进入读时隙读 16bit 数据，再进入读时序，读 8bit 的校验码（地址号+16bit 数据的高 8bit+16bit 数据的低 8bit）。
10	寄存器号	写寄存器	低 6 位为要写入数据的寄存器地址，发寄存器地址后，进入写时隙写 16bit 数据和 8bit 校验码（地址号+16bit 数据的高 8bit+16bit 数据的低 8bit），然后进入读时隙，读回 8bit 校验码。

2.2 命令一览

表 3 M5S-AIA03020C4 命令表

命令号	功能	描述
0x3F	启动模块，模块进入工作	刚上电时模块不会工作，需要发送此命令激活。
0x3E	关闭模块，模块退出工作	当工作完成后，可发此命令让模块停止工作，减少能耗。
0x3D	启动快速读模式（仅限 AI 模块）	让模块进入快速读模式，在该模式下，只能读 0x3F 寄存器，不能写或读其他寄存器，且在读 0x3F 寄存器时，不用输入地址号。
0x3C	关闭快速读模式（仅限 AI 模块）	退出快速读模式。

注：发送命令号时，高 2 位是 00,低 6 位为命令号。

2.3 寄存器一览

表 4 M5S-AIA03020C4 寄存器表

寄存器号	功能	描述
0x3F	瞬时采集到的模拟量值的 高 2 位	存储瞬时采集到的模拟量值，范围为 0-1024.
0x01	版本号	模块版本号 Vx.y x=高 8 位，y=低 8 位

注：读寄存器号时，高 2 位是 01,低 6 位为寄存器号。写寄存器号时，高 2 位是 10,低 6 位为寄存器号。

2.4 数据校验方法

M5S 读取数据或设置数据都要校验，具体如下：

- 1.写 1 字节命令后，读回 1 字节数据，若读回的数等于命令号，则写命令成功。
- 2.向寄存器写入数据时，要先将 3 字节（顺序：1 字节寄存器号，2 字节数据（高位在前））做 CRC 校验，得出 CRC 校验码。然后再将这 4 字节按寄存器号-数据高位-数据低位-CRC 校验码的顺序发送给 M5S，再读回 1 字节数据，若读回的数据等于 CRC 校验码则向寄存器写入数据成功。
- 3.读取寄存器中的数据时，先写 1 字节寄存器号（要读出的寄存器），再读回 3 个字节（数据高位-数据低位-CRC 校验码），然后将 3 个字节做 CRC 校验（数据高位-数据低位-寄存器地址）得到 CRC1，若 CRC1 等于读回的 CRC 值，则读出成功。

2.5 单点通信时写命令

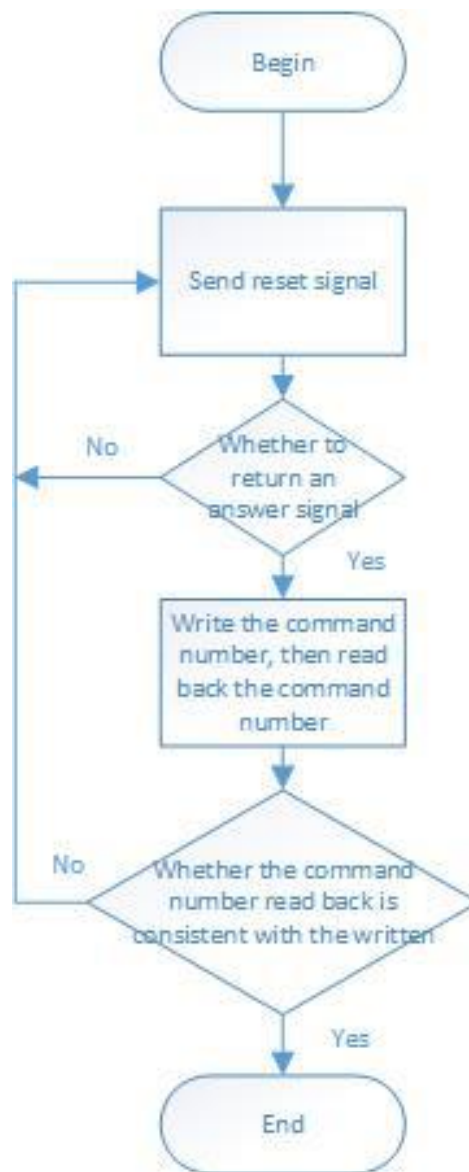


图 3 M5S-AIA03020C4 单点通信写命令流程图

写命令例程如下：

```
int ZD_M5S_send_nCmd( int nPin, int nCmd )
{
    int crc, cnt, t;
    cnt = 10;
    while( cnt-- )
    {
        if( nCmd == 0x3C )
            ZD_ONE_WIRE_BUS_Rst( nPin, 200 );           // Delay 2 ms
        else
            ZD_ONE_WIRE_BUS_Rst( nPin, 50 );           // Delay 500us

        if( ZD_ONE_WIRE_BUS_Check( nPin ) )
        {
            #if DEBUG_MSG
                printf( "fail send nCmd no M5S\r\n" );
            #endif
            //Module exiting fast mode failed, it should be restarted
            if( ( cnt <= 1 ) && ( nCmd == 0x3C ) )
            {
                cnt = 10;
                nCmd = 0x3F;
            }
            continue;
        }
        ZD_ONE_WIRE_BUS_Write_Byte( nPin, nCmd );
        crc = ZD_ONE_WIRE_BUS_Read_Byte( nPin );
        if( crc != nCmd )
        {
            #if DEBUG_MSG
                printf( "fail send nCmd error crc\r\n" );
            #endif
            continue;
        }
        break;
    }
    if( cnt <= 0 ) return -1;
    return 0;
}
```

