|  |
| --- |
| 西安邮电大学 |
| 软件设计规划 |
| 飞思卡尔杯电磁组软件设计 |
|  |
| **sx** |
| **2015/6/23** |

|  |
| --- |
| 对飞思卡尔杯电磁组软件从需求到编程的分析 |

**本规划目录与内容：**

1．**目的**

对飞思卡尔比赛的要求明确确定，列出需要实现的功能。

**2. 条件**

列出可以使用的资源。

**3. 总体模型**

根据资源设计可能的程序模型，完成所要求的功能。分析各个模型，选出最优的一个。

**4. 具体模型**

将选出的模型细分，将每一个功能所需要的功能明确。

**5. 程序概述**

写出程序，予以实现，并进行调试。

**6.调试**

优化，排错。

**1．目的**

电磁组飞思卡尔比赛的要求：

在比赛当天，一般：

车子由第三方保管，赛前有20分钟时间调整测试车子的时间，但是程序不可改变。

进入赛场后有60秒的时间做最后的准备。

比赛开始，放车子于起跑区，范围为1.5m长，30秒之内离开。为

永磁铁为起点信息。

比赛途中，四轮中两个不能位于跑道之外，两辆车不能接触。

到终点后，在3m之内停止。

有两次冲出跑道的机会。

成绩=max(t1, t2) + 5 \* |t2 – t1 |。

对于使用传感器的要求：

总数不超过16。

可用的有：

面阵CCD，TSL1401，光电管，激光发射管，电磁传感器， 其他自选传感器

赛道：

赛道中心线的曲率半径大于50cm。

坡道的坡度不超过 20°。 坡道可以不是对称的。 坡道的过渡弧长大于 10 厘米。 坡道的长度、高度 没有限制。一般情况下坡 道的总长度会在 1.5 米左 右。电磁 组的导引线铺设在坡道的表面。

其他要求：

车宽不超过25cm，分赛区预赛无坡道，其他俱全。

**2.条件**

**传感器：**

电磁传感器：

检测磁场，用于识别赛道。

超声波传感器：

检测障碍物，有效距离2cm-300cm。优点是对距离处理简单精确。

面阵CCD：图像

TSL1401：线性ccd，线图像

光电管：光线

蓝牙：保密，传输数据量小，1M/s，性价比高，

24g：与蓝牙相比传输速率大，但应用无蓝牙广泛。

红外通信：指向性，带宽大，干扰小稳定。功率高，距离短。

其他：

**编码器：**

欧姆龙200线单向。

**主控：**

**MKL25Z128VLK4：**

M0+内核，和各种外设。

**内核外设：**

**NVIC：**

31 - 3个IRQ，PIT共用一个IRQ，端口A、D各有一个IRQ可接收外部中断，DMA 的4 个通道有4个IRQ。其他的片内外设每个模块有一个IRQ。每个IRQ有四级中 断优先级。

**片内外设：**

**系统模块**：

**SIM：**

时钟分频，选择，门开关控制

**SMC：**

所有低功耗模式的进入与退出，事件触发。

**PMC：**

内部电压控制，低电压检测，两个电压跳变点，每一个点有四个警告等级。

**MCM：**

表明交叉开关的连接状况以及控制其仲裁方式，对buffer和cache的控制。

**Crossbar-Light Switch：**

控制所连接的各种主从方的并行通信。无硬件寄存器。

**Peripheral Bridge：**

控制交叉开关与外设的连接，独立时钟。

**DMA：**

63个外设插槽，4个随时连线的插槽，4个通道。前两个通道有触发功能。

**Watchdog：**

看门狗。可在SIM中关闭/打开。

**时钟模块：**

**MCG：**

对内外部时钟分/倍频，选择时钟源，选择输出时钟，FLL，PLL，外部时钟检测， 时钟修正。

**OSC：**

为MCG提供输入时钟，可直接为外设提供一些时钟。

**存储器和存储器接口：**

**F****lash Memory ：**

1Kbyte扇区，128kb，可编程、擦除，中断产生，非法访问保护。

**Flash** **Memory Controller：**

提供8,16,32位的读Flash的接口。buffer与cache。

**SRAM：**

16KB，SRAM\_L: 0x1FFF\_F000 – 0x1FFF\_FFFF ，SRAM\_U: 0x2000\_0000 – 0x2000\_2FFF

**定时计数：**

**TPM：**

三个模块，16位，模块0有6个通道，模块1、2各有2个通道，有输入捕捉，输出比较，PWM等功能。计数器可以以上，上下方式运行。可以响应触发，也可以在计数器溢出时产生硬件触发。时钟可选内部或外部。

