软件目录

[三.软件介绍](#三软件介绍)

笔记软件

1使用OneNote 管理自己做毕设时的笔记，将一些知识点，易忘内容，提醒事项记录在内，可以手机，ipad，PC同时在线记录。

2有道云笔记支持Markdown，Markdown是一种轻量级的「标记语言」，通常为程序员群体所用，目前它已是全球最大的技术分享网站 GitHub 和技术问答网站 StackOverFlow 的御用书写格式。

利用Markdown可以做什么？

1.代码高亮

2.制作待事项To-do List

3.高效绘制 流程图、序列图、甘特图、表格。方便我们梳理设计毕设流程图。

4.书写数学公式

Markdown常用代码：制作流程图时百度使用

3XMind 是一款非常实用的商业[思维导图](https://baike.baidu.com/item/%E6%80%9D%E7%BB%B4%E5%AF%BC%E5%9B%BE)软件

|  |  |  |  |
| --- | --- | --- | --- |
| XMind不仅可以绘制思维导图，还能绘制 | | | [鱼骨图](https://baike.baidu.com/item/%E9%B1%BC%E9%AA%A8%E5%9B%BE)、二维图、  [树形图](https://baike.baidu.com/item/%E6%A0%91%E5%BD%A2%E5%9B%BE)、  [逻辑图](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E5%9B%BE)、  [组织结构图](https://baike.baidu.com/item/%E7%BB%84%E7%BB%87%E7%BB%93%E6%9E%84%E5%9B%BE)（Org、Tree、Logic Chart、Fishbone）。并且，可以方便地从这些展示形式之间进行转换。 |
| 可以导入 | MindManager、FreeMind数据文件。灵活的定制节点外观、插入图标。丰富的样式和主题。输出格式有：HTML、图片 | |

|  |  |
| --- | --- |
| XMind 的文件可以导出成 | Microsoft Word、Microsoft ，  [PowerPoint](https://baike.baidu.com/item/PowerPoint)、  [PDF](https://baike.baidu.com/item/PDF)、图片（包括PNG、JPG、GIF、BMP 等）、  [RTF](https://baike.baidu.com/item/RTF)、  [TXT](https://baike.baidu.com/item/TXT)等格式， |
| Table | 子标题 |
| Enter | 同级标题 |
| 空格 | 输入文字 |

git

在我们编写程序的时候，需要不断修改程序，并且可能出现我们要借鉴使用昨天的程序，所以得经常保存不同版本的程序，直接在文件夹里面复制粘贴费事，而且占用大量存储空间，因此我们这里用到了git。

Git版本控制系统可以告诉你每次的改动，比如在第5行加了一个单词“Linux”，在第8行删了一个单词“Windows”。而图片、视频这些二进制文件，虽然也能由版本控制系统管理，但没法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是只知道图片从100KB改成了120KB，但到底改了啥，版本控制系统不知道，也没法知道。

除此之外，我们可以在在GitHub上免费托管Git仓库，因github是国外网站，进入可能需要时间等待，因此我们也可以使用国内的Git托管服务——[Gitee](https://gitee.com/?utm_source=blog_lxf)。

在使用git控制不同版本程序的时候我们经常使用一下命令码。

.1使用git时常用的基本操作

|  |  |
| --- | --- |
| 目录 |  |
| Pwd | 显示当前路径 |
| ls | 列出当前目录中的所有文件 |
| reset | 清屏 |
| 改变路径 |  |
| cd e:\GitRepository | 改变当前目录到\* |
| cd .. | 退回到上一个目录 |
| 文件/夹操作 |  |
| mkdir: | 新建一个文件夹 |
| touch : | 新建一个文件 |
| rm: | 删除一个文件 |
| mv | 移动文件,；mv 目标文件夹 目标文件 |
| 版本库操作 |  |
| Git init | 把当前目录文件夹设置为版本库 |
| git add readme.txt | 把.txt文件添加到版本库里面，可多次使用，最后使用一个commit提交 |
| $ git commit -m "this change explain" | 把本行代码上述添加到版本库里的文件一次性提交 |
| $ git status | 查看仓库文件是否修改、是否添加提交 |
| $ git diff | 查看仓库文件修改的地方 |
| $ git log | 查看某文件过去修改的不同版本日志。log：日志 |
| $ git reset --hard ID号前几位 | 把文件恢复到 某个版本，新版本/旧版本 |
| $ cat readme.txt | 查看文本内容 |
| $ git reflog | 查看用户曾经的命令，可显示ID号，方便更改版本。ref参考 |
| $rm test.txt | 删除工作区的某个文件 |
| $ git rm test.txt | 删除版本库中的某个文件 |
| $ git remote rm origin | 删除远程库origin |