2.6 单点通信时写寄存器数据

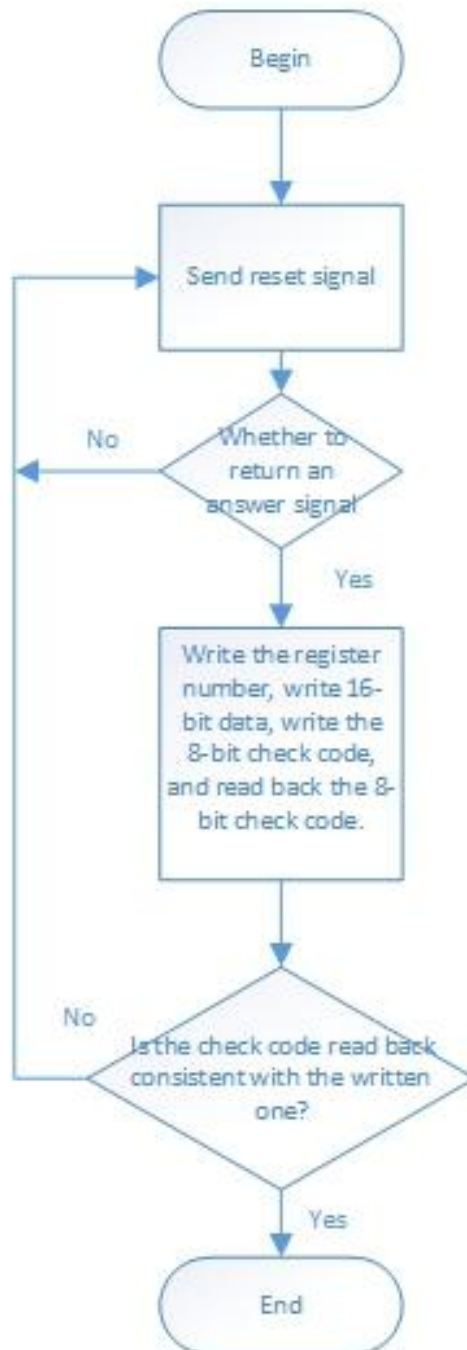


图 3 M5S-AIA03020C4 单点通信写寄存器数据流程图

写寄存器数据例程如下：

```
int ZD_M5S_set_value( int nPin, int add, int value )
{
    int crc, crc1, cnt;
    unsigned char crcbuf[3];
    cnt = 10;

    crcbuf[0] = ( 0xFF & ( value >> 8 ) );
    crcbuf[1] = ( value & 0xFF );
    crcbuf[2] = add;
    crc = cal_crc_table( crcbuf, 3 );
    while( cnt-- )
    {
        ZD_ONE_WIRE_BUS_Rst( nPin, 50 );           // Delay 500 us
        if( ZD_ONE_WIRE_BUS_Check( nPin ) )
        {
#ifdef DEBUG_MSG
            printf( "fail set value no M5S\r\n" );
#endif
            continue;
        }
        ZD_ONE_WIRE_BUS_Write_Byte( nPin, ( add | 0x80 ) );
        ZD_ONE_WIRE_BUS_Write_Byte( nPin, ( value >> 8 ) );
        ZD_ONE_WIRE_BUS_Write_Byte( nPin, ( value & 0xFF ) );
        ZD_ONE_WIRE_BUS_Write_Byte( nPin, crc );
        crc1 = ZD_ONE_WIRE_BUS_Read_Byte( nPin );
        if( crc1 != crc )
        {
#ifdef DEBUG_MSG
            printf( "fail set value error crc\r\n" );
#endif
            continue;
        }
        break;
    }
    if( cnt <= 0 ) return -1;
    return 0;
}
```

