基于新型制冷剂R290的热管理台架系统控制方案

# 一、方案概况

基于新型制冷剂R290的热管理台架系统控制由PC端的上位机与MCU端的下位机组合完成，实现对台架上压缩机、三通阀等元件的控制与数据读取。台架控制方案的示意图见图1。

其中，下位机根据对应的LIN通讯协议对LIN通讯元件进行控制，并输出PWM波调节水泵、散热器风扇等元件的转速。下位机同时循环读取各元件的状态，与上位机通过串口通信，上位机在获取数据后通过解析显示。操作员通过观察和操作上位机向下位机发送指令实现对台架的控制。

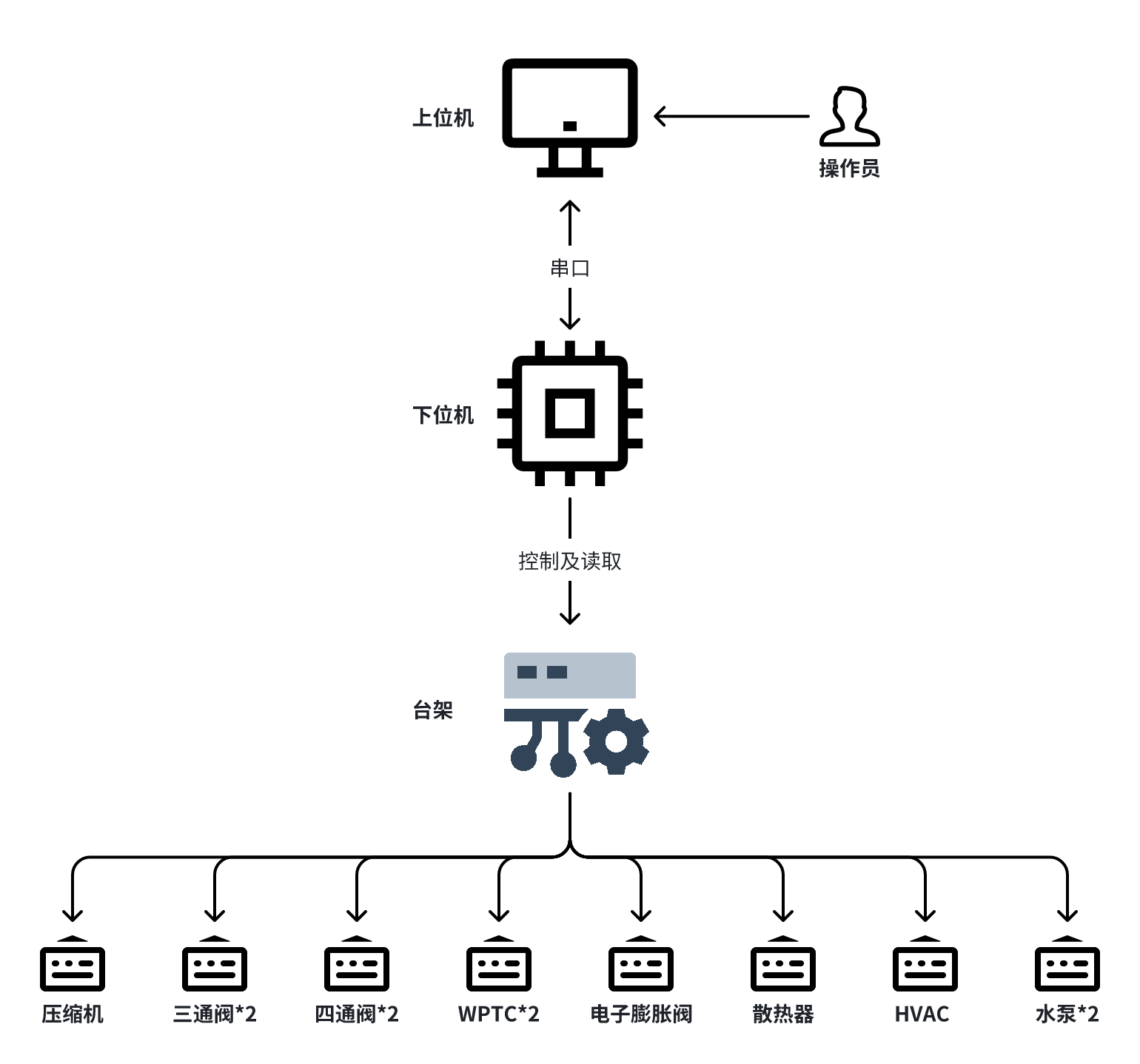


图1 台架控制方案示意图

# 二、下位机方案及进展

## 2.1下位机硬件

主控芯片使用国产车规级MCU芯片云途 YTM32B1ME0。暂时使用云途公司的评估板 YTM32B1ME0 -Q144 评估板进行功能的开发。

EVB板载资源如下：

* EVB MCU：YTM32B1ME0-LQFP144
* EVB 供电：12V/USB-5V 可选
* MCU 供电：5V/3.3V 可选
* 6 通道板载 LED 控制
* 2 通道按键输入
* 4 通道 CAN 通讯接口
* 4 通道 LIN 通讯接口
* 2 通道电位计/模拟量输入

其中MCU（YTM32B1ME0-LQFP144）片上资源如图2所示，能够满足对台架的控制需求。

|  |  |
| --- | --- |
|  | 评估板 |
