多作物联合收获机智能化监控系统

**V1.0**

**源代码**

一、main.c中

int main() {

////////////////////System Init//////////////////////////////

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //设置系统中断优先级分组2

delay\_init(168); //初始化延时函数

uart\_init(115200); //初始化串口 波特率为115200

USART2\_Init(115200); //初始化usart串口2

USART3\_Init(115200); //初始化usart串口3

BEEP\_Init();//初始化蜂鸣器

My\_RTC\_Init();

RTC\_Set\_WakeUp(RTC\_WakeUpClock\_CK\_SPRE\_16bits,0); //初始化RTC

CAN1\_Mode\_Init(CAN\_SJW\_1tq,CAN\_BS2\_6tq,CAN\_BS1\_7tq,6,CAN\_Mode\_Normal);

TIM3\_Int\_Init(999,71); //初始化定时器1ms

LCD\_Init();

BUTTON\_Init();

////////////////////can Init//////////////////////////////

Init\_3402(ADDRESS\_3402);

enable\_TPDO(ADDRESS\_3402, \_TPDO1, TPDOCOMMAND\_NO);

enable\_TPDO(ADDRESS\_3402, \_TPDO2, TPDOCOMMAND\_NO);delay\_ms(100);

enable\_TPDO(ADDRESS\_3402, \_TPDO3, TPDOCOMMAND\_NO);

enable\_TPDO(ADDRESS\_3402, \_TPDO4, TPDOCOMMAND\_NO);delay\_ms(100);

enable\_TPDO(ADDRESS\_5801\_1, \_TPDO1, TPDOCOMMAND\_YES);

enable\_TPDO(ADDRESS\_5801\_1, \_TPDO2, TPDOCOMMAND\_YES);delay\_ms(100);

enable\_TPDO(ADDRESS\_5801\_1, \_TPDO3, TPDOCOMMAND\_YES);

enable\_TPDO(ADDRESS\_5801\_1, \_TPDO4, TPDOCOMMAND\_YES);delay\_ms(100);

enable\_TPDO(ADDRESS\_5801\_2, \_TPDO1, TPDOCOMMAND\_YES);

enable\_TPDO(ADDRESS\_5801\_2, \_TPDO2, TPDOCOMMAND\_YES);delay\_ms(100);

enable\_TPDO(ADDRESS\_5801\_2, \_TPDO3, TPDOCOMMAND\_YES);

enable\_TPDO(ADDRESS\_5801\_2, \_TPDO4, TPDOCOMMAND\_YES);delay\_ms(100);

Init\_4055(id);

Init\_5801(ADDRESS\_5801\_1, 0xff);delay\_ms(100);

Init\_5801(ADDRESS\_5801\_2, 0x07);delay\_ms(100);

enable\_TPDO(ADDRESS\_5801\_2, \_TPDO2, TPDOCOMMAND\_YES);delay\_ms(100);

Init\_7017(ADDRESS\_7017);delay\_ms(100);

enable\_TPDO(ADDRESS\_7017, \_TPDO1, TPDOCOMMAND\_YES);delay\_ms(100);

enable\_TPDO(ADDRESS\_7017, \_TPDO2, TPDOCOMMAND\_YES);delay\_ms(100);

////////////////5801 清零////////////////////////

u8 clear\_temp[8] = {0, 0, 0, 0, 1, 1, 1, 1};

Set\_5801\_ClearCount(ADDRESS\_5801\_2, clear\_temp);//清零计数通道

////////////////////SD卡 FATFS Init////////////////////////

usmart\_dev.init(84); //初始化USMART

W25QXX\_Init(); //初始化W25Q128

my\_mem\_init(SRAMIN); //初始化内部内存池

my\_mem\_init(SRAMCCM); //初始化CCM内存池

if(SD\_Init()) //初始化SD卡

;//SD Card Error!

else

{

exfuns\_init();//为fatfs相关变量申请内存

f\_mount(fs[0],"0:",1);//挂载SD卡

if(exf\_getfree("0",NULL,NULL))

my\_comscreen.sd\_ready\_flag = 0;//SD Card Fatfs Error!

else

my\_comscreen.sd\_ready\_flag = 1;//SD Card Ready!