2.7 单点通信时读寄存器数据

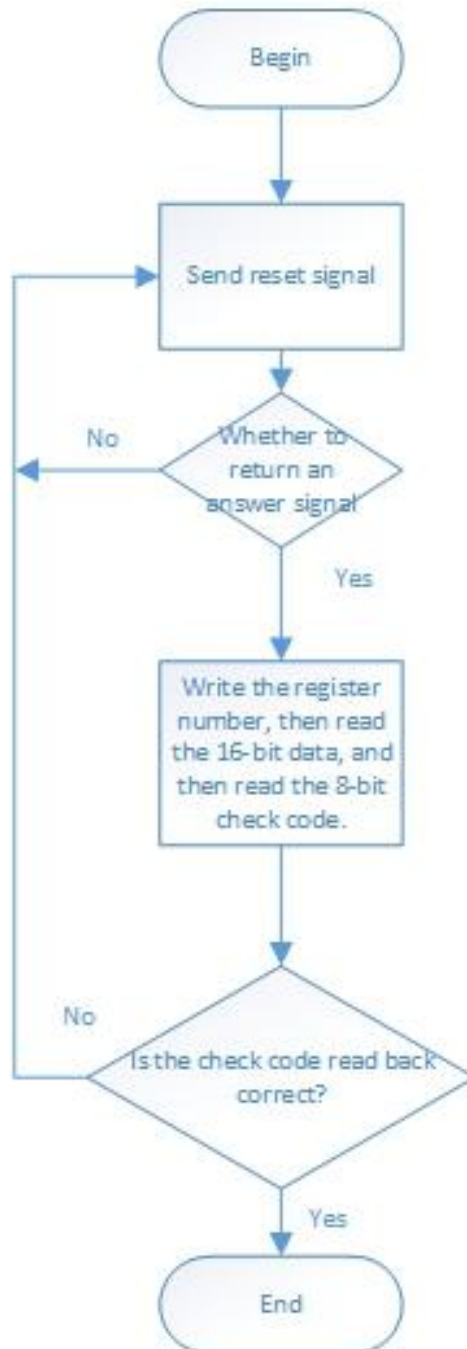


图 3 M5S-AIA03020C4 单点通信写寄存器数据流程图

读寄存器数据例程如下：

```
int ZD_M5S_get_value( int nPin, int add )
{
    int crc,crc1, cnt, value;
    unsigned char crcbuf[3];
    cnt = 10;

    while( cnt-- )
    {
        ZD_ONE_WIRE_BUS_Rst( nPin, 50 );           // Delay 500 us
        if( ZD_ONE_WIRE_BUS_Check( nPin ) )
        {
            #if DEBUG_MSG
                printf( "fail get value no M5S\r\n" );
            #endif
            continue;
        }
        ZD_ONE_WIRE_BUS_Write_Byte( nPin, ( add | 0x40 ) );
        value = ZD_ONE_WIRE_BUS_Read_Byte( nPin );
        value <<= 8;
        value |= ZD_ONE_WIRE_BUS_Read_Byte( nPin );
        crc1 = ZD_ONE_WIRE_BUS_Read_Byte( nPin );
        crcbuf[0] = (0xFF & ( value >> 8 ) );
        crcbuf[1] = ( value & 0xFF );
        crcbuf[2] = add;
        crc = cal_crc_table( crcbuf, 3 );
        if( crc != crc1 )
        {
            #if DEBUG_MSG
                printf( "fail get value error crc\r\n" );
            #endif
            continue;
        }
        break;
    }
    if( cnt <= 0 ) return -1;
    return value;
}
```

3. 通信时序

3.1 复位信号

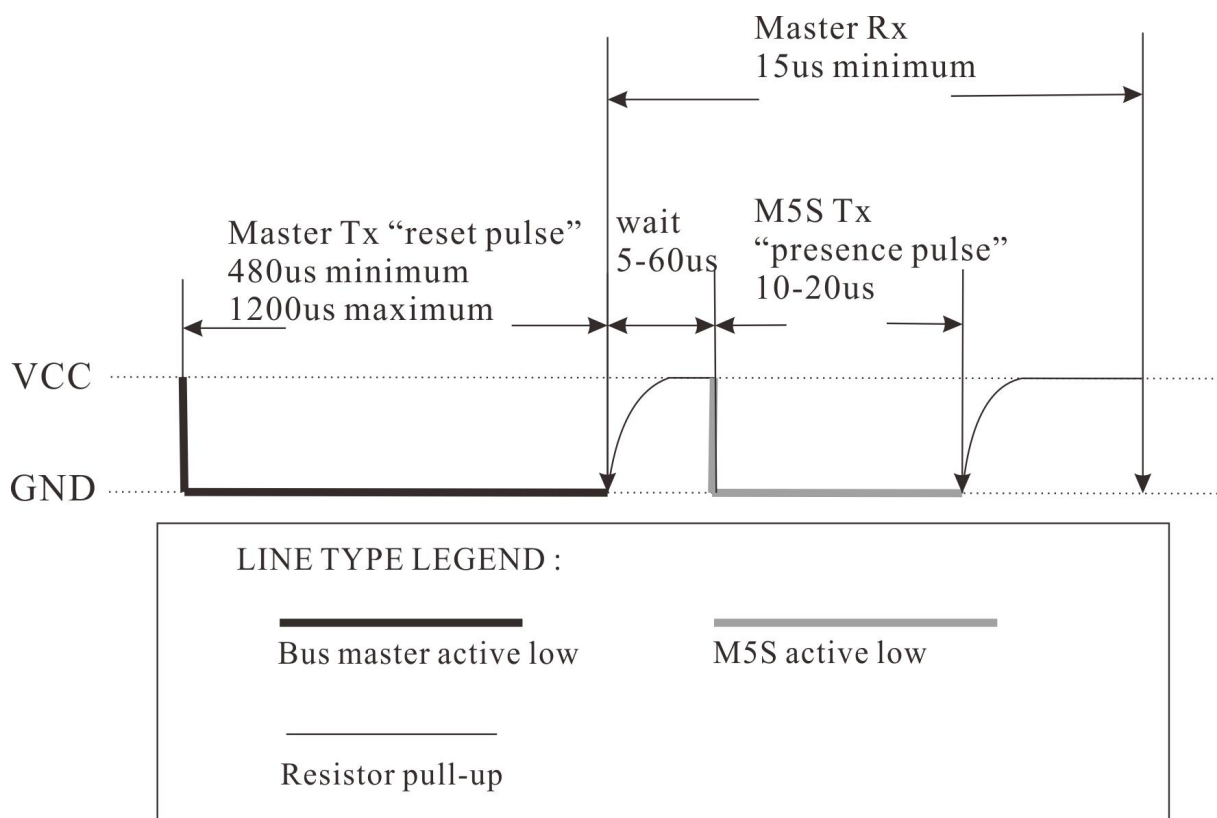


图 3 M5S-AIA03020C4 复位信号时序图

主机先拉低 480-1200us 的低电平，然后释放总线 25-60us，M5S 会拉低 10-20us 的低电平响应给主机，再保持高电平。主机成功收到响应后，跟随通过写时隙对 M5S 操作。