**LPTMR：**

16位，外部计数时有4个输入通道，可计时，或脉冲计数。有预分频器/故障 滤波器，硬件触发输出，所有低功耗模式中可以运行。

**PIT：**

两个计数器，32位。可以使用为周期的触发DMA、中断。可以形成计数器链 以扩展计数范围。

**RTC：**

32位的秒计数器，带翻转保护和32位的报警器。16位的带补偿的预分频器， 1Hz方波输出。

**模拟模块：**

**ADC：**

最高分辨率16位，4对差分和24个单端模拟输入，可配置的抽样和转换时 间，连续转换，硬件触发转换，硬件均值，可编程的比较值，温度传感器。

**DAC：**

12位。输出从1/4096Vin到Vin，步长是1/4096Vin。

**CMP：**

正反两个8输入的多路复用器，其中两个使用内部源，剩余6个使用外部 源。附带一个6位的DAC作为一个内部源的来源。可以用比较结果来触发其 他外设。

**通信模块：**

**UART：**

三个模块，全双工标准不归零形式，可编程的波特率，中断，DMA或查询操 作。分隔符产生与检测。

**SPI：**

两个模块，8位长度。主模式或从模式操作，全双工或半双工模式。可编程的 位传输速率，DMA传输支持。

**I2C：**

兼容I2C-Bus规格，多主机操作，十位地址外扩，DMA支持。

**USB：**

USB1.1和2.0兼容，16个双向端点，DMA或FIFO数据流接口，低功耗，On-T he-Go协议逻辑。

**其他：**

**GPIO：**

在以0xF800\_0000为基地址的存储区可以通过内核直接单周期访问，也可以在 0x4000\_f000处通过交叉开关多周期访问。

**TSI：**

支持16个外部电极，16位分辨率寄存器，DMA传输。

**3.总体模型**

在系统的功能，效率不受影响的情况下，根据系统简单化和模块化的原则，作出如下模型与限制：

**模型：**



**条件：**

在数据采集时，将外界的信息完全地采集进存储器中保存，除非中断，否则数据处 理和输出执行完全由存储器中的数据决定。也就是说，除了中断发生，在数据采集 阶段以后，程序在一个循环内的执行结果将是确定的。

程序的下一阶段的执行开始依赖于上一阶段的执行完毕。

在这个条件下，数据流图会很简单，所以就不必在画出来了，在编程概述一节已将接口 函数和数据类型定义给出。

**分析**

系统具体的流程为执行各种所需功能的初始化，然后采集外界数据，包括io采集各种控制标志，AD采集道路信息，测距传感器的距离信息，无线模块的两车之间的通讯信息。然后执行数据处理，包括开始、停止信号的判断；综合道路、两个车之间距离、通讯状况得出角度、速度。最后输出执行，采取适当的输出方式使控制信息及时准确地到达实际，比如PID。

在这个过程中必须有的是调试功能，包括按键，显示屏，串口进行调试功能的实现，其中，按键，串口放入中断，显示屏功能放入循环。

**根据以上过程，作出下列选择：**

**道路信息的采集：**

有如下方案：

1.对每一个传感器，实现采集一个正交方向的最大最小值做为参考，然后以此 将以后的AD值标准化。优点是cpu处理方便，计算准确，缺点是操 作费时间，不易掌握。

2.在采集过程中，以一次采集数据中的最大值和最小值作为唯一结果的参考。 优点是操作方便，缺点是cpu不容易处理，不能合理扩展。

综上，决定在道路信息的采集方案上采用第一中方案的简化版，即值采集最大值， 最小值默认为600。

**电机控制：**

PID，使用编码器测得当前速度。

**舵机控制：**

差分处理，近似将上次的计算结果看做舵机下一次的实际角度。可以适当提高响应 速度。

**测距：**

分离式的超声波传感器，可以有效的排除干扰，增大检测范围。

**无线通讯：**

蓝牙加UART。满足要求即可。

**调试部分：**

按键用3个，一个功能选择，两个调整当前所选功能状态。串口同样选择UART和 蓝牙。显示屏为128\*64的OLED。

**4.具体模型**

**初始化**

初始化阶段是做启动时的准备工作的，只执行一次。

包括各个模块中所产生的初始化函数，是各个模块初始化的集合。也可以做一些额外的启动准备。比如延时。

**数据采集**

数据采集模块对外界的信号收集，并进行适当的处理，对应的数据有：

1.启动/停止信号，外部io电平输入信号。

2.干簧管检测信号，外部io电平输入信号。

3.归一化参考值采集开始/结束信号，外部io电平输入信号。

4.速度档位控制信号，暂定3个档位，使用3个外部io电平输入信号。

5.距离信号的提取，使用超声波时，应该在中断中或DMA使检测的结果存处于一个内 存区，然后在数据采集阶段读取。

6.通讯信号的提取，使用UART时，应该使用UART中断或DMA将结果存储区既定的内 存区，然后在数据采集阶段读取。

7.通过电磁传感器采集到的信息，在这一个信息上，应该将所的到的数字值归一化为一 个标准区间里的值，以使模块间接口统一。

**数据处理：**

数据处理模块的输入信号是数据采集模块中的所有采集到的信息。所需完成的工作有：

根据各种开关量决定程序走向。

根据道路信息，建立适当的模型得出当前理想的，所需达到的角度，速度。角度的 计算根据AD值求出偏移值，再到车子相对于赛道的斜率，再将斜率转化为角度。 速度根据角度确定。