}

////////////////串口屏初始化//////////////////////////////

systicket\_init();//配置时钟节拍

queue\_reset();//清空串口接收缓冲区

delay\_ms(300);//延时等待串口屏初始化完毕,必须等待300ms

/////////////////LCD初始化//////////////////////////////

POINT\_COLOR=RED;//设置字体为红色

LCD\_ShowString(20,20,200,16,16,"can\_send\_msg");

POINT\_COLOR=BLUE;//设置字体为蓝色

qsize size = 0;

u8 canbuf[8];

u8 key;

my\_comscreen.sys\_init\_ready=1;

while(1)

{

qsize size = 0;

USART3\_Receive\_Data(my\_comscreen\_value.cmd\_buffer,&size);

if(size>0)

screen\_ID\_case();

if(my\_comscreen.scr\_can\_but\_flag == 1)

rec\_can\_value(my\_comscreen.scr\_5801\_result\_int, my\_comscreen.scr\_7017\_result\_int);

// BUTTON\_C1 = 1;

set(BUTTON\_B5,BUTTON\_C1);

set(BUTTON\_B6,BUTTON\_C2);

set(BUTTON\_B7,BUTTON\_C3);

set(BUTTON\_B8,BUTTON\_C4);

set(BUTTON\_B12,BUTTON\_C5);

set(BUTTON\_B13,BUTTON\_C6);

}

}

二、timer.c中

void TIM3\_IRQHandler(void)

{

if(TIM\_GetITStatus(TIM3,TIM\_IT\_Update)==SET) //溢出中断

{

static int timer\_count = 0;

timer\_count++;

//sd卡检测

if(my\_comscreen.sys\_init\_ready==1 && timer\_count%1000==0)

{

if(SD\_GetState() == SD\_CARD\_ERROR)

{

my\_comscreen.sd\_ready\_flag = 0;

const u8 LEN = 64;

u8comscreen\_send\_msg[LEN] = {0xEE,0xB1,0x10,0x00,0x00,

0x00,0x06,0x53,0x44,

0xBF,0xA8,0xB4,0xED,0xCE,

0xF3,0xFF,0xFC,0xFF,0xFF };

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

}

else

{

my\_comscreen.sd\_ready\_flag = 1;

const u8 LEN = 64;

u8 comscreen\_send\_msg[LEN] = {0xEE,0xB1,0x10,0x00,0x00,0x00,0x

06,0x53,0x44,0xBF,0xA8,0xD5,0xFD,0xB3

,0xA3,0xFF,0xFC,0xFF,0xFF };

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

}

}

//can接收

if(timer\_count%300==0 && my\_comscreen.scr\_can\_but\_flag == 1)

{

//5801

static u8 countconscreen = 0;

// rec\_can\_value(my\_comscreen.scr\_5801\_result\_int, my\_comscreen.scr\_7017\_result\_int);

sprintf(my\_comscreen.scr\_5801\_result\_char[0], "%.2f", my\_comscreen.scr\_5801\_result\_int[0]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[1], "%.2f", my\_comscreen.scr\_5801\_result\_int[1]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[2], "%.2f", my\_comscreen.scr\_5801\_result\_int[2]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[3], "%.2f", my\_comscreen.scr\_5801\_result\_int[3]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[4], "%.2f", my\_comscreen.scr\_5801\_result\_int[4]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[5], "%.2f", my\_comscreen.scr\_5801\_result\_int[5]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[6], "%.2f", my\_comscreen.scr\_5801\_result\_int[6]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[7], "%.2f", my\_comscreen.scr\_5801\_result\_int[7]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[8], "%.2f", my\_comscreen.scr\_5801\_result\_int[8]\*0.6);

sprintf(my\_comscreen.scr\_5801\_result\_char[9], "%.2f", my\_comscreen.scr\_5801\_result\_int[9]\*0.6);

u32 temp = my\_comscreen.scr\_5801\_result\_int[11] + my\_comscreen.scr\_5801\_result\_int[12] + my\_comscreen.scr\_5801\_result\_int[13] + my\_comscreen.scr\_5801\_result\_int[15];

// sprintf(my\_comscreen.scr\_5801\_result\_char[10], "%d", my\_comscreen.scr\_5801\_result\_int[10]);

// sprintf(my\_comscreen.scr\_5801\_result\_char[11], "%d", my\_comscreen.scr\_5801\_result\_int[11]);

// sprintf(my\_comscreen.scr\_5801\_result\_char[12], "%d", my\_comscreen.scr\_5801\_result\_int[12]);

// sprintf(my\_comscreen.scr\_5801\_result\_char[13], "%d", my\_comscreen.scr\_5801\_result\_int[13]);

sprintf(my\_comscreen.scr\_5801\_result\_char[14], "%d", my\_comscreen.scr\_5801\_result\_int[14]);

sprintf(my\_comscreen.scr\_5801\_result\_char[15], "%d", temp);

const u8 LEN = 64;

u8 comscreen\_send\_msg[LEN];

u8 need[5];

need[0]=0;need[1]=1;need[2]=2;need[3]=3;need[4]=4;

getsendmsg\_5801(comscreen\_send\_msg, 5, 1, need);

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

need[0]=5;need[1]=6;need[2]=7;need[3]=8;need[4]=9;

getsendmsg\_5801(comscreen\_send\_msg, 5, 2, need);

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

need[0]=14;need[1]=15;

getsendmsg\_5801(comscreen\_send\_msg, 2, 3, need);