复位信号和检测是否存在模块的函数如下：

```
//-----
// brief: Generate a high level reset signal at any time, in units of 1us
// Function name: void ZD_ONE_WRIE_BUS_Rst( int num, int time )
// Param: Pinnum Raspberry Pi GPIO pin, corresponding to the wiring number, time: Reset time
// retval: None
// other : This function is used up to call ZD_ONE_WRIE_BUS_Check()1us
//
// Revision time : 2018/6/11
//-----
void ZD_ONE_WRIE_BUS_Rst( int num, int time )
{
    int retry = time;
    ONE_WRIE_BUS_IO_OUT( num ); //Set the communication pin to the output state
    ONE_WRIE_BUS_OUT( num, 0 ); //Pull pin low
    while( retry-- )delayMicroseconds( 1 ); //Delay 1000us
    ONE_WRIE_BUS_OUT( num, 1 ); //Pull high pin
}
//-----
// brief: Check if the M5S sends a presence signal
// Function name: int ZD_ONE_WRIE_BUS_Check( int num )
// Param: Pinnum Raspberry Pi GPIO pin, corresponding to the wiring number
// retval: 0: Yes 1: No
// other : This function is used up to call ZD_ONE_WRIE_BUS_Check()1us
//
// Revision time : 2018/6/11
//-----
int ZD_ONE_WRIE_BUS_Check( int num )
{
    int retry = 0;
    ONE_WRIE_BUS_IO_IN( num ); // Set the communication pin to the input state
    while( ONE_WRIE_BUS_IN( num ) && retry < 350 )
    {
        retry++;
        delayMicroseconds( 1 );
    };
    if( retry >= 350 )
    {
#ifdef debug //-----
        printf( "fail Check 1----- \r\n" );
#endif //-----
        return 1;
    }
}
```



```

//-----
// brief: Check if the M5S sends a presence signal
// Function name: int ZD_ONE_WRIE_BUS_Check( int num )
// Param:  Pinnum Raspberry Pi GPIO pin, corresponding to the wiring number
// retval: 0: Yes 1: No
// other : This function is used up to call ZD_ONE_WRIE_BUS_Check()1us
//
// Revision time : 2018/6/11
//-----
int ZD_ONE_WRIE_BUS_Check( int num )
{
    int retry = 0;
    ONE_WRIE_BUS_IO_IN( num ); // Set the communication pin to the input state
    while( ONE_WRIE_BUS_IN( num ) && retry < 350 )
    {
        retry++;
        delayMicroseconds( 1 );
    };
    if( retry >= 350 )
    {
#if debug //-----
        printf( "fail Check 1----- \r\n" );
#endif //-----
        return 1;
    }
    else retry = 0;
    while( !ONE_WRIE_BUS_IN( num ) && retry < 350 )
    {
        retry++;
        delayMicroseconds( 1 );
    };
    if( retry >= 300 )
    {
#if debug //-----
        printf( "fail Check 2----- \r\n" );
#endif //-----
        return 1;
    }
    retry = 70;
    while( retry-- )delayMicroseconds( 1 );
    return 0;
}

