48P-Min-W

48P引脚ECU

工作进度

- [x] FREERTOS系统移植

驱动层

- [x] PWM驱动
- [x] ADC+DMA驱动
- [x] 电流采集驱动
- [x] GPIO电平采集
- [ ] 脉冲检测驱动
- [x] CAN驱动
- [ ] 铁电存储器驱动

应用层

裸机层

电气层(Electric_L]

- [ ] 继电器

架构

传感器读取数据->待处理数组

读取待处理数组处理->放入完成数组

读取待处理数组处理->发出数据

Logic_L(逻辑层)

CAN解析 全车逻辑运算

- [x] 整车初始化
    - [ ] ADC_DMA初始化PWM初始化

- [ ] ADC读取
- [ ] CAN数据解析（扩展帧和标准帧）
- [ ] 滤波算法用于动态数据处理
- [ ] 液压数据滤波

Driver_L(驱动层)

1. PMOS_PWM控制引脚设置
2. ADC滤波

DEBUG

用于管理调试,以及打印调试,编写伏特加(上位机名)协议

对象

1. 仪表
2. 手柄
3. 油门
4. 刹车传感器(微动开关)
5. 刹车液压传感器
6. 运动电机
7. 工作电机
8. 电池箱
9. 大灯
10. 喇叭

重要控制

1.调整数据

2.限位

3.校验

控制

数据接收

数据发送

任务	名称	
逻辑处理	Main_Logical_Processing();	
任务发送	Send_Message_StartTask	
CAN消息处理	Analytic_CAN();	

引脚定义

LED灯

序号	别名	引脚网络	作用
0	LED0_PE7	PE7	LED0
1	LED1_PE8	PE8	LED1
2	LED2_PE10	PE10	RUN

PWM_MOS

序号	别名	引脚网络	作用	
0	PMOS0_T10CH1	PB8	大灯	
1	PMOS1_T2CH1	PA15	大臂推	ADC1_IN14
2	PMOS2_T2CH2	PB3	倒车灯	
3	PMOS3_T8CH4	PC9	大臂收	ADC1_IN15
4	PMOS4_T8CH3	PC8	快换推	
5	PMOS5_T8CH2	PC7	快换收	
6	PMOS6_T8CH1	PC6	右转向	
7	PMOS7_T4CH4	PD15	左转向	
8	PMOS8_T4CH3	PD14	液压锁阀	
9	PMOS9_T4CH2	PD13	电机使能	

序号	别名	引脚网络	作用	
10	PMOS10_T4CH1	PD12	喇叭	
11	PMOS11_T12CH2	PB15	12V继电器	
12	PMOS12_T12CH1	PB14	警示灯	
13	PMOS13_T2CH4	PB11	刹车继电器1	
14	PMOS14_T2CH3	PB10	刹车继电器2	
15	PMOS15_T1CH4	PE14	高压-继电器	
16	PMOS16_T1CH3	PE13	预充继电器	
17	PMOS17_T1CH2	PE11	高压+继电器	
18	PMOS18_T1CH1	PE9	翻斗推	ADC1_IN4
19	PMOS19_T14CH1	PA7	翻斗收	ADC1_IN5

ADC

序号	别名	DMA序号	引脚网络		作用
0	AI0_ADC1CH13	3	PC3		油门角度
1	AI1_ADC1CH12	1	PC2		刹车5V
2	AI2_ADC1CH11	2	PC1		旋钮
3	AI3_ADC1CH10	0	PC0		\
4	AI4_ADC1CH2	4	PA2		\
5	AI5_ADC1CH1	5	PA1		\
6	AI6_ADC1CH0	6	PA0		\
7	ADC1_EL_IN4	7	PA4	MOS18	电流采集(ADC信号)
8	ADC1_EL_IN5	8	PA5	MOS19	\
9	ADC1_EL_IN14	9	PC4	MOS1	\
10	ADC1_EL_IN15	10	PC5	MOS3	\
11	AIV12_ADC1CH9	11	PB1		板载供电电源12V

DMA数据传输顺序

ADC_0~ADC_6

48P-W-Min ADC通道分布图

2(4)	1(5)	0(6)		不可测量电压,用于低电压，拉低
13(0)	12(1)	11(2)	10(3)	有电压

CNT脉冲计数

序号	别名	引脚网络	作用
0	SPD0_T13CH1	PA6	
1	SPD1_T3CH1	PB4	
2	SPD1_T3CH2	PB5	
3	SPD3_T3CH3	PB0	