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

//7017

sprintf(my\_comscreen.scr\_7017\_result\_char[0], "%d", my\_comscreen.scr\_7017\_result\_int[0]);

sprintf(my\_comscreen.scr\_7017\_result\_char[1], "%d", my\_comscreen.scr\_7017\_result\_int[1]);

sprintf(my\_comscreen.scr\_7017\_result\_char[2], "%d", my\_comscreen.scr\_7017\_result\_int[2]);

sprintf(my\_comscreen.scr\_7017\_result\_char[3], "%d", my\_comscreen.scr\_7017\_result\_int[3]);

sprintf(my\_comscreen.scr\_7017\_result\_char[4], "%d", my\_comscreen.scr\_7017\_result\_int[4]);

sprintf(my\_comscreen.scr\_7017\_result\_char[5], "%d", my\_comscreen.scr\_7017\_result\_int[5]);

sprintf(my\_comscreen.scr\_7017\_result\_char[6], "%d", my\_comscreen.scr\_7017\_result\_int[6]);

// sprintf(my\_comscreen.scr\_7017\_result\_char[7], "%d", my\_comscreen.scr\_7017\_result\_int[7]);

need[0]=0;need[1]=1;need[2]=2;need[3]=3;need[4]=4;

getsendmsg\_7017(comscreen\_send\_msg, 5, 4, need);

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

need[0]=5;need[1]=6;

getsendmsg\_7017(comscreen\_send\_msg, 2, 8, need);

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

}

//sd卡写入

if(timer\_count%200==0 && my\_comscreen.sd\_ready\_flag == 1 && my\_comscreen.scr\_sd\_but\_flag == 1)

{

LCD\_ShowString(200,200,200,16,16,"sdon");

{

FIL fil;

UINT bww;

u8 res;

char temp[200];

RTC\_TimeTypeDef RTC\_TimeStruct;

RTC\_DateTypeDef RTC\_DateStruct;

RTC\_GetTime(RTC\_Format\_BIN,&RTC\_TimeStruct);

RTC\_GetDate(RTC\_Format\_BIN,&RTC\_DateStruct);

sprintf(temp, "0:/5801\_20%d-%02d-%02d %02dh.xls", RTC\_DateStruct.RTC\_Year, RTC\_DateStruct.RTC\_Month, RTC\_DateStruct.RTC\_Date, RTC\_TimeStruct.RTC\_Hours);

res=f\_open(&fil,temp,FA\_CREATE\_NEW | FA\_WRITE);

if(res)

{

f\_open(&fil,temp,FA\_OPEN\_EXISTING | FA\_WRITE);

res=f\_lseek(&fil,f\_size(&fil));

sprintf(temp, "20%d-%02d-%02d\t%02d:%02d:%02d\t", RTC\_DateStruct.RTC\_Year, RTC\_DateStruct.RTC\_Month, RTC\_DateStruct.RTC\_Date,RTC\_TimeStruct.RTC\_Hours,RTC\_TimeStruct.RTC\_Minutes,RTC\_TimeStruct.RTC\_Seconds);

for(int i=0; i<10; i++)

{

char tempp[20];

sprintf(tempp, "%s\t", my\_comscreen.scr\_5801\_result\_char[i]);

strcat(temp,tempp);

}

{

char tempp[20];

sprintf(tempp, "%s\t", my\_comscreen.scr\_5801\_result\_char[15]);

strcat(temp,tempp);

sprintf(tempp, "%s\t", my\_comscreen.scr\_5801\_result\_char[14]);

strcat(temp,tempp);

}

strcat(temp,"\n");

f\_write(&fil,temp,strlen(temp), &bww);

f\_close(&fil);

}

else

{

sprintf(temp, "日期\t时间\t前进速度\t拨禾轮转速\t输送槽转速\t割台搅龙转速\t滚筒转速\t风机转速\t杂余搅龙转速\t籽粒搅龙转速\t碎草装置转速\t发动机转速\t籽粒夹带损失\t籽粒清选损失\n");

f\_write(&fil,temp,strlen(temp), &bww);

f\_close(&fil);

}

}

{

FIL fil;

UINT bww;

u8 res;

char temp[200];

RTC\_TimeTypeDef RTC\_TimeStruct;

RTC\_DateTypeDef RTC\_DateStruct;

RTC\_GetTime(RTC\_Format\_BIN,&RTC\_TimeStruct);

RTC\_GetDate(RTC\_Format\_BIN,&RTC\_DateStruct);

sprintf(temp, "0:/7017\_20%d-%02d-%02d %02dh.xls", RTC\_DateStruct.RTC\_Year, RTC\_DateStruct.RTC\_Month, RTC\_DateStruct.RTC\_Date, RTC\_TimeStruct.RTC\_Hours);

res=f\_open(&fil,temp,FA\_CREATE\_NEW | FA\_WRITE);

if(res)