.2Git bash分支命令码

|  |  |
| --- | --- |
| 分支 |  |
| $ git checkout -- readme.txt | 让工作区文件撤销恢复到最近一次git commit或git add时的状态，是哪种状态由文件是否复制到了暂存区stage |
| $ git checkout -b dev origin/dev | 本地git创建的dev分支  与github里origin仓库里的dev分支对应 |
| =$git branch dev  $git checkout dev | $创建dev分支  $切换到dev分支 |
| $git checkout master | 切换到master分支 |
| $git branch | 查看分支 |
| $ git branch -d dev | 删除分支dev |
| $ git branch -D dev | 删除未提交的分支（分支修改了半天不用它了） |
| $ git merge dev | 把分支dev合并到当前分支  merge：合并 |
| $ git merge --no-ff -m "merge with no-ff" dev | 合并分支要创建一个新的commit，所以加上-m参数，把commit描述写进去。不使用快进合并 |
| git log --graph | 查看分支合并图。 |
| Git stash | 当前文件修改之后没有add到暂存区，没有commit提交时用本行代码把工作现场储藏起来，储藏到stash |
| git stash list | 查看存放到stash的工作现场 |
| Git stash apply | 把stash里的工作现场恢复到曾经的路径 |
| Git stash drop | 把stash里的内容删除 |
| git cherry-pick ID号 | 把某次提交的修改复制到当前分支上。写32的时候经常用到，在主控板做了修改，驱动板也要做同样的，直接用本行代码 |

.3Git bash标签命令码

|  |  |
| --- | --- |
| 打标签 |  |
| Git tag v1.0 | 默认在最新提交的commit上打上标签v1.0 |
| Git tag v0.9 f52c633 | F52c633 需要打标签的ID号 |
| $ git tag -a v0.1 -m "version 0.1 released" 1094adb | -a指定标签名  -m指定说明文字 |
| Git tag -d v1.0 | 删除标签 |
| 查看标签 |  |
| Git tag | 查看标签标签按字母排序，不是发布时间 |
| git show v0.9 | 查看标签信息 |

MDK介绍

本次毕设MDK版本为MDK5.29，该版本使用 uVision5 IDE 集成开发环境，是目前针对 ARM 处理器的最新版本，是 Cortex M 内核处理器的良好开发工具。

MDK5 由两个部分组成：

|  |  |
| --- | --- |
| MDK Core |  |
| Software Packs | 可以独立于工具链进行新芯片支持和中间库的升级。 |

MDK Core 又分成四个部分：

|  |  |
| --- | --- |
| uVision IDE with Editor | （编辑器） |
| ARMC/C++ Compiler | （编译器）， |
| Pack Installer | （包安装器） |
| uVision Debugger with Trace | （调试跟踪器） |

Software Packs（包安装器）又分为三个小部分，：

|  |  |
| --- | --- |
| Device | （芯片支持）， |
| CMSIS | （ARM Cortex 微控制器软件接口标准） |
| Mdidleware | （中间库）通过包安装器， |

