**柳工高空作业车控制器下控**

**基础软件开发 - MCU部分**

**软件设计说明书**

**2022.05.06**

# 目录

[目录 2](#_Toc102813925)

[1. 软件代码包文件结构 3](#_Toc102813926)

[1.1 目录结构说明 3](#_Toc102813927)

[2. 调试环境搭建 4](#_Toc102813928)

[2.1 编译配置如图 4](#_Toc102813929)

[2.1 EVB架设 5](#_Toc102813930)

[2.2 产品板架设 5](#_Toc102813931)

[2.3 EVB烧录 5](#_Toc102813932)

[2.4 程序调试 6](#_Toc102813933)

[3. MCU 模块硬件配置信息 7](#_Toc102813934)

[3.1硬件设计 7](#_Toc102813935)

[3.2 软件设计对应的硬件资源 7](#_Toc102813936)

[3.3 端口配置表 7](#_Toc102813937)

[4. RTOS 任务设计 7](#_Toc102813938)

[4.1 可以使用的RTOS资源 7](#_Toc102813939)

[4.2 任务的规划 9](#_Toc102813940)

[4.2.1 一次性任务 9](#_Toc102813941)

[4.2.2 常驻任务 9](#_Toc102813942)

[4.3软件架构 / Software Architecture Decision 9](#_Toc102813943)

[**5. 测试CONSOLE命令 10**](#_Toc102813944)

[5.1 Console命令测试能够覆盖到的组件 10](#_Toc102813945)

[5.2 具体的Console命令 10](#_Toc102813946)

[5.2.1 列出所有命令 10](#_Toc102813947)