{

f\_open(&fil,temp,FA\_OPEN\_EXISTING | FA\_WRITE);

res=f\_lseek(&fil,f\_size(&fil));

sprintf(temp, "20%d-%02d-%02d\t%02d:%02d:%02d\t", RTC\_DateStruct.RTC\_Year, RTC\_DateStruct.RTC\_Month, RTC\_DateStruct.RTC\_Date,RTC\_TimeStruct.RTC\_Hours,RTC\_TimeStruct.RTC\_Minutes,RTC\_TimeStruct.RTC\_Seconds);

for(int i=0; i<7; i++)

{

char tempp[20];

sprintf(tempp, "%s\t", my\_comscreen.scr\_7017\_result\_char[i]);

strcat(temp,tempp);

}

strcat(temp,"\n");

f\_write(&fil,temp,strlen(temp), &bww);

f\_close(&fil);

}

else

{

sprintf(temp, "日期\t时间\t拨禾轮高度\t割台高度\t鱼鳞筛开度\t仿形板1角度\t仿形板2角度\t超声波1\t超声波2\n");

f\_write(&fil,temp,strlen(temp), &bww);

f\_close(&fil);

}

}

}

//电机

if(my\_comscreen.scr\_4055\_motor1\_but\_ena\_flag == 1 && timer\_count % (11-my\_comscreen.scr\_4055\_motor1\_slider\_value) == 0 )

{

static u8 flag = 0;

if(flag == 0)

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg & 0xFB;//11111011

else if(flag == 1)

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg | 0x04;//00000100

Set\_4055\_Switchoutput(ADDRESS\_4055, my\_comscreen.scr\_4055\_motor\_can\_send\_msg);

flag = !flag;

}

if(my\_comscreen.scr\_4055\_motor2\_but\_ena\_flag == 1 && timer\_count % (11-my\_comscreen.scr\_4055\_motor2\_slider\_value) == 0 )

{

static u8 flag = 0;

if(flag == 0)

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg & 0x7F;//01111111

else if(flag == 1)

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg | 0x80;//10000000

Set\_4055\_Switchoutput(ADDRESS\_4055, my\_comscreen.scr\_4055\_motor\_can\_send\_msg);

flag = !flag;

}

//溢出清零

if(timer\_count == 10000) timer\_count = 0;

}

TIM\_ClearITPendingBit(TIM3,TIM\_IT\_Update); //清除中断标志位

}

三、canmodule.c中

void Init\_3402(uint32\_t id)

{

Set\_3402\_AOoutput\_type(id, Output\_pin\_AO0, type\_V);delay\_ms(100);

Set\_3402\_AOoutput\_type(id, Output\_pin\_AO1, type\_V);delay\_ms(100);

// enable\_TPDO(id, \_TPDO1, TPDOCOMMAND\_NO);delay\_ms(100);

// enable\_TPDO(id, \_TPDO2, TPDOCOMMAND\_NO);delay\_ms(100);

// enable\_TPDO(id, \_TPDO3, TPDOCOMMAND\_NO);delay\_ms(100);

// enable\_TPDO(id, \_TPDO4, TPDOCOMMAND\_NO);delay\_ms(100);

}

u8 Set\_3402\_Analogoutput(uint32\_t id, u16 ao\_0, u16 ao\_1)

{

u8 mbox;

u8 ao\_00, ao\_01, ao\_10, ao\_11;

ao\_00 = ao\_0 & 0x00ff; //低八位

ao\_01 = ao\_0 >> 8; //高八位

ao\_10 = ao\_1 & 0x00ff; //低八位

ao\_11 = ao\_1 >> 8; //高八位

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x200+id; // 标准标识符为0

TxMessage.ExtId=0x200+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=5;

TxMessage.Data[0]=0x03;

TxMessage.Data[1]=ao\_00;

TxMessage.Data[2]=ao\_01;

TxMessage.Data[3]=ao\_10;

TxMessage.Data[4]=ao\_11;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_3402\_AOoutput\_type(uint32\_t id, Output\_pin pin, Output\_type type)

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2F;

TxMessage.Data[1]=0x09;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=pin;

TxMessage.Data[4]=type;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

void Init\_4055(uint32\_t id)

{

Set\_4055\_Safety\_output(id, 0xFF);delay\_ms(100);

enable\_TPDO(id, \_TPDO1, TPDOCOMMAND\_NO);delay\_ms(100);

enable\_TPDO(id, \_TPDO2, TPDOCOMMAND\_NO);delay\_ms(100);

enable\_TPDO(id, \_TPDO3, TPDOCOMMAND\_NO);delay\_ms(100);

enable\_TPDO(id, \_TPDO4, TPDOCOMMAND\_NO);delay\_ms(100);