```

3.2 主机发写时隙

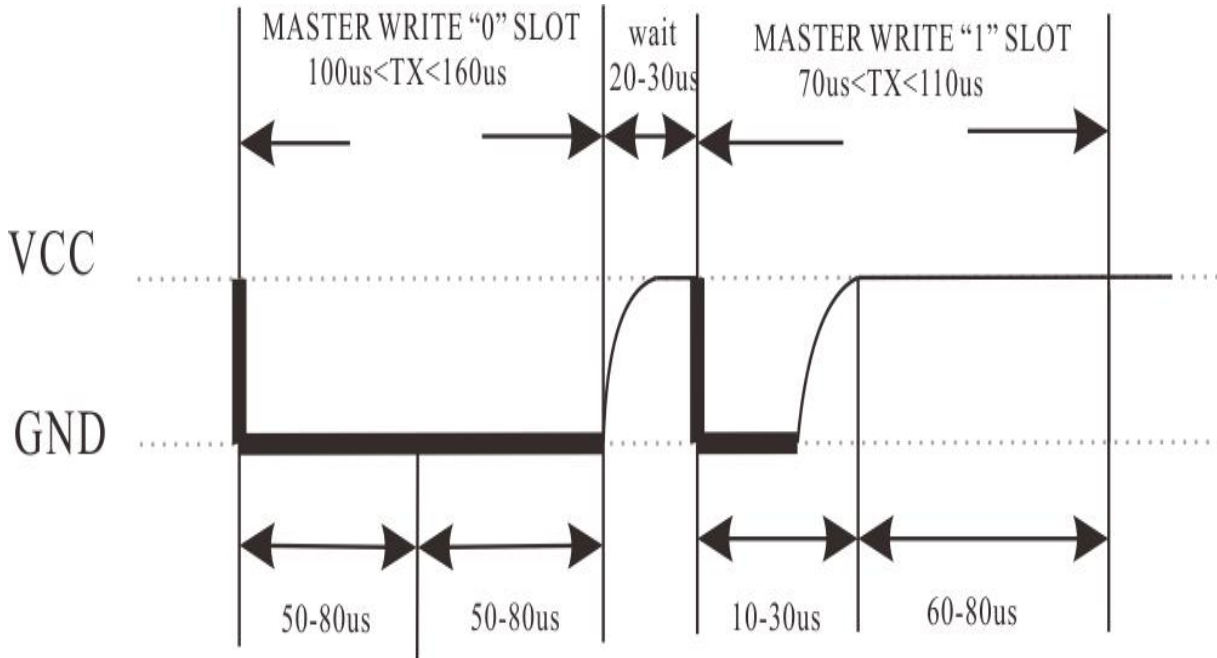


图 4 M5S-AIA03020C4 写 “0” 、 “1” 信号时序图

主机向 M5S 写 “0” ，先拉低总线 50-80us 后，再拉高总线 50-80us。

主机向 M5S 写 “1” ，先拉低总线 10-30us 后，再拉高总线 60-80us。

M5S 会在主线拉低电平后计算低电平时间是否超过 40us ，是则为 0 ，否则为 1。且低电平时间不能超过 130us ，否则为无效信号。

主机写时隙写入 1byte 例程如下：

```
//-----  
// brief: Write one byte of M5S  
// Function name: void ZD_ONE_WRIE_BUS_Write_Byte( int num , int dat )  
// Param:  Pinnum:Raspberry Pi GPIO pin, corresponding to the wiring number dat:Byte written  
// retval: None  
// other : Low starting  
//  
// Revision time : 2018/6/11  
//-----  
void ZD_ONE_WRIE_BUS_Write_Byte( int num , int dat )  
{  
    int j;  
    int testb;  
    int retry;  
    ONE_WRIE_BUS_IO_OUT( num );  
    for( j = 1; j <= 8; j++ )  
    {  
        testb = dat & 0x01;  
        dat = dat >> 1;  
        if( testb )  
        {  
            ONE_WRIE_BUS_OUT( num, 0 ); // Write 1  
            retry = 10;  
            while( retry-- )delayMicroseconds( 1 );  
            ONE_WRIE_BUS_OUT( num, 1 );  
            retry = 70;  
            while( retry-- )delayMicroseconds( 1 );  
        }  
        else  
        {  
            ONE_WRIE_BUS_OUT( num, 0 ); // Write 0  
            retry = 60;  
            while( retry-- )delayMicroseconds( 1 );  
            ONE_WRIE_BUS_OUT( num, 1 );  
            retry = 50;  
            while( retry-- )delayMicroseconds( 1 );  
        }  
    }  
}
```

3.3 主机发读时隙

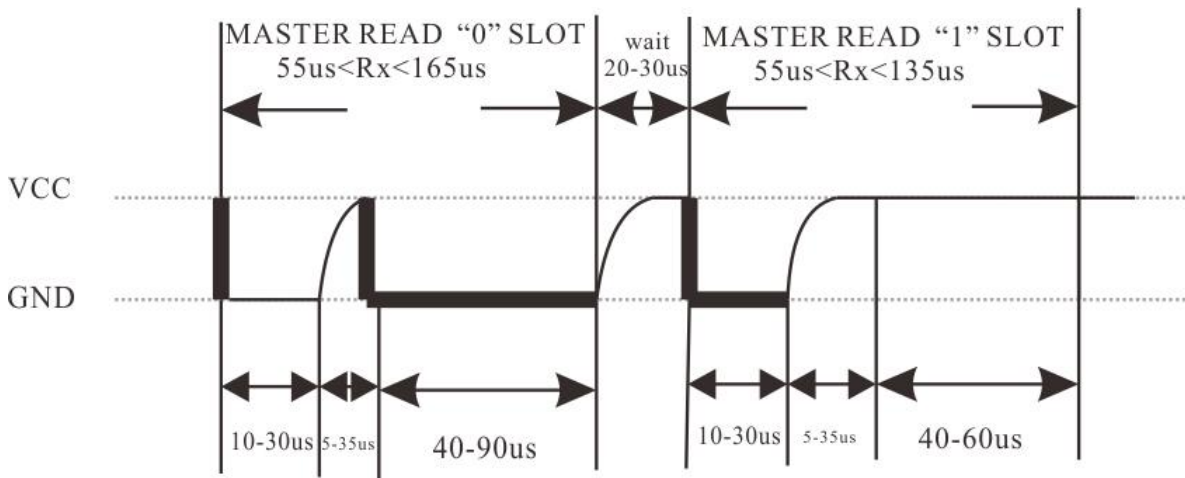


图 5 M5S-AIA03020C4 读 “0” “1” 信号时序图

主机向 M5S 先拉低总线 10-30us 后，释放总线，然后如果要读的数为 “0” 则 M5S 拉低总线 40-90us，然后在保持高电平。

主机向 M5S 先拉低总线 10-30us 后，释放总线，然后如果要读的数为 “1” 则 M5S 保持高电平 40-60us。

M5S 会在总线拉低电平后计算低电平时间是否保持 30us，是则开始读数。且低电平时间不能超过 130us，否则为无效信号。要读的数如果为 “0”，M5S 会在总线释放后拉低总线，否则保持总线高电平。