| 图2 MCU片上资源方框图 | 图3 EVB板载资源示意图 |

## 2.2 下位机软件

MCU配置通过云途公司提供的配置工具YT CONFIG TOOL。通过该工具对MCU的资源进行基础配置。

使用 Visual Studio Code 进行代码编辑，CMake构建项目，JTAG接口烧录。在 VS Code 中进行Debug。

|  |  |
| --- | --- |
|  |  |
| 图4 MCU配置工具YT CONFIG TOOL | 图5 在VS code中后续开发 |

EVB评估板及J-Link烧录器如图6所示：

|  |  |
| --- | --- |
|  |  |
| 图6 EVB评估板及J-Link烧录器 | |

### 2.2.1 MCU功能配置

#### 2.2.1.1 时钟配置

使用默认时钟配置，外部高速晶振提供三个主时钟频率（CORE\_CLK、FAST\_CLK、SLOW\_CLK）分别为120MHz、120MHz、40MHz。经过分频器、锁相环等后MCU主频为120MHz。时钟树配置如图7所示：

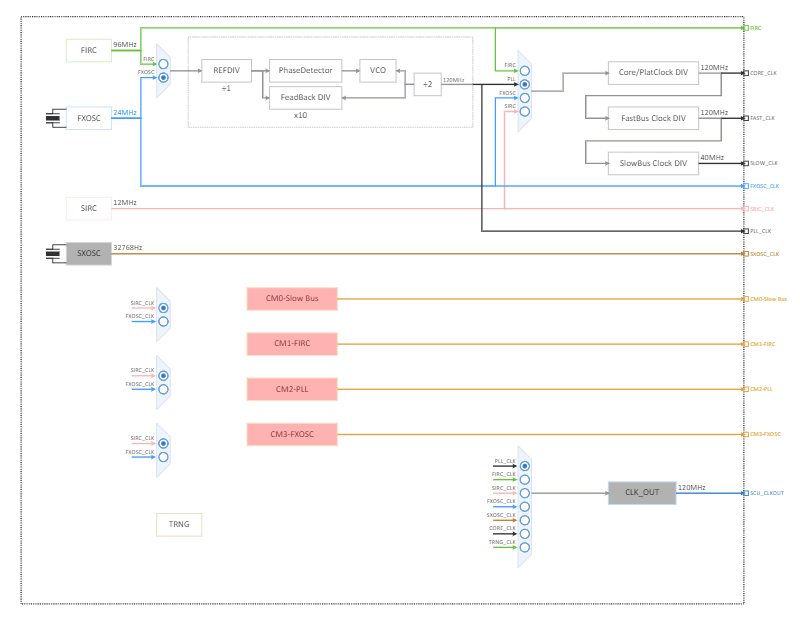


图7 时钟树配置

#### 2.2.1.2 串口配置

通过片上外设 通用异步收发器(Universal Asynchronous Receiver/Transmitter) 实现UART串口与上位机通信，实现Debug中的调试、输出信息、传输数据的功能，是开发过程中不可缺少的步骤。