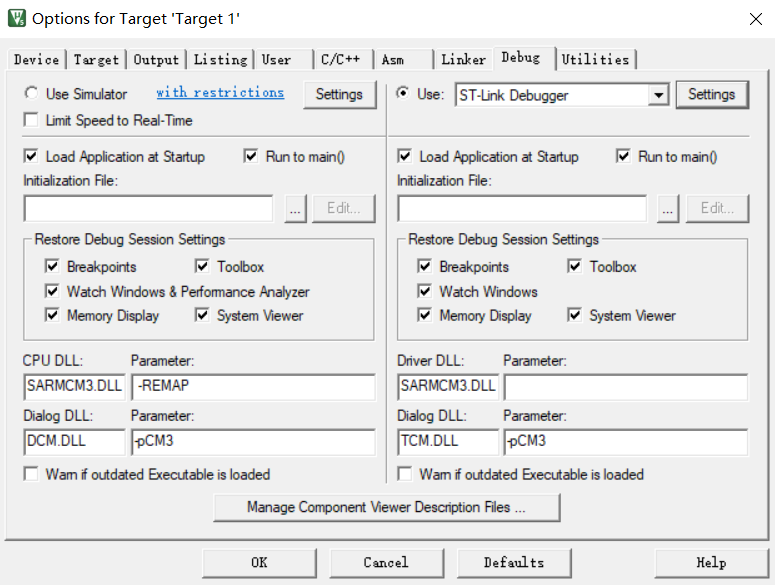
我们可以安装最新的组件，从而支持新的器件、提供新的设备驱动库以及最新例程等，加速产品开发进度

MDK下载安装

1. 在 [www.keil.com](http://www.keil.com) 官网 下载MDK5.29软件，并下载芯片的开发包Keil.STM32F1xx\_DFP
2. STM32F1开发有寄存器版本,HAL库版本，标准库版本。HAL 库和标准库  
   本质上是一样的，都是提供底层硬件操作 API，而且在使用上也是大同小异。这里选用标准库版本。

程序下载方式

STM32F1的程序下载方式有USB、串口、 JTAG、 SWD 。但串口只能下载代码，并不能实时跟踪调试，而JTAG占用IO口线较多，所以这里选择SWD下载，选择调试工具ST-LINK。除此之外还有 JLINK 、ULINK 等调试工具可供选择。