}

u8 Set\_4055\_Switchoutput(uint32\_t id, u8 digital\_output)

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x200+id; // 标准标识符为0

TxMessage.ExtId=0x200+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=1;

TxMessage.Data[0] = digital\_output;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_4055\_Safety\_output(uint32\_t id, u8 type\_DO)

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2F;

TxMessage.Data[1]=0x01;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=0x01;

TxMessage.Data[4]=type\_DO;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

void Init\_5801(u8 address, u8 value)

{

Set\_5801\_Frequency\_Type(address, value);delay\_ms(100);//0xFF所有通道设为测频率模式

Set\_5801\_Filter\_Time(address, Passageway\_CF0, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Filter\_Time(address, Passageway\_CF1, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Filter\_Time(address, Passageway\_CF2, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Filter\_Time(address, Passageway\_CF3, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Filter\_Time(address, Passageway\_CF4, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Filter\_Time(address, Passageway\_CF5, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Filter\_Time(address, Passageway\_CF6, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Filter\_Time(address, Passageway\_CF7, Filtertime\_10);delay\_ms(100);//设置通道0 1滤波时间

Set\_5801\_Excessive\_Time(address, Passageway\_CF0, 0x32);delay\_ms(100);

Set\_5801\_Excessive\_Time(address, Passageway\_CF1, 0x32);delay\_ms(100);

Set\_5801\_Excessive\_Time(address, Passageway\_CF2, 0x32);delay\_ms(100);

Set\_5801\_Excessive\_Time(address, Passageway\_CF3, 0x32);delay\_ms(100);

Set\_5801\_Excessive\_Time(address, Passageway\_CF4, 0x32);delay\_ms(100);

Set\_5801\_Excessive\_Time(address, Passageway\_CF5, 0x32);delay\_ms(100);

Set\_5801\_Excessive\_Time(address, Passageway\_CF6, 0x32);delay\_ms(100);

Set\_5801\_Excessive\_Time(address, Passageway\_CF7, 0x32);delay\_ms(100);

lapsetime\_TPDO(address, \_TPDO1, 10);delay\_ms(100);

lapsetime\_TPDO(address, \_TPDO2, 10);delay\_ms(100);

lapsetime\_TPDO(address, \_TPDO3, 10);delay\_ms(100);

lapsetime\_TPDO(address, \_TPDO4, 10);delay\_ms(100);

event\_TPDO(address, \_TPDO1, 50);delay\_ms(100);

event\_TPDO(address, \_TPDO2, 50);delay\_ms(100);

event\_TPDO(address, \_TPDO3, 50);delay\_ms(100);//100

event\_TPDO(address, \_TPDO4, 50);delay\_ms(100);

}

u8 rec\_5801\_ImpulseValue(uint32\_t ID, u32 \*resut\_int)//读取频率值

{

u8 buf[8];

CanRxMsg RxMessage;

if( CAN\_MessagePending(CAN1,CAN\_FIFO0)==0) return 0; //没有接收到数据,直接退出

CAN\_Receive(CAN1, CAN\_FIFO0, &RxMessage);//读取数据

for(u32 i=0;i<RxMessage.DLC;i++)

buf[i]=RxMessage.Data[i];

uint32\_t id = RxMessage.StdId;

static int count = 0;

count+=50;

LCD\_ShowxNum(count,70,id,4,16,0);//delay\_ms(1000);

if(count > 800)count = 0;

if(id == 0x180+ID)

{

resut\_int[0] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道0

resut\_int[1] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道1

static int i = 0;i++;

LCD\_ShowxNum(10,100,i,4,16,0);//delay\_ms(1000);

}

else if(id == 0x280+ID)

{

resut\_int[2] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道2

resut\_int[3] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道3

static int i = 0;i++;

LCD\_ShowxNum(10,120,i,4,16,0);//delay\_ms(1000);

}

else if(id == 0x380+ID)

{

resut\_int[4] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道4

resut\_int[5] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道5

static int i = 0;i++;

LCD\_ShowxNum(10,140,i,4,16,0);//delay\_ms(1000);

}

else if(id == 0x480+ID)

{

resut\_int[6] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道6

resut\_int[7] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道7

static int i = 0;i++;

LCD\_ShowxNum(10,160,i,4,16,0);//delay\_ms(1000);

}

return RxMessage.DLC;

}

u8 rec\_5801\_CountValue(uint32\_t ID, u32 resut\_int[8])//读取计数值

{

u8 buf[8];

CanRxMsg RxMessage;

if( CAN\_MessagePending(CAN1,CAN\_FIFO0)==0) return 0; //没有接收到数据,直接退出

CAN\_Receive(CAN1, CAN\_FIFO0, &RxMessage);//读取数据

for(u32 i=0;i<RxMessage.DLC;i++)

buf[i]=RxMessage.Data[i];

uint32\_t id = RxMessage.StdId;

if(id == 0x180+ID)