UART功能的引脚配置表及协议参数如下：

表1 UART引脚配置

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能** | **IO** | **MCU Pin NO.** | **Direction** | **Interrupt config** | **Interrupt Status** |
| UART-RXD | PTA\_8 | 144 | Input | 接收中断+空闲中断 | No clear |
| UART-TXD | PTA\_9 | 143 | Output | 发送中断 | clear |

* 波特率：115200 bit/s
* 无奇偶校验
* 停止位：1
* 字长：8
* 传输类型：中断传输
* 接收方式：环形缓冲区

为了优化调试和数据串口输出的目的，移植了开源printf库，实现了线程安全的类PC端的输出效果，见图8。

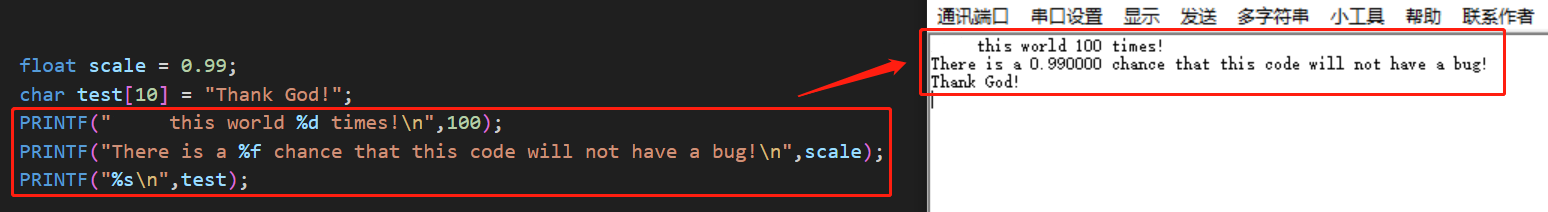


图8 移植printf库串口输出效果

串口接收数据的方式使用接收中断+空闲中断实现，使用接收中断获取单字节数据并存入环形缓冲区中，在空闲中断中向主任务发送信号量表明传输结束。

#### 2.2.1.3 定时器

通过对定时器的设置，可以实现PWM输出和输入捕获，高级定时器eTMR引脚配置见表2。

表2 高级定时器eTMR引脚配置

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **功能** | **IO** | **MCU Pin NO.** | **eTMRx\_CHx** | **CLK\_SRC** | **Prescaler** | **Align\_Mode** |
| PWM output | PTC\_0 | 53 | eTMR1\_CH6 |  | 60 | 向上计数 |
| Input capture | PTB\_12 | 98 | eTMR0\_CH0 |  | 60 | 双边捕获 |

计数器周期计算方法如下：



根据水泵说明书要求的：水泵的PWM信号频率额定值为100Hz。设置分频Prescaler为120，则​为1MHz，产生100Hz的PWM波所对应的eTMR计数器周期值Period为10000，对应的计数器周期值MOD为9999。这个大小小于int16的值（16位eTMR只用15位计时32767），可以使用。