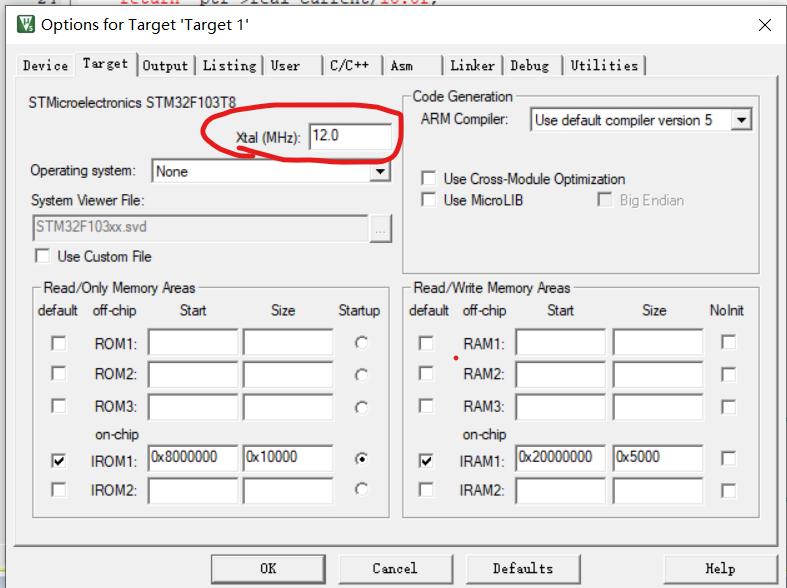


程序仿真调试

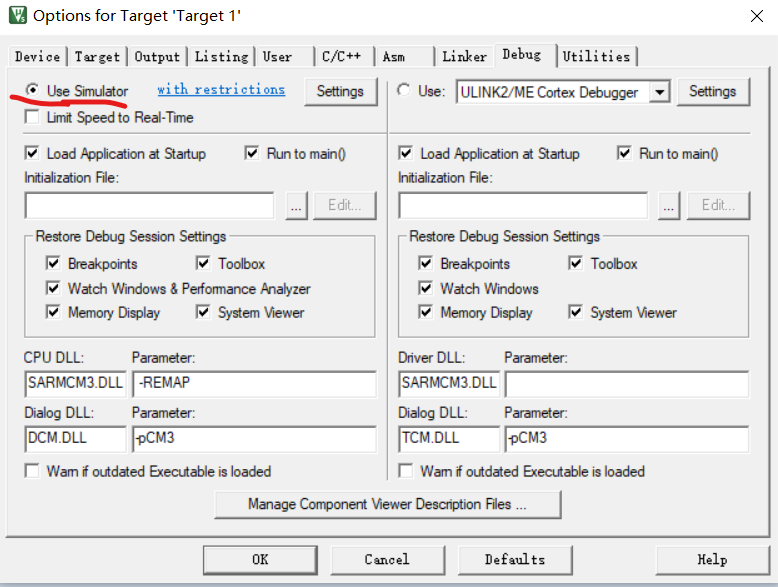
|  |  |
| --- | --- |
| 软件仿真调试 | 无实物，纯软件和代码调试,  只能调试简单的I/O口操作，高电平低电平简单的看了出来 |
| 硬件仿真调试 | 通过下载器与电路板连接起来，可调试复杂外设BUG |
|  | 看到代码在单片机内部里面如何运行、变量是如何传递、变量的变化情况等等。 |

MDK仿真调试步骤

1.进入Target,输入单片机芯片所用的晶振：



2.进入Debug,选择 Use Simulator



3.仿真无法运行，修改在debug下面配置成如下参数

|  |  |  |  |
| --- | --- | --- | --- |
| SARMCM3.DLL | -REMAP | SARMCM3.DLL | -REMAP |
| DARMSTM.DLL | -pSTM32F103T8 | TARMSTM.DLL | -pSTM32F103T8 |

|  |  |
| --- | --- |
| SARMCM3.DLL | 是一个动态链接库，SARMCM3表示是ARM Cortex-M3架构 |
| DARMSTM.DLL、TARMSTM.DLL | 指明了是STMicroelectronics（意法半导体）的ARM架构的产品。 |
| -pCM3 | 表示Cortex-M3系列的参数，由于Cortex-M3只是个大框架，具体的厂商实现细节不一样，所以得修改的具体点。 |

4.

点击调试按钮：

C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\17B55C2C.tmp

5.调试方法

|  |  |
| --- | --- |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F1930F1A.tmp | 黄色的箭头代表的就是程序当前运行的位置 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F74BB638.tmp | RST（复位），单击之后，程序就会跑到最开始的位置运行，即可回到main函数的开头处 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5AD0D106.tmp | 全速运行，单击之后程序就会全速跑起来 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\42618B04.tmp | 停止按钮，当程序全速运行起来时候，单击停止按钮程序就会立即停止，可以观察程序运行到哪里去了。 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\A5539BB2.tmp | Step，单步执行，当碰到子函数时，则进入子函数。 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\3D2F2690.tmp | Step Over:单步执行，碰见子函数时不进入，将子函数当作一个整体来运行。 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B5959B1E.tmp | Step Out；单步执行，程序若在子函数内部执行，则跳出子函数。 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\442B94DC.tmp | Run to Cursor Line:运行到光标处。 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\FE7DBB4A.tmp | Serial Windows :串口输出窗口。 |
| C:\Users\lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\CE98A1E8.tmp | Analysis Windows；逻辑分析窗口，该窗口下有Logic Analyzer，可以查看波形、变量等数值的观察窗口、 程序运行的时间。 |
| 断点 | 单击复位之后，可以看到C语言程序的窗口左侧有灰色或者保持着原来的颜色，其中有灰色的地方是我们可以设置断点的地方，不能设置断点的地方是因为keil具有程序优化的功能，如果我们想在任何地方都可以设置断点的话，需要更改工程选项里的优化等级，把等级设置为0即可，即高速keil不要对程序进行优化。 |
| 计算单片机运行了多长时间 | 设置断点和查看sec，sec显示了单片机运行的时间。sec位于Registers窗口里面 |
| 查看某一个变量或者寄存器的值, | 点击view-> watch windows,在里面的name选项输入想要查看的变量的名字或者寄存器的名字，随着程序的运行，我们就可以实时看到他们的值的变化 |