# 1. 软件代码包文件结构

## 1.1 目录结构说明

├── Application ---- 开发代码所在目录

│   ├── API

│   ├── BOARD

│   ├── BSP

├── Common ---- 算法目录

│   ├──md5.c

│   ├──md5.h

├── Core ---- 程序时钟中断等入口目录

│   ├──Inc

│   ├──Src

├── Drivers ---- HAL驱动目录

│   ├──CMSIS

│   ├──STM32F4xx\_HAL\_Driver

├── FATFS ---- 文件系统目录

│   ├──App

│   ├──Target

├── MDK-ARM ---- MDK启动目录

├── Middlewares ---- 中间件目录

│   ├──Ring

│   ├──ST

│   ├──Third\_Party

├── USB\_HOST ---- USB APP目录

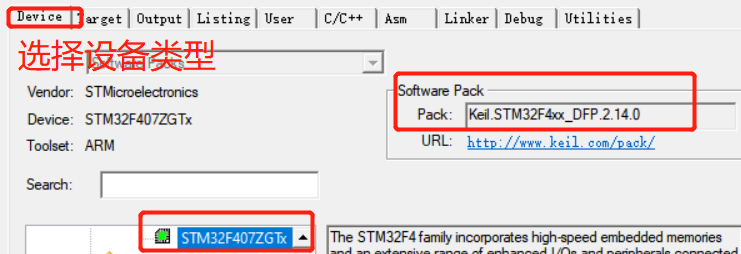
│   ├──App

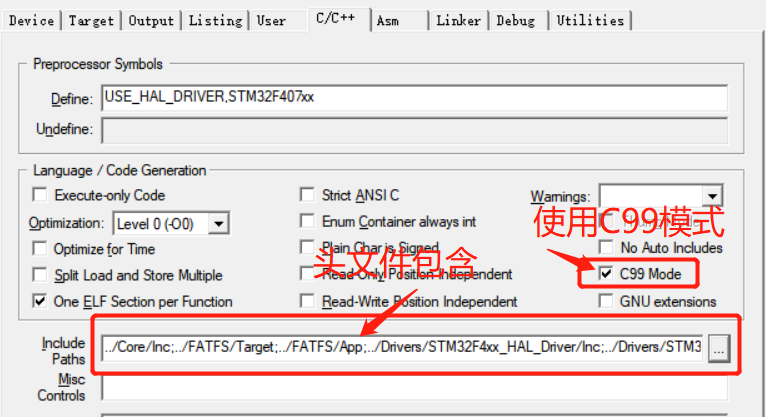
│   ├──Target

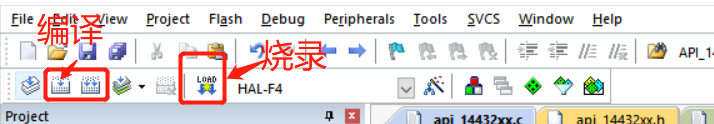
└── HAL-F4.ioc ---- MX CUBE配置

# 2. 调试环境搭建

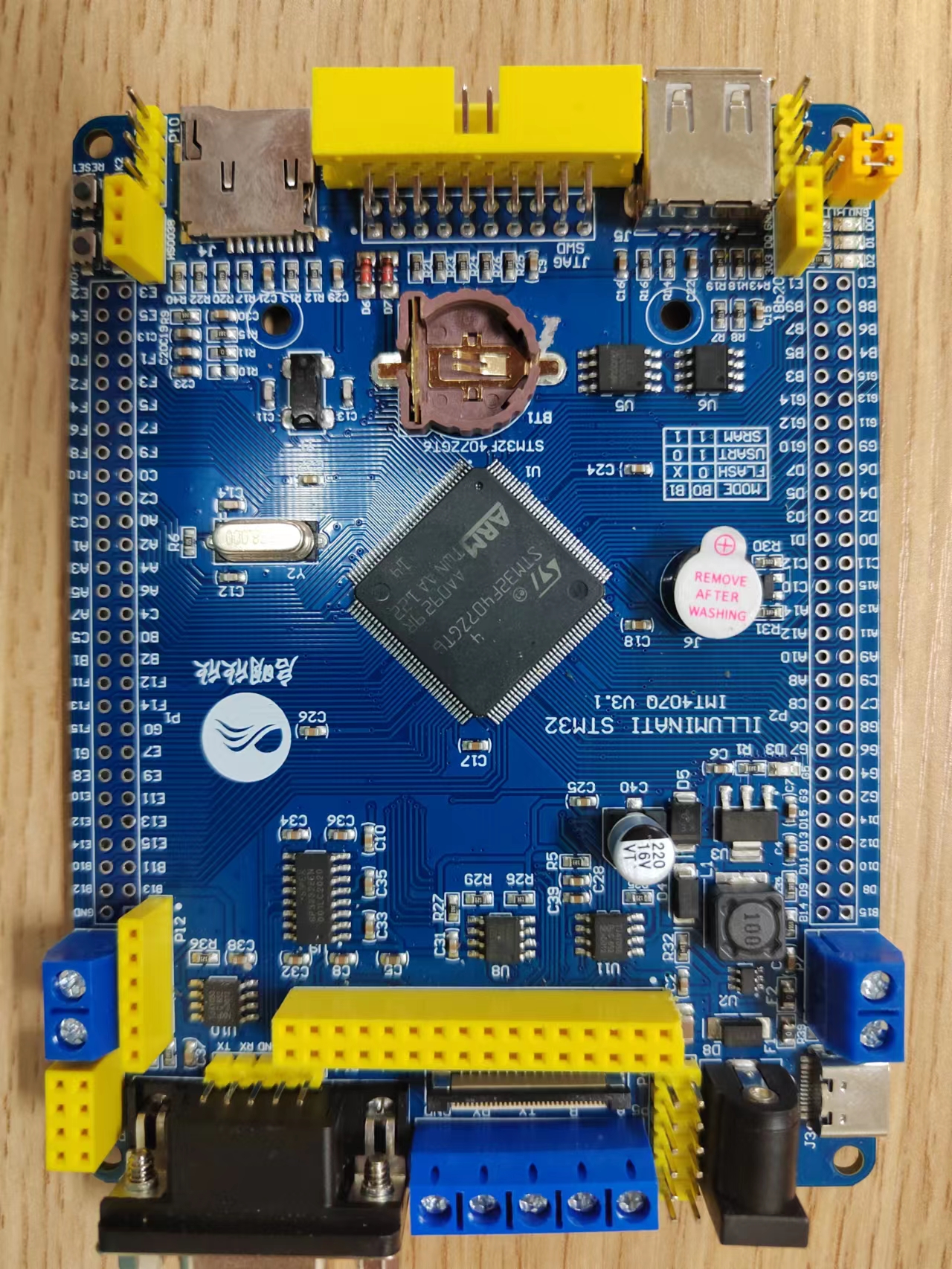