{

resut\_int[0] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道0

resut\_int[1] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道1

}

else if(id == 0x280+ID)

{

resut\_int[2] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道2

resut\_int[3] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道3

}

else if(id == 0x380+ID)

{

resut\_int[4] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道4

resut\_int[5] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道5

}

else if(id == 0x480+ID)

{

resut\_int[6] = buf[0]+buf[1]\*256+buf[2]\*65536+buf[3]\*16777216;//通道6

resut\_int[7] = buf[4]+buf[5]\*256+buf[6]\*65536+buf[7]\*16777216;//通道7

}

return RxMessage.DLC;

}

u8 Set\_5801\_Frequency\_Type(uint32\_t id, uint8\_t type)//设置模块为测频或计数模式，0为计数模式，1为测频模式

{ //由type的值确定各个通道的具体模式

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2B;

TxMessage.Data[1]=0x00;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=0x01;

TxMessage.Data[4]=type;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_5801\_ClearCount(uint32\_t id, u8 count\_clear[8])//清零计数通道

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x200+id; // 标准标识符为0

TxMessage.ExtId=0x200+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=count\_clear[0];//0x01//清零

TxMessage.Data[1]=count\_clear[1];//0x00//不清零

TxMessage.Data[2]=count\_clear[2];//0x01//清零

TxMessage.Data[3]=count\_clear[3];//0x01//清零

TxMessage.Data[4]=count\_clear[4];//0x00//不清零

TxMessage.Data[5]=count\_clear[5];//0x01//清零

TxMessage.Data[6]=count\_clear[6];//0x01//清零

TxMessage.Data[7]=count\_clear[7];//0x00//不清零

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_5801\_Filter\_Time(uint32\_t id, EMPPassageway Passageway, EMPFiltertime Filtertime)//设置模块测频通道滤波时间

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x580+id; // 标准标识符为0

TxMessage.ExtId=0x580+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2B;

TxMessage.Data[1]=0x07;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=Passageway;//具体配置哪个通道

TxMessage.Data[4]=Filtertime;//滤波时间

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_5801\_Excessive\_Time(uint32\_t id, EMPPassageway Passageway, uint8\_t Excessivetime)//设置模块测频超时时间（设置为最小频率的周期）

{ //0x32=50，50\*100ms=5s，定时5s

u8 mbox; //0x0A=10，10\*100ms=5s，定时1s

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2B;

TxMessage.Data[1]=0x08;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=Passageway;

TxMessage.Data[4]=Excessivetime;//设置的测频更新时间（单位 100ms）所以更新时间是 Excessivetime\*100ms

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

void Init\_7017(uint32\_t ID)

{

Set\_7017\_AiType(ID, AiType\_5V);delay\_ms(100);

Set\_7017\_AiEnable(ID, AiChannel\_ALL);delay\_ms(100);

Set\_7017\_AiDataType(ID, ANALOG\_VALUE);delay\_ms(100);

Set\_7017\_AiNumberOfFilters(ID, AiNumOfFilters(5));delay\_ms(100);

lapsetime\_TPDO(ID, \_TPDO1, 999);delay\_ms(100);

lapsetime\_TPDO(ID, \_TPDO2, 999);delay\_ms(100);

event\_TPDO(ID, \_TPDO1, 100);delay\_ms(100);

event\_TPDO(ID, \_TPDO2, 100);delay\_ms(100);

}

u8 rec\_7017\_AiValue(uint32\_t ID, u16 resut\_char[8])//读取 AI 模拟量输入

{

u8 buf[8];

CanRxMsg RxMessage;

if( CAN\_MessagePending(CAN1,CAN\_FIFO0)==0) return 0; //没有接收到数据,直接退出

CAN\_Receive(CAN1, CAN\_FIFO0, &RxMessage);//读取数据

for(u32 i=0;i<RxMessage.DLC;i++)

buf[i]=RxMessage.Data[i];

uint32\_t id = RxMessage.StdId;

if(id == 0x180+ID)

{

resut\_char[0] = (u16)(buf[0]+buf[1]\*256);

resut\_char[1] = (u16)(buf[2]+buf[3]\*256);

resut\_char[2] = (u16)(buf[4]+buf[5]\*256);

resut\_char[3] = (u16)(buf[6]+buf[7]\*256);

static int m = 0;m++;

LCD\_ShowxNum(70,100,m,4,16,0);//delay\_ms(1000);

}

else if(id == 0x280+ID)

{

resut\_char[4] = (u16)(buf[0]+buf[1]\*256);

resut\_char[5] = (u16)(buf[2]+buf[3]\*256);

resut\_char[6] = (u16)(buf[4]+buf[5]\*256);

resut\_char[7] = (u16)(buf[6]+buf[7]\*256);

static int i = 0;i++;