四.软件介绍

STM32F103ZET6

主控程序

主控程序用到的模块：

1

CAN1初始化：CAN一次性传送一个ID数据时，可以传送8帧数据

can1接收中断程序：

接收5个扇形装甲板的击打信息：

ID:0x301-305

Data[0]：0未击中 1击打正确 2击打错误

接收风车电机的信息:

ID:0x204

Data[0]Data[1]：编码器机械角度

Data[2]Data[3]：速度\_每分钟转数

Data[4]Data[5]：实际电流值

Data[7]：电机温度

2.

USART1初始化

USART1接收中断：

仅接收颜色数据：红色/蓝色

3.

TIM2初始化：移位定时器

TIM3初始化：计算时间击打能量机关

定时器定时时间计算：

定时器时钟频率f=时钟源频率/(预分频值+1)=72MHz/ (psc+1)

定时器定时时间=(自动重装载周期值+1) / 定时器时钟频率=(arr+1) /f=(arr+1) \*t

Tim2\_Init(999,7199); // 72Mhz/7200=10000 0.0001\*1000=0.1s=100ms

Tim3\_Init(6999,7199); // 0.0001\*7000=0.7s=700ms

TIM2向上计数溢出中断：随机数种子seed++

TIM3向上计数溢出中断：计数一定时间运行模式进入acting模式，并开始time\_count++;

4.LED初始化

5.

中断优先级：抢占优先级 子优先级CAN1接收：2 0

USART1： 1 0

TIM2: 7 1

TIM3: 6 0

6.

延时函数

7.

初始化PID相关参数

循环程序：

Fail\_Task：

5个扇形上的装甲板信息清0，.ready=0，hitted=0;并通过CAN1发送，发送5个扇形 的：颜色|准备状态|运行模式 TIM3重新使能

Success\_Task：

通过CAN1发送5个扇形的：颜色|准备状态|运行模式，激活成功

TIM3重新使能

Acting任务：

刚上电，随机让一个扇叶准备

检查5个扇形装甲板的击打状态：击打错误，模式=失败

检查某个被选中的扇形装甲板的击打状态：击打成功，选择下一次的扇形装甲板并通过CAN发送5个扇形的：颜色|准备状态|运行模式

处理can接收数据：

ID:0X201

获得：PID控制实际值

PID控制计算并CAN1发送数据

ID:0x200,3508电机

Data[]:pid.out电机输出电流值

主控程序总结：

主控32 接收5个扇形装甲板的击中信息；旋转标靶电机的机械角度，速度值，电流值，温度值；灯的颜色。通过一系列处理数据，确定灯的闪亮灯，进行PID控制。并通过CAN发送5个扇形的：颜色|准备状态|运行模式；电机的PID控制电流值

主控程序IO口：

CAN1: A11,A12

USART1: A2发送A3接收

LED: A5 6 7 8 B0 1 2 5 6 7

点阵驱动板程序

程序用到的模块：

点阵驱动IO口GPIOA0-8初始化配置输出

用来读取扇形装甲板ID1-5的拨码开关初始化配置

TIM2初始化：箭头移动

TIM2中断：CAN1发送此时某个扇形装甲板的击打信息.hitted

点阵驱动LED数据移动，用于图形移动

TIM3初始化：图形累加

返回不同片上时钟的频率： 检验晶振是否正确

CAN1初始化：

can1接收中断程序：

接收ID:0X300

接收扇形击打板被击打信息

Data[0]-Data[5]:5个扇形装甲板信息

Data[]的每一位：

7 6 Colour Ready 3 ActingFailed ActingSucess Acting

按键传感器IO口以及外部中断初始化配置

外部中断程序：

通过ready和acting的状态来确定此时某个扇形装甲板的击打信息.hitted

并通过CAN1发送击打信息

获取此时拨码开关确定的扇形装甲板ID

循环程序：

看门狗喂狗

对点阵显示的控制:

刚开始：点阵驱动LED全亮

总的击打状态：击打成功点阵驱动LED全？ ，击打错误点阵驱动LED全亮

击打错误：点阵驱动LED全亮 ，运行模式清0

击打成功：TIM3使能，点阵驱动LED Up ，运行模式清0

击打进行中，没打中：TIM3使能， DataFlag=BoardFlag=0;

点阵驱动部分的程序说明

void DataShift(unsigned char Data)：

若数据为1个1、7个0这种 JP3\_SI=1；

否则 JP3\_SI=0；

void DataSelect(char SlcNumber)：

选择数字=001，JP3\_A=1

选择数字=010，JP3\_B=1

选择数字=100，JP3\_C=1

否则JP3\_A=0，JP3\_B=0，JP3\_C=0

void ring\_shift\_left(unsigned char \*data)：

调整8个数据:将每个数据的最低位调整到最高位

void DotArrayShowFull(Bit16 colour,u8 State)：

确认点阵驱动IO口的状态让点阵全亮/灭

总结：点阵驱动程序 通过CAN1中断接收点阵驱动的颜色，5个扇形装甲板的状态信息.ready、激活成功失败。通过外部中断确定5个扇形装甲板的击打信息..hitted并通过CAN1发送给灯条板和主控板，并。

程序IO口：

点阵驱动IO口：GPIOA0-8:推挽输出，置高电平

拨码开关输入IO口：GPIOB-3-6，读取扇形装甲板ID号1-5，颜色ID(颜色ID暂时不用)

传感器接口:外部中断:A9

Can1: A11can接收 A12can发送

点阵驱动IO口：

#define JP3\_A PAout(0)

#define JP3\_B PAout(1)

#define JP3\_C PAout(2)

#define JP3\_SCK PAout(3)

#define JP3\_RCK PAout(4)

#define JP3\_SI PAout(5)

#define JP3\_OE\_Blue PAout(6)

#define JP3\_OE\_Red PAout(7)

#define JP3\_OE\_Green PAout(8)

#define Blue 0

#define Red 1

灯条程序

程序模块：

CAN1初始化：CAN一次性传送一个ID数据时，可以传送8帧数据

can1接收中断程序：

接收5个扇形装甲板的击打信息：

ID:0x301-305

Data[0]：0未击中 1击打正确 2击打错误

接收信息

ID:0x300，CAN\_MASTER\_ID

Data[0]：运行模式acting = 1, //正在激活

success = 2, //激活成功

fail = 4, //激活失败

Data[1]- Data[5]：5个扇形装甲板的准备状态ready

LED初始化配置：A5 6 7 9

B0 1 2 5 6 7

循环程序：

运行模式一直检测：

Success：

5个扇形装甲板依次随机全被击打，激活成功

灯条LED 10个IO口全亮

Fail：

未能在 2.5 秒内击中随机点亮的装甲模块 /击中非随机点亮的装甲模块，激活 失败

灯条LED 10个IO口全灭；

5个扇形装甲板的.ready .hitted 全为0

Acting：

查询5个扇形装甲板的.ready .hitted信息

.ready=1 .hitted=0：里面的灯灭，外面装甲板周围的灯亮

.ready= .hitted=1：全亮

.ready= .hitted=2：全灭

灯条程序总结：

灯条主控通过CAN1接收5个扇形装甲板的准备状态ready、击打信息（是否打中，击打错误）以及运行模式正在激活acting/激活成功 success/ 激活失败fail。并实时自动运行灯条显示：(正在激活)仅装甲板外面的灯亮、（激活成功）全亮、（激活失败）全灭

程序IO口：

CAN1: A11,A12

LED: A5 6 7 9 B0 1 2 5 6 7

程序定义说明

灯条程序：

flabellum[i].ready=1;允许击打某扇形装甲板

flabellum[i].ready=0;不允许击打某扇形装甲板

flabellum[i].hitted=0;某扇形装甲板未被击打

flabellum[i].hitted=1; 允许击打的某扇形装甲板被击打

flabellum[i].hitted=2; 不允许击打的某扇形装甲板被击打