**输出执行：**

输出执行阶段是对数据处理阶段处理结果的执行，包括：

选择合适的控制方法，输出速度，角度到实际。比如PID。

进行无线通讯，对另一个车子报告自身的状态，包括启动/停止信号，速度档位， 速度，角度，还有将超声波传感器的测距值进行共享。如果有必要，可以附 加其他信息，比如道路状况。

**资源划分：**

道路检测：6个电磁传感器，一个ADC（6个单端通道）

速度驱动：1个电机，1个tpm（2个通道）

转向：1个舵机，1个tpm

测距：超声波传感器，1个tpm，一个io

通讯：蓝牙，1个UART

测速：1个编码器，1个LPTRM，1个PIT

开始/停止信号，干簧管信号，归一化参考值采集信号，三个速度档：6个10口。

按键调试：3个io。

显示调试：1个OLED，5个io。

串口调试：1个uart。

**5.程序概述**

在写程序这一阶段，按各个模块分类，概述如下

**初始化：**

文件名：

InitEM.c, InitEM.h：

收集各个初始化函数，并安排其顺序。

主要函数（即主要对外函数）：

void InitEM();

**数据采集：**

数据采集包括两车距离信息的采集，也就是以合适的方法、间隔驱动超声波传感器。

文件名：

DataCollection.c, DataCollection.h：

收集数据。

主要函数：

int\_8 DataCollection(DataEM\_T \*DataEM);其中DataEM\_T为所处理数据的 结构体类型。

**数据处理：**

文件名：

DataProcessing.c, DataProcessing.h:

处理数据。

主要函数：

int\_8 DataProcessing(DataEM\_T DataEM, OutputD\_T \*OutputV);//其中 OutputD\_T为数据处理结果类型。

**输出执行：**

输出执行需包括通讯信号通过UART的输出。

由于PID要求时间精度比较高，所以PID主要放入中断执行。

输出执行分为两个文件，一个完成电机和舵机的标准控制，另一个调用其并采用适当的 控制算法完成输出。

文件名：

OutputControl.c, OutputControl.h：

执行输出。

主要函数：

int\_8 OutputControl(OutputD\_T OutputV);

文件名：

DriveMS.c, DriveMS.h：

电机和舵机的标准化接口的实现。

主要函数：

int\_8 PowerM(int\_32 Power);//控制电机功率

int\_8 DegreeS(int\_32 Degree);//控制舵机角度

**调试：**

文件名：

DebugEM.c, DebugEM.h:

调试。

主要函数：

void DisplayOLED(Display\_T Display);//显示

void UartDebug();//串口

void ButtonAdd();//按键

void ButtonSub();

void ButtonSelection();

**数据类型定义（即模块之间的接口）：**

**DataEM\_T**：采集阶段所采集的信息集合，作为数据采集和数据处理之间的交接。

typedef struct

{

Flag\_T FlagC;//控制标志

uint\_16 Distance;//两车距离，和AnotherDataE.Distance只有一个有效。

Message\_T Message;//接收到的通讯信息

ResultAD\_T ResultAD[6]; //AD结果

}DataEM\_T;//数据采集阶段的数据集合

以下为DataEM\_T定义中所依赖的各个类型定义：

typedef struct

{

uint\_16 StartF : 1;//开始信号

uint\_16 RunF : 1;//干簧管有关信号，这个信号标志应该是与干簧管传感器输出对应，且 //是边沿敏感

uint\_16 NormalF : 1;//归一化采集信号

uint\_16 GearF1 : 1;//速度一档，最低

uint\_16 GearF2 : 1;//速度二档，中

uint\_16 GearF3 : 1;//速度三档，最高

}Flag\_T;//控制标志类型定义

typedef struct

{

uint\_16 StartF : 1;//开始信号

uint\_16 RunF : 1;//干簧管有关信号

}AnotherFlag\_T;//另一个车的控制标志类型定义

typedef struct

{

char Communication[15];//从communication[0]依次为速度（4个字符），角度（4个字符）， //距离（3个字符），StartF（1个字符）， RunF（1个字符），速 //度档位的选择（1字符）

AnotherFlag\_T Flag;//控制标志

uint\_16 Distance;//距离

int\_16 Speed;//速度

uint\_16 Angle;//角度

}Message\_T;//通讯信息类型定义

typedef struct

{

uint\_16 RealV;//实际转换值

uint\_16 NormalV;//归一化后的值

uint\_16 Max;//参考最大值

uint\_16 min;//参考最小值

}ResultAD\_T;//AD转换结果有关的信息类型定义

**OutputD\_T**：数据处理阶段的结果，输出执行阶段的输入。

typedef struct

{

int\_16 Speed;//速度

uint\_16 Angle;//角度

Message\_T Message;//要发送的信息

}OutputD\_T；

备注：数据类型有修改，参考实际程序的.h文件

**6.调试**