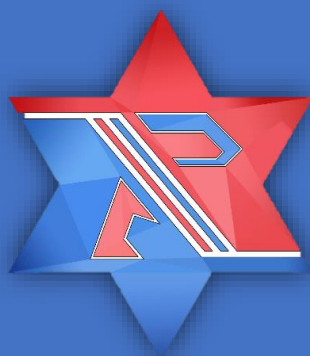


裁判系统使用手册 V22.1.0



电控组 徐翔宇 2021.12.05








目录

一.	移植步骤.....	2
1 .	文件和路径添加.....	2
2 .	头文件修改.....	2
3 .	串口配置.....	2
4 .	DMA 相关配置	3
5 .	函数移植.....	4
二.	移植成功验证:	4
1 .	打印验证.....	4
2 .	客户端 UI 画图验证.....	5
三.	裁判系统数据调用.....	5
1 .	调用接口函数.....	5
2 .	接口函数编写.....	6
四.	问题或其他需求.....	6

一. 移植步骤

1. 文件和路径添加

在 CODING 代码仓库 JUDGE 下载 JUDGE 文件夹，解压后放在自己的工程路径上，将 CRCCheck.c, judge.c, 放在自己的工程分组中，并将头文件添加到 keil 路径中。

	CRCCheck.c	2020/11/28 15:29	c_file	7 KB
	CRCCheck.h	2020/12/5 11:19	H 文件	1 KB
	judge.c	2021/12/4 22:39	c_file	119 KB
	judge.h	2021/12/4 21:11	H 文件	30 KB
	judgeStruct.h	2021/12/4 22:07	H 文件	17 KB
	RoboMaster 2021 裁判系统串口协议附...	2021/3/5 20:07	Microsoft Edge ...	1,489 KB
	裁判系统使用说明.docx	2021/5/3 13:21	Microsoft Word ...	1,692 KB

2. 头文件修改

打开 judgeStruct.h 文件，该文件里面大部分是通信协议中各结构体的定义。请确保开头的包含有 stm32f4xx.h 头文件，然后依据自己的打印串口更改第五行的头文件，以确保裁判系统初始化函数能打印调试信息。

```
1  #ifndef __JUDGESTRUCT_H__
2  #define __JUDGESTRUCT_H__
3
4  #include "stm32f4xx.h"
5  // #include "usart.h"
6  #include "stdio.h"
7  #include "string.h"
```

打开 CRCCheck.h 文件，在第 4 行，如果自己工程里面没有 stm32f4xx.h，把 stm32f4xx.h 改为自己工程里面对应的头文件。

3. 串口配置

根据自己用作裁判系统通信的串口的 GPIO 口是哪个端口，修改第 37 行和 38 行，只需把字母 B 改成其他端口即可。根据自己用作裁判系统的串口引脚修改第 39、40 行 (Tx) 引脚和 41、42 (Rx) 引脚。

```
34  /*****串口和DMA引脚配置表*****/
35
36  //引脚相关，更改RX,TX引脚为裁判系统串口IO口分组和引脚
37  #define Judge_UseGPIO_GPIO_PeriphClock RCC_AHB1Periph_GPIOB
38  #define Judge_UseGPIO_GPIO_Typedef_Name GPIOB
39  #define Judge_UseGPIO_GPIO_PINSource_TX GPIO_PinSource10
40  #define Judge_UseGPIO_GPIO_GPIO_Pin_TX GPIO_Pin_10
41  #define Judge_UseGPIO_GPIO_PINSource_RX GPIO_PinSource11
42  #define Judge_UseGPIO_GPIO_GPIO_Pin_RX GPIO_Pin_11
```

然后将 45-53 行的串口名改为自己使用的串口名。如果是串口 4、5、7、8，将串口名改为 UART+对应数字，如果是串口 1、6、2、3，需将串口名改为 USART+对应数字。

```

44 //串口相关, 更改串口为所使用串口数字
45 #define Judge_UseUART_PeriphClock   RCC_APB1Periph_USART3
46 #define Judge_UseUART_GPIOAF_Name  GPIO_AF_USART3
47 #define Judge_UseUART_Typedef_Name  USART3
48 #define Judge_UseUART_IRQn         USART3_IRQn
49 #define Judge_UseUART_IRQHandler    USART3_IRQHandler
50
51 //函数名相关, 更改数字为当前串口
52 #define Judge_UseUART_Function_Name  USART3_Init
53 #define Judge_UseUART_SendChar       USART3_SendChar