## 2.1 编译配置如图

● 

● 

● 

## 2.1 EVB架设



芯片引脚

芯片引脚

RS232

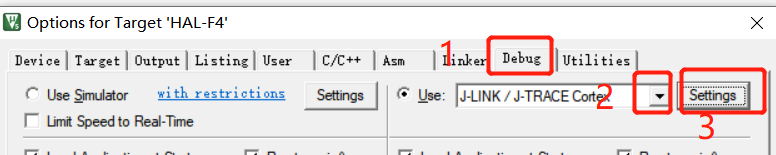
USB

电源

MCU烧录口

## 2.2 产品板架设

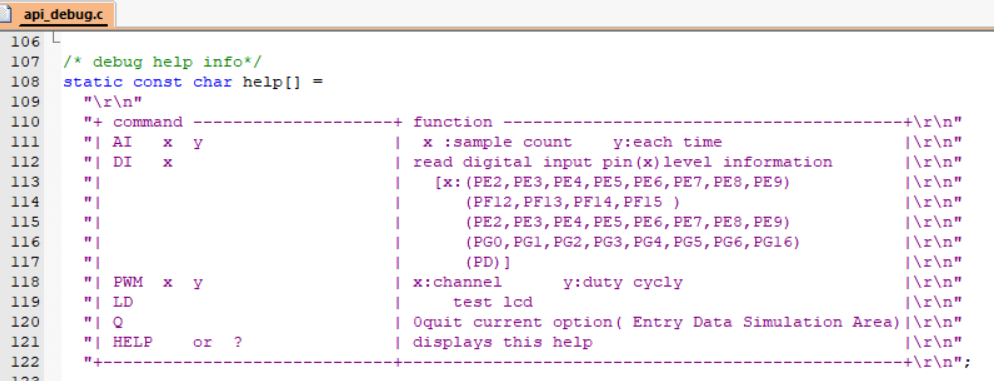
## 2.3 EVB烧录

* STM32F407烧录时,可以使用多种烧录工具。当前我们使用的主要为LJLINK。接上JLINK后如下图:
* 
* 
* 
* 配置完成后烧录程序

## 2.4 程序调试

* MCU支持断点调试等功能，而且可以使用了Console交互调试技术；
* Console交互调试基于串口，在MCU目标机器（开发板或者目标机主板）和电脑间，使用物理串口虚拟出一个机器APP部分的Console端口，类似于Windows的超级终端或者Linux的Terminal终端，Console提供的命令和输出，由App代码来决定。
* 使用Console debug功能在include.h中定义：
  + #define USE\_DEBUG\_INFO

注释掉这句，则调试接口不会编译进程序

* Console 启动时，界面如下：
* 以上的Console端口输出的内容，对应在APP代码中的位置是：
* 
* 后续APP开发时，针对特定需要调试的功能或函数，在Console中添加测试项以及参数带入，并观察程序的相应输出(LOG, Console以及其他数据流)，来进行程序的调试和测试。