开关量采集

序号	别名	引脚网络	作用
0	DI0_PD11	PD11	
1	DI1_PD10	PD10	
2	DI2_T9CH2	PE6	
3	DI3_T9CH1	PE5	
4	DI4_PD3	PD3	钥匙2档
5	DI5_PD4	PD4	钥匙1档

CAN

CAN1(500K)

序号	别名	引脚网络	作用
0	CAN1_RX	PD0	z
1	CAN1_TX	PD1	

CAN2(250K)

序号	别名	引脚网络	作用
0	CAN2_RX	PB13	
1	CAN2_TX	PB12	

行走控制

寸进
2000z
最大扭矩150

乌龟
4000z
扭矩最大120

兔子
7200z
最大扭矩60

液压控制

300-550控制区间

部件控制协议

手柄协议

序号	名称	键	备注
0	hglsJoy.button1	喇叭	
1	hglsJoy.button2	3RD锁定	
2	hglsJoy.button3	第三波轮液锁定	
3	hglsJoy.button4	空挡	
4	hglsJoy.button5	高低速	
5	hglsJoy.button6	寸进	

序号	名称	键	备注
6	hglsJoyXYZ.Xaxis	前进后退	
	hglsJoyXYZ.Yaxis	TODO	
	hglsJoyXYZ.Zaxis	TODO	

纠正点

1.仪表台线束

警报灯

2.

文件创建

```
#ifndef __SENSOR_ADC_H__
#define __SENSOR_ADC_H__

#endif
```

##

Test测试区

MOS_PWM

规划

```
PMOS18_T1CH1
PMOS17_T1CH2
PMOS16_T1CH3
PMOS15_T1CH4

PMOS1_T2CH1
PMOS2_T2CH2
```

```
PMOS14_T2CH3
PMOS13_T2CH4

PMOS10_T4CH1
PMOS9_T4CH2
PMOS8_T4CH3
PMOS7_T4CH4

PMOS_T8CH1
PMOS5_T8CH2
PMOS4_T8CH3
PMOS3_T8CH4

PMOS0_T10CH1

PMOS12_T12CH1

PMOS11_T12CH2

PMOS19_T14CH1
```

测试函数

```
//PWM配置区域

__HAL_TIM_SetCompare(&htim1, TIM_CHANNEL_1, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim1, TIM_CHANNEL_2, 500);    //通过修改比较值来改变占空比

__HAL_TIM_SetCompare(&htim1, TIM_CHANNEL_3, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim1, TIM_CHANNEL_4, 500);    //通过修改比较值来改变占空比

__HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_2, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_3, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_4, 500);    //通过修改比较值来改变占空比

__HAL_TIM_SetCompare(&htim4, TIM_CHANNEL_1, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim4, TIM_CHANNEL_2, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim4, TIM_CHANNEL_3, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim4, TIM_CHANNEL_4, 500);    //通过修改比较值来改变占空比

__HAL_TIM_SetCompare(&htim8, TIM_CHANNEL_1, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim8, TIM_CHANNEL_2, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim8, TIM_CHANNEL_3, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim8, TIM_CHANNEL_4, 500);    //通过修改比较值来改变占空比

__HAL_TIM_SetCompare(&htim10, TIM_CHANNEL_1, 500);    //通过修改比较值来改变占空比

__HAL_TIM_SetCompare(&htim12, TIM_CHANNEL_1, 500);    //通过修改比较值来改变占空比
__HAL_TIM_SetCompare(&htim12, TIM_CHANNEL_2, 500);    //通过修改比较值来改变占空比

__HAL_TIM_SetCompare(&htim14, TIM_CHANNEL_1, 500);    //通过修改比较值来改变占空比
```

```
HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_1);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_3);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_4);  //开启PWM通道1

HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_1);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_2);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_3);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_4);  //开启PWM通道1

HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_1);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_3);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_4);  //开启PWM通道1

HAL_TIM_PWM_Start(&htim8,TIM_CHANNEL_1);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim8,TIM_CHANNEL_2);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim8,TIM_CHANNEL_3);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim8,TIM_CHANNEL_4);  //开启PWM通道1

HAL_TIM_PWM_Start(&htim10,TIM_CHANNEL_1);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim12,TIM_CHANNEL_1);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim12,TIM_CHANNEL_2);  //开启PWM通道1
HAL_TIM_PWM_Start(&htim14,TIM_CHANNEL_1);  //开启PWM通道1
```