主机读时隙读入 1bit 和 1byte 的例程：

```
//-----
// brief: Read a byte from M5S
// Function name: int ZD_ONE_WRIE_BUS_Read_Byte( int num )
// Param:   Pinnum Raspberry Pi GPIO pin, corresponding to the wiring number
// retval:  Byte to read
// other :  M5S is low starting
//
// Revision time : 2018/6/11
//-----
int ZD_ONE_WRIE_BUS_Read_Byte( int num )
{
    int i, j, dat;
    dat = 0;
    for( i = 1; i <= 8; i++ )
    {
        j = ZD_ONE_WRIE_BUS_Read_Bit( num );
        dat = ( j << 7 ) | ( dat >> 1 );
    }
    return dat;
}
//-----
// brief: Read a bit from M5S
// Function name: int ZD_ONE_WRIE_BUS_Read_Bit( int num )
// Param:   Pinnum Raspberry Pi GPIO pin, corresponding to the wiring number
// retval:  Number to read
// other :
//
// Revision time : 2018/6/11
//-----
int ZD_ONE_WRIE_BUS_Read_Bit( int num )
{
    int data;
    int retry;
    ONE_WRIE_BUS_IO_OUT( num );    //Set the communication pin to the output state
    ONE_WRIE_BUS_OUT( num, 0 );    //Pull down
    retry = 10;
    while( retry-- )delayMicroseconds( 1 );
    ONE_WRIE_BUS_IO_IN( num );    //Set the communication pin to the input state
    retry = 20;
    while( retry-- )delayMicroseconds( 1 );
    if( ONE_WRIE_BUS_IN( num ) )
    {
```

```
    data = 1; //Detect communication pin level
    retry = 50;
    while( retry-- )delayMicroseconds( 1 );
}
else
{
    data = 0;
    retry = 90;
    while( retry-- )delayMicroseconds( 1 );    //Delay 90us
}
return data;
}
```

4. 电气特性及封装

4.1 实物封装及 PCB 布局

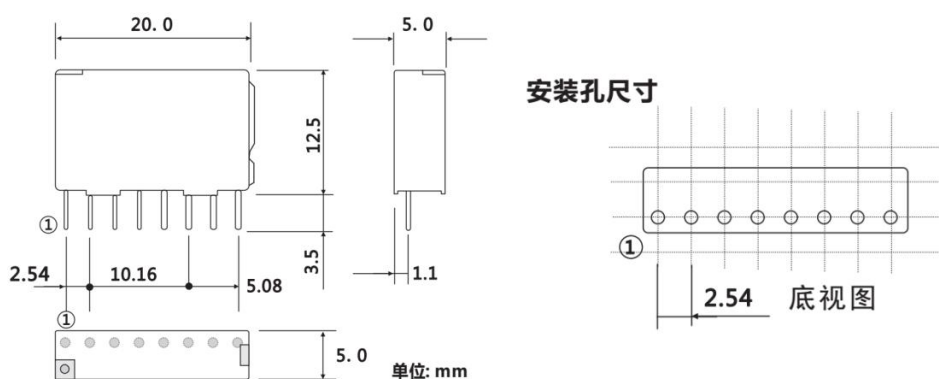
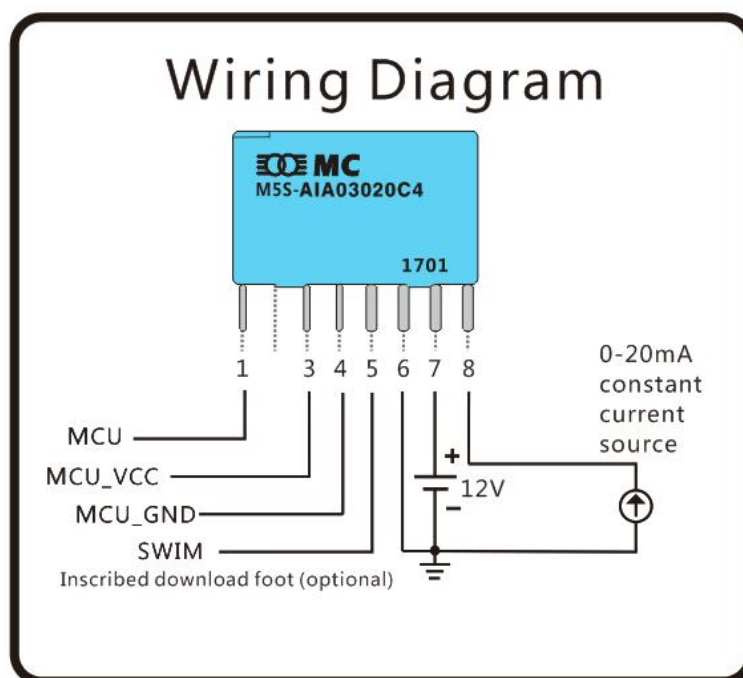