```

除此之外, 如果是使用串口 1, 6, 还需要将第 45 行中的 RCC_APB1PeriphClockCmd 改为 RCC_APB2PeriphClockCmd, 同时找到 judge.c 文件的 1731 行将 APB1 改为 APB2。

```

1730 RCC_AHB1PeriphClockCmd(Judge_UseGPIO_GPIO_PeriphClock, ENABLE);
1731 RCC_APB1PeriphClockCmd(Judge_UseUART_PeriphClock, ENABLE);
1732 RCC_AHB1PeriphClockCmd(Judge_UseDMA_PeriphClock_RX, ENABLE);
1733 RCC_AHB1PeriphClockCmd(Judge_UseDMA_PeriphClock_TX, ENABLE);
1734

```

4. DMA 相关配置

确定裁判通信使用串口后, 通过下图判断自己使用的串口的 DMA, 数据流, 通道数字。

表 35. DMA1 请求映射

外设请求	数据流 0	数据流 1	数据流 2	数据流 3	数据流 4	数据流 5	数据流 6	数据流 7
通道 0	SPI3_RX		SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX		SPI3_TX
通道 1	I2C1_RX		TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
通道 2	TIM4_CH1		I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
通道 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
通道 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
通道 5	UART8_TX ⁽¹⁾	UART7_TX ⁽¹⁾	TIM3_CH4 TIM3_UP	UART7_RX ⁽¹⁾	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX ⁽¹⁾	TIM3_CH3
通道 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2		TIM5_UP	
通道 7		TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

表 36. DMA2 请求映射

外设请求	数据流 0	数据流 1	数据流 2	数据流 3	数据流 4	数据流 5	数据流 6	数据流 7
通道 0	ADC1		TIM8_CH1 TIM8_CH2 TIM8_CH3		ADC1		TIM1_CH1 TIM1_CH2 TIM1_CH3	
通道 1		DCMI	ADC2	ADC2		SPI6_TX ⁽¹⁾	SPI6_RX ⁽¹⁾	DCMI
通道 2	ADC3	ADC3		SPI5_RX ⁽¹⁾	SPI5_TX ⁽¹⁾	CRYP_OUT	CRYP_IN	HASH_IN
通道 3	SPI1_RX		SPI1_RX	SPI1_TX		SPI1_TX		
通道 4	SPI4_RX ⁽¹⁾	SPI4_TX ⁽¹⁾	USART1_RX	SDIO		USART1_RX	SDIO	USART1_TX
通道 5		USART6_RX	USART6_RX	SPI4_RX ⁽¹⁾	SPI4_TX ⁽¹⁾		USART6_TX	USART6_TX
通道 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
通道 7		TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX ⁽¹⁾	SPI5_TX ⁽¹⁾	TIM8_CH4 TIM8_TRIG TIM8_COM

根据程序右方注释修改 Tx 和 Rx 的 DMA 相关配置。

```

57 //DMA配置相关
58 //RX的DMA
59 #define Judge_UseDMA_PeriphClock_RX RCC_AHB1Periph_DMA1 //数字改为当前串口RX的DMA数字
60 #define Judge_UseDMA_IRQChannel_RX DMA1_Stream1_IRQn //数字改为对应DMA和数据流数字
61 #define Judge_UseDMA_Stream_TypeDef_Name_RX DMA1_Stream1 //数字改为数据流数字
62 #define Judge_UseDMA_ClearIT_RX DMA_IT_TCIF1 //数字改为数据流数字
63 #define Judge_UseDMA_Channel_RX DMA_Channel_4 //数字改为通道数字
64 #define Judge_UseDMA_ClearFlag DMA_FLAG_TCIF1 | DMA_FLAG_FE1 | DMA_FLAG_DMEIF1 | DMA_FLAG_TE1 | DMA_FLAG_HTIF1 //数字改为通道数字
65 //函数
66 #define Judge_UseDMA_IRQHandler_RX DMA1_Stream1_IRQHandler //数字改为对应DMA和数据流数字
67 //TX的DMA
68 #define Judge_UseDMA_PeriphClock_TX RCC_AHB1Periph_DMA1 //数字改为当前串口TX的DMA数字
69 #define Judge_UseDMA_IRQChannel_TX DMA1_Stream3_IRQn //数字改为对应DMA和数据流数字
70 #define Judge_UseDMA_Stream_TypeDef_Name_TX DMA1_Stream3 //数字改为对应DMA和数据流数字
71 #define Judge_UseDMA_ClearIT_TX DMA_IT_TCIF3 //数字改为数据流数字
72 #define Judge_UseDMA_Channel_TX DMA_Channel_4 //数字改为通道数字
73 //函数
74 #define Judge_UseDMA_IRQHandler_TX DMA1_Stream3_IRQHandler //数字改为对应DMA和数据流数字