# 3. MCU 模块硬件配置信息

## 3.1硬件设计

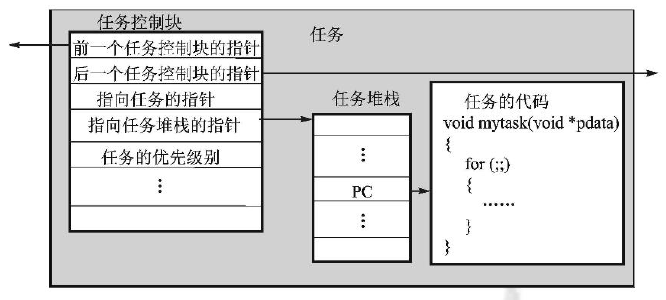
## 3.2 软件设计对应的硬件资源

## 3.3 端口配置表

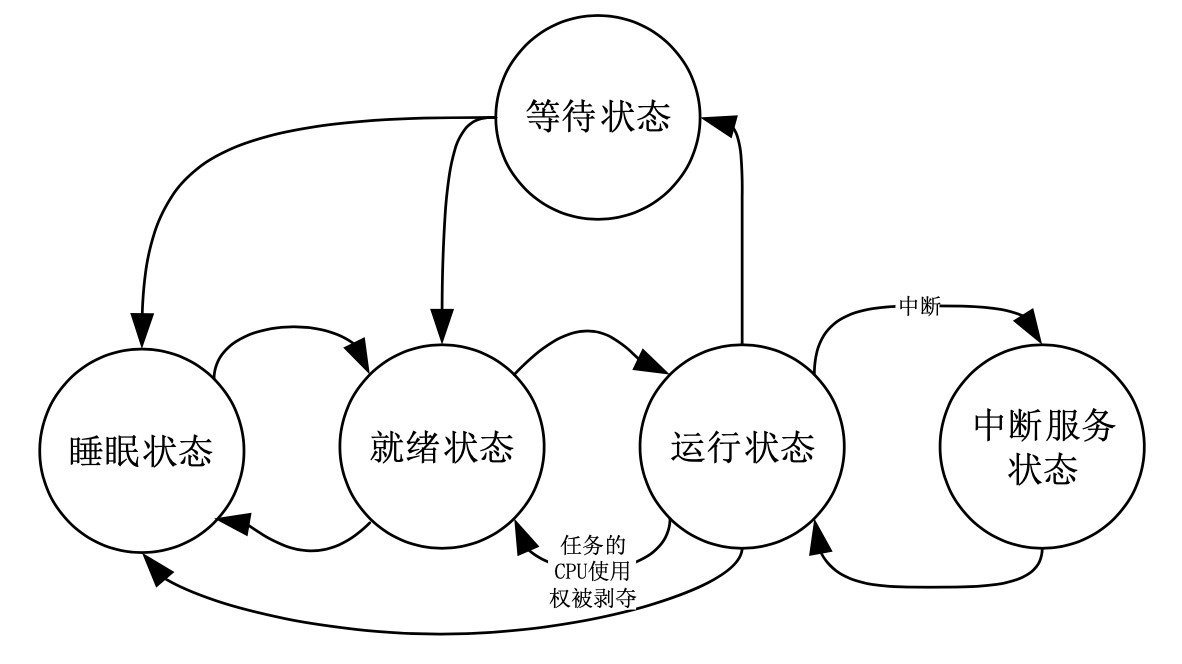
# 4. RTOS 任务设计

## 4.1 可以使用的RTOS资源

* 任务(task)
  + 任务及其内存结构

page10image4540864

* + 任务状态



* + 任务的创建和删除
* 事件通知(event)

*事件通知是 ECx00U&EGx00U 系列 QuecOpen 模块的线程核心通讯方法。事件通知可以结合信号量， 队列，事件标志等通讯方法的多数优点。ECx00U&EGx00U 系列 QuecOpen 模块的多数 RTOS 功能，如 软件看门狗，定时器的运行，也可以通过事件通知实现。因此建议如果线程间涉及数据交互，可以使用事件通知的方式实现。*

* 信号量(semaphore)

*一个复杂的应用程序拆分为多个小任务时，要求多个小任务之间能够很好的协调和同步。比如一个应 用被拆分为任务 A 和任务 B，任务 B 需要等待任务 A 完成某个操作之后才能进行后续工作，此时任务 B 可以处于等待状态，直到任务 A 完成特定的操作后，向任务 B 发送一个信号，通知任务 B 可以执行后续工作。*

* 互斥锁(mutex)

*互斥锁是一个二值型信号量，即发送出来的信号值最大为 1。它用于保护共享资源不被多个任务同时访 问，以免造成共享资源的上下文在同一个任务访问前后不一致，产生非预期的结果，而这些非预期的结果 有时是致命的。所以互斥锁在保护共享资源时尤为重要。*

* 消息队列(queue)

*多任务之间除了发送信号量的方式进行沟通交流之外，还可以通过消息队列进行消息传递。消息队列 用于一个任务向另外一个任务传递消息。传递的消息方式是将待传递的消息根据消息的起始地址和消息大 小，拷贝到消息队列中，然后向等待消息的任务发送信号;等待消息的线程将消息队列中的消息拷贝到本 地的缓冲区中，然后将队列中的消息移除。*

* 定时器(timer)

*定时器，顾名思义，就是用来计时的，定时时间到达后，通过 callback 的形式通知到用户。EC200U-CN的定时器，支持在系统 service 中运行 callback,也支持在指定的 task 中运行 callback。*

* 软件看门狗(software watch dog)