LCD\_ShowxNum(70,120,i,4,16,0);//delay\_ms(1000);

}

return RxMessage.DLC;

}

u8 set\_7017\_DOBit(uint32\_t id, u8 a)//设置 DO 开关量输出

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x200+id; // 标准标识符为0

TxMessage.ExtId=0x200+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=1;

if(a == 0)

TxMessage.Data[0]=0x00;

else if(a == 1)

TxMessage.Data[0]=0x01;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 rec\_7017\_msg(uint32\_t \*id, u8 \*buf)

{

CanRxMsg RxMessage;

if( CAN\_MessagePending(CAN1,CAN\_FIFO0)==0) return 0; //没有接收到数据,直接退出

CAN\_Receive(CAN1, CAN\_FIFO0, &RxMessage);//读取数据

for(u32 i=0;i<RxMessage.DLC;i++)

buf[i]=RxMessage.Data[i];

\*id = RxMessage.StdId;

return RxMessage.DLC;

}

u8 send\_7017\_msg(uint32\_t id)

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x40;

TxMessage.Data[1]=0x09;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=0x01;

TxMessage.Data[4]=0x00;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_7017\_AiType(uint32\_t id, AiType type)//设置模块的 AIN 采集类型

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2F;

TxMessage.Data[1]=0x04;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=0x01;

TxMessage.Data[4]=type;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_7017\_AiEnable(uint32\_t id, AiChannel channel)//设置模块的 AI 采集通道使能

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2F;

TxMessage.Data[1]=0x03;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=0x01;

TxMessage.Data[4]=channel;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_7017\_AiDataType(uint32\_t id, AiDataType datatype)//设置模块 AI 的数据格式

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2F;

TxMessage.Data[1]=0x08;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=0x01;

TxMessage.Data[4]=datatype;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

u8 Set\_7017\_AiNumberOfFilters(uint32\_t id, AiNum Ainum)//设置模块的采集滤波次数

{

u8 mbox;

u16 i=0;

CanTxMsg TxMessage;

TxMessage.StdId=0x600+id; // 标准标识符为0

TxMessage.ExtId=0x600+id; // 设置扩展标示符（29位）

TxMessage.IDE=0; // 使用扩展标识符

TxMessage.RTR=0; // 消息类型为数据帧，一帧8位

TxMessage.DLC=8;

TxMessage.Data[0]=0x2B;

TxMessage.Data[1]=0x09;

TxMessage.Data[2]=0x20;

TxMessage.Data[3]=0x01;

TxMessage.Data[4]=Ainum;

TxMessage.Data[5]=0x00;

TxMessage.Data[6]=0x00;

TxMessage.Data[7]=0x00;

mbox= CAN\_Transmit(CAN1, &TxMessage);

while((CAN\_TransmitStatus(CAN1, mbox)==CAN\_TxStatus\_Failed)&&(i<0XFFF))i++; //等待发送结束

if(i>=0XFFF)return 1;

return 0;

}

四、comscreen.c中

void screen\_ID\_case(void)

{

uint16 screen\_id = PTR2U16(&((PCTRL\_MSG)my\_comscreen\_value.cmd\_buffer)->screen\_id);

switch(screen\_id)

{

case SCR\_MAIN\_ID:

break;

case SCR\_5801\_1\_ID:

control\_ID\_case();

break;

case SCR\_5801\_2\_ID:

control\_ID\_case();

break;

case SCR\_7017\_1\_ID:

control\_ID\_case();

break;

case SCR\_7017\_2\_ID:

control\_ID\_case();

break;

case SCR\_4055\_ID:

control\_ID\_case();

break;

case SCR\_3402\_ID:

control\_ID\_case();

break;

case SCR\_SET\_ID:

control\_ID\_case();

break;

default:

break;

}

}

void control\_ID\_case(void)

{

uint16 control\_id = PTR2U16(&((PCTRL\_MSG)my\_comscreen\_value.cmd\_buffer)->control\_id);

u8 button\_state = ((PCTRL\_MSG)my\_comscreen\_value.cmd\_buffer)->param[1];

uint32 slider\_4055\_value = PTR2U32(((PCTRL\_MSG)my\_comscreen\_value.cmd\_buffer)->param);

static uint32 slider1\_value = 0;

static uint32 slider2\_value = 0;

switch(control\_id)

{

case SCR\_5801\_1\_CAN\_BUT\_ID:

my\_comscreen.scr\_can\_but\_flag = button\_state;

break;

case SCR\_5801\_1\_SD\_BUT\_ID:

my\_comscreen.scr\_sd\_but\_flag = button\_state;

break;

case SCR\_4055\_MOTOR1\_BUT\_ENA\_ID:

if(button\_state == 0)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg & 0xFE;//11111110

my\_comscreen.scr\_4055\_motor1\_but\_ena\_flag = 0;