图 6 M5S-AIA03020C4 外形尺寸， 接线图和 PC 板布局图

4.2 接线图



5. 应用案例

5.1 Raspberry

1.在 linux 环境下安装 wiringPi 库（C 语言运行脚本），步骤如下：

如果在你的平台上还没有安装 GIT 工具，可以输入以下命令：

```
sudo apt-get install git-core
```

如果在这个过程中出现错误，尝试更新软件，例如输入以下指令：

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

紧接着可以通过 GIT 获得 wiringPi 的源代码

```
git clone git://git.drogon.net/wiringPi
```

若需要更新 wiringPi。

```
cd wiringPi
```

```
git pull origin
```

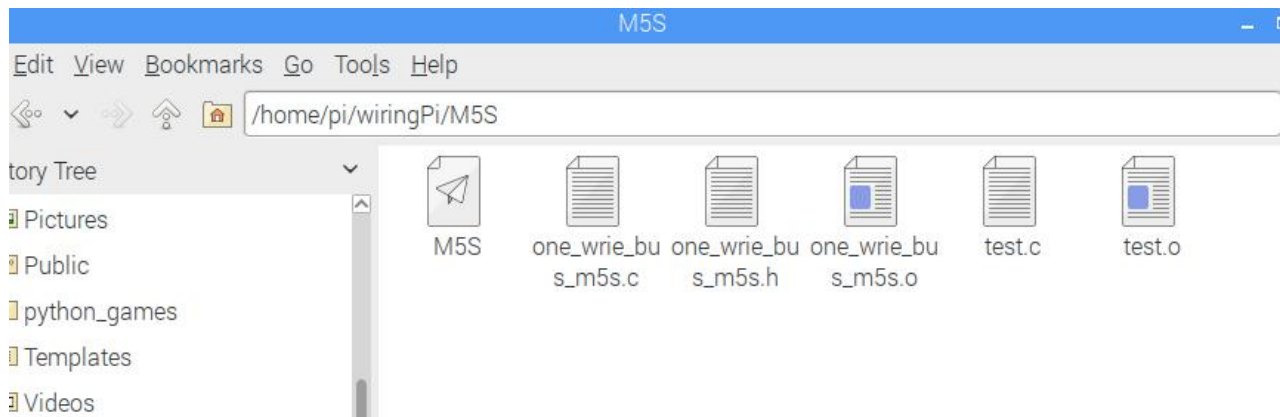
进入 wiringPi 目录并安装 wiringPi

```
cd wiringPi
```

```
./build
```

build 脚本会帮助你编译和安装 wiringPi

2.将 M5S/Sample 下的 M5Sv1.6-English 文件发送到树莓派，文件在 linux 目录如下：



3.打开命令终端输入：

```
cd wiringPi
```

```
cd M5Sv1.6-English
```

```
gcc -c test.o
```

```
gcc -c one_wrie_bus_m5s.c
```

```
gcc test.o one_wrie_bus_m5s.o -o M5S -l wiringPi
```

```
sudo ./M5S
```

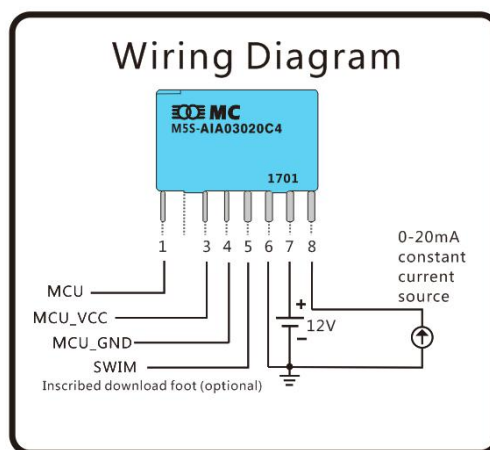
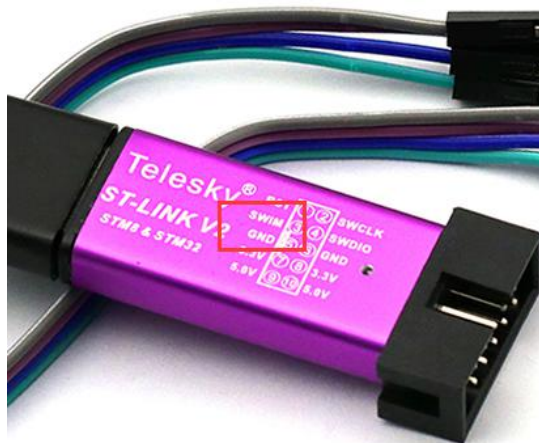
运行程序。