```

5. 函数移植

将 judge.h 文件末尾 636 行的 Judge_Init 函数放到自己兵种的初始化函数中。

```

634 /*****裁判系统初始化函数*****/
635 void Judge_Init(void);

```

根据自己兵种的需求，将下面需要动态更新的函数以 10ms 的周期放在定时器或任务函数中，注意使用多个动态画图函数时，第一个函数必须使用，同时需要输入一些自己兵种如云台 yaw.pitch，摩擦轮，拨轮开关标志，超级电容电压电量等数据，详见注释。

```

639 /*****需要动态更新的UI画图函数*****/
640 /*****请按下面的顺序使用函数，10ms更新一次*****/
641 /*
642 @FunctionName:
643 @Brief:画图时间标志更新
644 @Notice:动态画图必须使用
645 @LatestUpdate:2021.12.04
646 @Author:
647 */
648 void Drawing_StateUpdate(void);
649 /*
650 @FunctionName:
651 @Brief:增益显示
652 @Notice:
653 @LatestUpdate:
654 @Author:
655 */
656 void Draw_Buff(void);

```

例如下图：

```

void UI_Task(void*p_arg) //用于自定义操作界面
{
    uint16_t uitempl = 0;
    portTickType currentTime;
    while(1)
    {
        currentTime = xTaskGetTickCount(); //获取当前系统时间
        uitempl++;
        Drawing_StateUpdate();
        Draw_Buff();
        Draw_RobotStatus();
        Draw_FricRammer(GetGimbalState()->fric_state,GetGimbalState()->rammer_state);
        Draw_CapV(CalcCap()->capvol,CalcCap()->cappercent);
        Draw_YawPitch(GetGimbalState()->yaw,GetGimbalState()->pitch);

        vTaskDelayUntil(&currentTime, UI_TASK_PERIOD/portTICK_RATE_MS);
    }
}

```

二. 移植成功验证：

1. 打印验证

完成上述配置后，将代码烧录到与裁判系统相连的主控板上，连接打印串口会出现如下

结果:

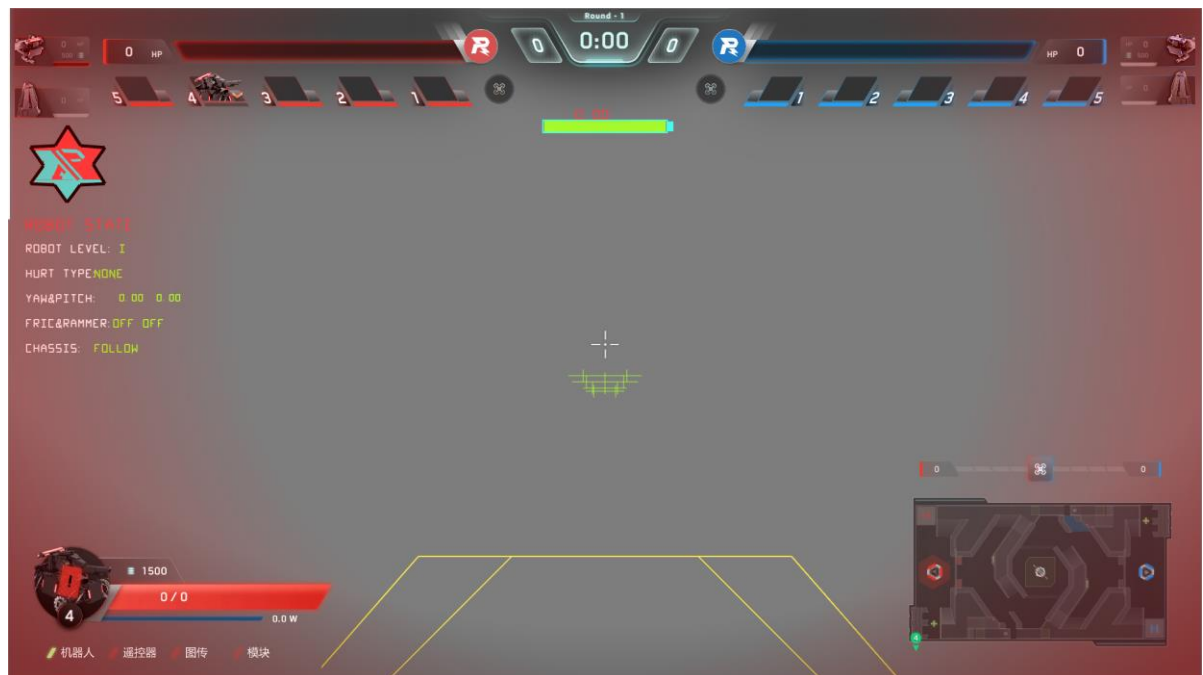
[22:42:42.295] 裁判系统接线错误

连接上裁判系统串口后会出现如下打印结果:

[22:39:42.375] 裁判系统初始化成功

2. 客户端 UI 画图验证

在 RM 官网下载操作手客户端, 打开服务器并将裁判系统主控模块联网后, 打开操作手客户端, 按 P 键登录自己机器人兵种, 可以看到如下界面画出来:



三. 裁判系统数据调用

1. 调用接口函数

可以直接调用 judge.c 最后写的接口函数, 根据自己的需求调用函数返回数据

```
//下面是接口函数区, 需要什么数据就写函数返回什么数据
//如下:

ext_game_robot_status_t* Judge_GetRobotState(void)
{
    return &judge_rcv_mesg.robotState;
}

ext_game_robot_pos_t* Judge_GetRobotPosition(void)
{
    return &judge_rcv_mesg.robotPosition;
}
```

可以查看返回的结构体类型获得需要的数据, 调用过程如下:

```
printf("Bullet_Speed:%f\n", Judge_shoot_data()->bullet_speed);
```


2. 接口函数编写

目前裁判系统串口解包可以得到如下数据：

```
typedef struct
{
    ext_game_state_t          gameState;//比赛状态信息
    ext_game_result_t         gameResult;
    ext_game_robot_HP_t       robotHP;
    ext_ICRA_buff_debuff_zone_status_t BuffDebuffStatus; //人工智能挑战赛加成与惩罚区状态
    ext_event_data_t          fieldEvent;//场地事件
    ext_supply_projectile_action_t supplyAction;//补给站动作
    referee_warning_t         warning;//警告信息
    ext_dart_remaining_time_t  DartRemainTime;//飞镖发射口倒计时
    ext_game_robot_status_t    robotState;//机器人状态
    ext_power_heat_data_t      powerHeat;//实时功率热量
    ext_game_robot_pos_t       robotPosition;//机器人位置
    ext_buff_t                robotBUFF;//机器人增益
    ext_aerial_robot_energy_t  droneEnergy;//空中机器人能量状态
    ext_robot_hurt_t           robotHurt;//伤害数据
    ext_shoot_data_t           robotShoot;//射击数据
    ext_bullet_remaining_t     bulletRemaining;//剩余子弹数，只针对无人机和哨兵
    ext_rfid_status_t          rfid_status;//机器人 RFID 状态
    ext_dart_client_cmd_t      DartClientCmd;//飞镖机器人客户端指令数据
    ext_student_interactive_all_data_t student_interactive_all_data;//机器人间学生串口数据
    self_control_interactive_all_data_t self_control_interactive_all_data;//自定义控制命令交互数据
    ext_robot_command_t        ext_robot_command;//小地图交互数据
    key_and_mouse_t           key_and_mouse;//鼠标和键盘数据
} receive_judge_t;
//解包后获得的数据存在此结构体内
```

可以按照上方接口函数格式返回自己需要的数据。

四. 问题或其他需求

移植遇到问题，或者 UI、裁判系统等方面有其他需求，请联系徐翔宇—QQ: 2579228846
电话：19822920653，或者 coding 任务评论留言。