}

else if(button\_state == 1)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg | 0x01;//00000001

my\_comscreen.scr\_4055\_motor1\_but\_ena\_flag = 1;

}

Set\_4055\_Switchoutput(ADDRESS\_4055, my\_comscreen.scr\_4055\_motor\_can\_send\_msg);

break;

case SCR\_4055\_MOTOR1\_BUT\_DIR\_ID:

if(button\_state == 0)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg & 0xFD;//11111110

my\_comscreen.scr\_4055\_motor1\_but\_dir\_flag = 0;

}

else if(button\_state == 1)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg | 0x02;//00000001

my\_comscreen.scr\_4055\_motor1\_but\_dir\_flag = 1;

}

Set\_4055\_Switchoutput(ADDRESS\_4055, my\_comscreen.scr\_4055\_motor\_can\_send\_msg);

break;

case SCR\_4055\_MOTOR1\_SLIDER\_ID:

my\_comscreen.scr\_4055\_motor1\_slider\_value = slider\_4055\_value;

break;

case SCR\_4055\_MOTOR2\_BUT\_ENA\_ID:

if(button\_state == 0)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg & 0xDF;//11111110

my\_comscreen.scr\_4055\_motor2\_but\_ena\_flag = 0;

}

else if(button\_state == 1)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg | 0x20;//00000001

my\_comscreen.scr\_4055\_motor2\_but\_ena\_flag = 1;

}

Set\_4055\_Switchoutput(ADDRESS\_4055, my\_comscreen.scr\_4055\_motor\_can\_send\_msg);

break;

case SCR\_4055\_MOTOR2\_BUT\_DIR\_ID:

if(button\_state == 0)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg & 0xBF;//11111110

my\_comscreen.scr\_4055\_motor3\_but\_dir\_flag = 0;

}

else if(button\_state == 1)

{

my\_comscreen.scr\_4055\_motor\_can\_send\_msg = my\_comscreen.scr\_4055\_motor\_can\_send\_msg | 0x40;//00000001

my\_comscreen.scr\_4055\_motor3\_but\_dir\_flag = 1;

}

Set\_4055\_Switchoutput(ADDRESS\_4055, my\_comscreen.scr\_4055\_motor\_can\_send\_msg);

break;

case SCR\_4055\_MOTOR2\_SLIDER\_ID:

my\_comscreen.scr\_4055\_motor2\_slider\_value = slider\_4055\_value;

break;

case SCR\_3402\_SLIDER1\_ID:

slider1\_value = PTR2U32(((PCTRL\_MSG)my\_comscreen\_value.cmd\_buffer)->param);

Set\_3402\_Analogoutput(ADDRESS\_3402, slider1\_value, slider2\_value);

break;

case SCR\_3402\_SLIDER2\_ID:

slider2\_value = PTR2U32(((PCTRL\_MSG)my\_comscreen\_value.cmd\_buffer)->param);

Set\_3402\_Analogoutput(ADDRESS\_3402, slider1\_value, slider2\_value);

break;

case SCR\_SET\_SD\_BUT\_ID:

{

u32 total,free;

if(SD\_GetState() == SD\_CARD\_ERROR)

{

if(SD\_Init())

{

my\_comscreen.sd\_ready\_flag = 0;

const u8 LEN = 64;

u8 comscreen\_send\_msg[LEN] = {0xEE,0xB1,0x10,0x00,0x07,0x00,0x02,0x53,0x44,0xBF,0xA8,0xB4,0xED,0xCE,0xF3,0xFF,0xFC,0xFF,0xFF };

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

}

else

{

exfuns\_init(); //为fatfs相关变量申请内存

f\_mount(fs[0],"0:",1); //挂载SD卡

if(exf\_getfree("0",&total,&free))//SD Card Fatfs Error!

{

my\_comscreen.sd\_ready\_flag = 0;

const u8 LEN = 64;

u8 comscreen\_send\_msg[LEN] = {0xEE,0xB1,0x10,0x00,0x07,0x00,0x02,0x53,0x44,0xBF,0xA8,0xB4,0xED,0xCE,0xF3,0xFF,0xFC,0xFF,0xFF };

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

}

else

goto A;//SD Card Ready!

}

}

else

{

exf\_getfree("0",&total,&free);

A:

total = total>>10;

free = free>>10;

my\_comscreen.sd\_ready\_flag = 1;

const u8 LEN = 64;

u8 comscreen\_send\_msg[LEN] = {0xEE,0xB1,0x10,0x00,0x07,0x00,0x02,0x53,0x44,0xBF,0xA8,0xD5,0xFD,0xB3,0xA3,0xFF,0xFC,0xFF,0xFF };

my\_USART\_Send\_Data(comscreen\_send\_msg,LEN);

}

break;

}

}

}