6. 固件更新

6.1 安装软件

- 1、软件目录在 M5S/tool 中，解压 en.stvp-stm8。
- 2、解压后点击安装。

6.2 STlink 接线

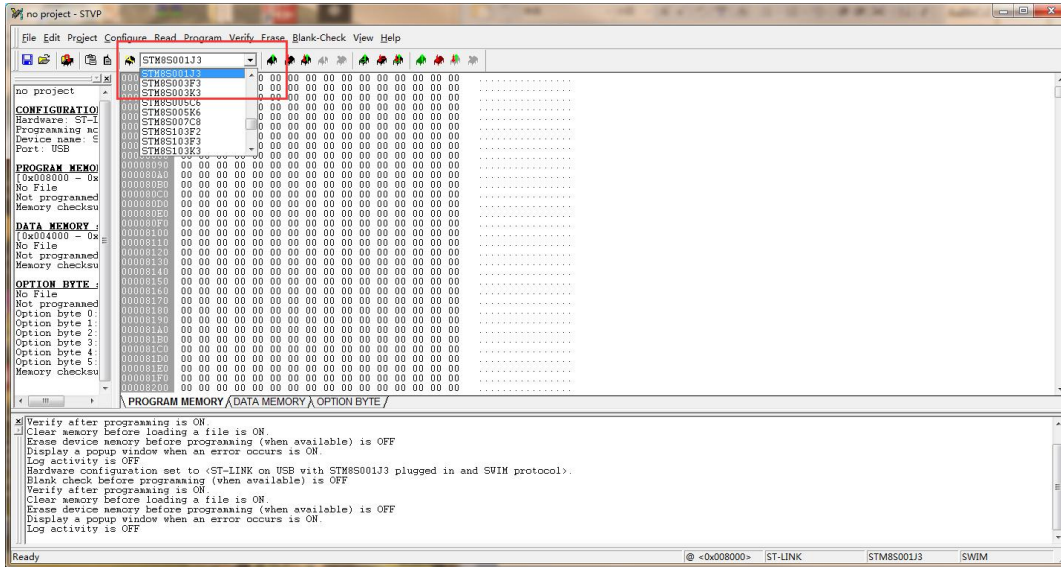


M5S 的引脚号	连接说明
5	STlink_SWIM
6	STlink_GND 和 GNV
7	12V

6.3 配置软件

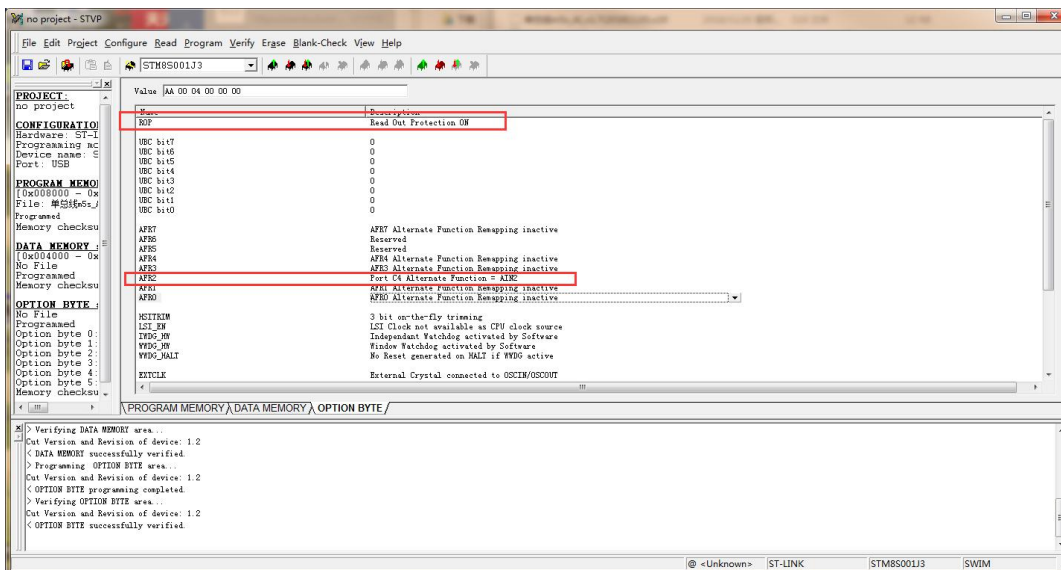
打开烧写软件，ST Visual Programmer (刚刚安装完成的软件)，选择 STMS001J3 芯片。

如下图。

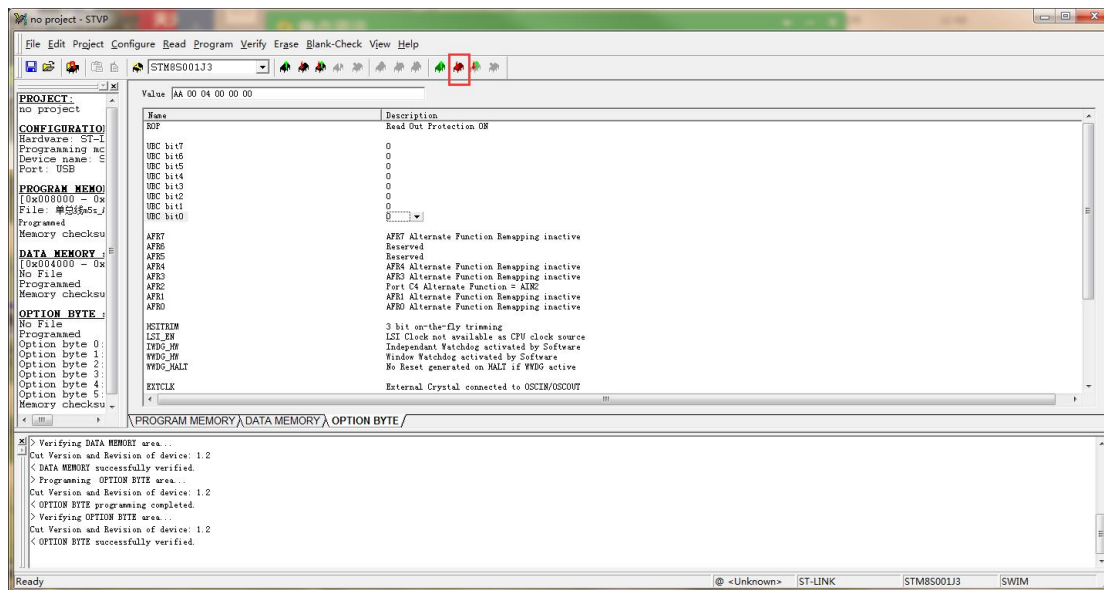


然后 File-open 选择 M5Sv1.7.s19 文件，导入固件。

然后设置 ROP 为 Read Out Protection ON，再设置 AFR2 为 Port C4 Alternate Function = AIN2。如下图。



然后按 Program-All tabs 下载固件，完成更新。如下图。



下载时会弹出如下图的对话框，按是即可。