*在某个 task 陷入死循环，或者一直占用 CPU，导致比这个 task 优先级的 task 永远得不到调度，发生这种情况时，软件看门狗将重启模块。*

## 4.2 任务的规划

### 4.2.1 一次性任务

### 4.2.2 常驻任务

## 4.3软件架构 / Software Architecture Decision

* 分层设计概念:
* 功能模块对外调用的模块封装成一个个API，将底层驱动做个API以供功能模块调用。（各个功能模块可以独立编译，或者调用驱动层接口 ）
* API分为驱动层API和应用层API，而不是所有程序都调用驱动层API。(整个应用中都调用驱动层API会导致应用中驱动调用随处可见，无法移植和最大限度的复用)

总体分 硬件驱动层-->系统服务层-->业务逻辑层-->应用层

硬件驱动层: 硬件驱动层包含板载硬件资源正常运行所需的所有驱动程序并提供API给功能模块调用。

系统服务层: 系统服务层包括实现具体功能的函数，通过调用驱动层API实现相应功能，同时提供可调用的API给业务逻辑层。

业务逻辑层: 业务逻辑层包括产品整体功能的各个业务流程，通过调用功能模块层的API实现。

应用层: 应用层将各个业务逻辑进行整合调用，完成整个产品的功能

* 层与层之间不能跨层调用。
* 模块与模块各自独立，无依赖关系。
* 模块提供统一的接口供上层调用，模块的内外接口分明。
* 模块的功能只能增，不能改。
* 各个功能模块层也还可以进行继续分层，比如接口层、驱动层、硬件层。

■ 优势:

● 如果驱动变动了,只需更改驱动层,应用层不受影响。

● 如果功能模块变动了，只需升级相应的功能模块，其他的模块不受影响，应用层也不受影响。

● 按照这种逻辑设计好之后，主要的工作就是在业务逻辑层。应用层则为程序的总体流程和框架，主要调用业务逻辑层实现不同的功能

### 5. 测试CONSOLE命令

## 5.1 Console命令测试能够覆盖到的组件

* GPIO
* ADC
* TIMER

### 5.2 具体的Console命令

## 5.2.1 列出所有命令

Cmd> ?

+ command --------------------+ function ----------------------------------------+

| AI x y | x :sample count y:each time |

| DI x | read digital input pin(x)level information |

| | [x:(PE2,PE3,PE4,PE5,PE6,PE7,PE8,PE9) |

| | (PF12,PF13,PF14,PF15 ) |

| | (PE2,PE3,PE4,PE5,PE6,PE7,PE8,PE9) |

| | (PG0,PG1,PG2,PG3,PG4,PG5,PG6,PG16) |

| | (PD)] |

| PWM x y | x:channel y:duty cycly |

| LD | test lcd |

| Q | 0quit current option( Entry Data Simulation Area)|

| HELP or ? | displays this help |

+-----------------------------+--------------------------------------------------+

* 测试模拟量输出1次每次1ms
* Cmd> ai 1 1
* > DO5\_ADC012\_IN0\_PA0 0.000000
* > DO6\_ADC012\_IN1\_PA1 0.000000
* > DO7\_ADC012\_IN2\_PA2 0.000000
* > DO8\_ADC012\_IN3\_PA3 3.300000
* > ADC01\_IN4\_PA4 0.000000
* > ADC01\_IN5\_PA5 0.000000
* > ADC01\_IN6\_PA6 0.000000
* > ADC01\_IN7\_PA7 0.000000
* > ADC01\_IN8\_PB0 0.000000
* > ADC01\_IN9\_PB1 0.000000
* > DO1\_ADC012\_IN10\_PC0 0.000000
* > DO2\_ADC012\_IN11\_PC1 0.000000
* > DO3\_ADC012\_IN12\_PC2 0.000000
* > DO4\_ADC012\_IN13\_PC3 0.000000
* > ADC01\_IN14\_PC4 0.000000
* > ADC01\_IN15\_PC5 0.000000
* 测试开关量输入pe2引脚电平，可以测量EVB芯片引脚电平确定接口输出是否正常
* Cmd> di pe2
* > KEY1\_PE2 level high
* 测试pwm占空比，目前只测试TIMER4的通道为1 2 3通道，50为占空比
* Cmd> pwm 1 50
* <I>[ 349188]
* > PWM1\_IN0 输出 50

后期还在增加个功能测试