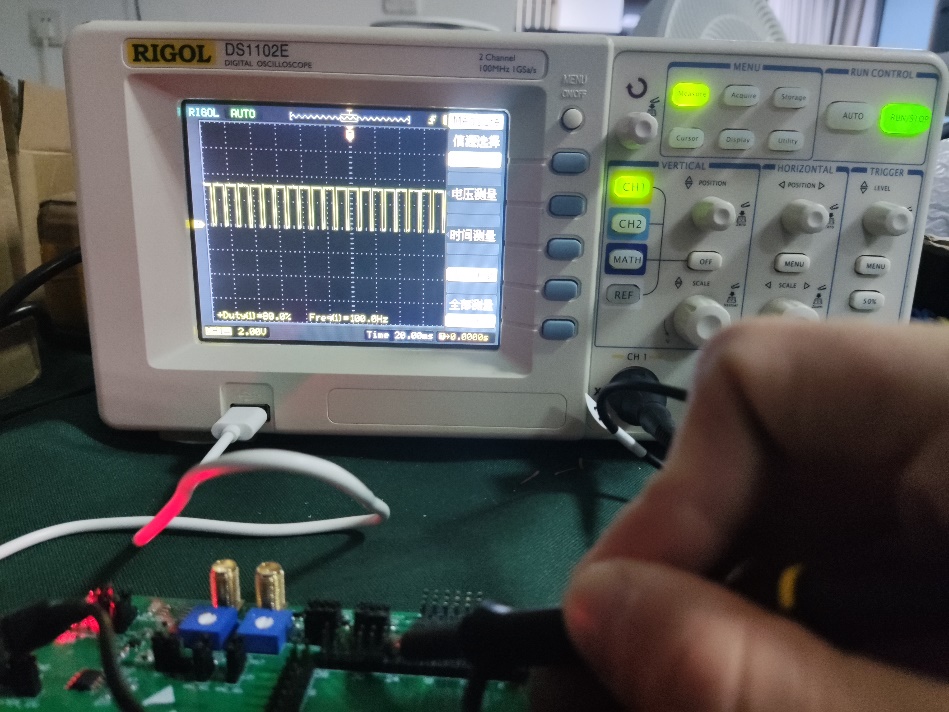


图 9 示波器显示生成的PWM波

为完成脉冲捕获，使用eTMR0的channel0配置了输入捕获功能。其中对于定时器的时钟计算方法同PWM。计算捕获周期的方法为：



在测试过程中，将板子上的PWM输出直接连接到输入捕获引脚上，计算得到的结果通过串口输出结果如下：

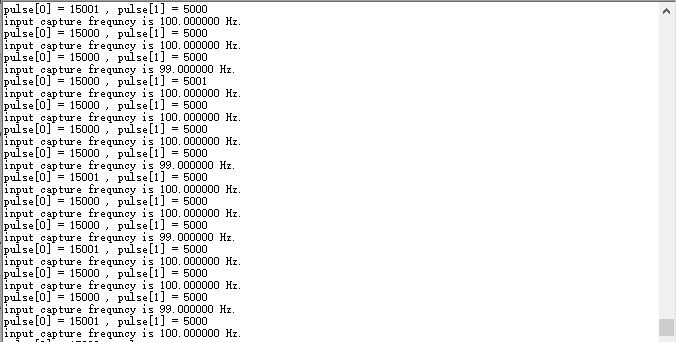


图 10 串口显示捕获的PWM波频率

#### 2.2.1.4 LINFlexD

台架上的大部分控制元件与下位机间的通讯方式是LIN通讯，故而在MCU中配置LIN通讯功能。值得注意的是，YTM32B1ME0芯片中继承了4个片上外设LINFlexd，外界LIN收发器可以快速配置LIN通讯功能，加快了开发。LINFlexD 控制器是芯片的一个 IP，支持 Local Interconnect Network (LIN) 协议，可作 LIN 主机或作 LIN 从机。

表3 LINFlexd引脚配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能 | IO | MCU Pin NO. | Direction | CLK\_SRC |
| LINflex1 | PTC\_8(RX)  PTC\_9(TX)  PTA\_7(SLEEP\_EN) | 81  80  83 | Input  Output  Output |  |

|  |  |
| --- | --- |
|  |  |
| 图10 评估板飞线示意图 | 图11 三通阀LIN测试接线图 |

配置LIN传输协议后，注册中断回调函数，在回调函数中添加处理的内容。对LIN功能进行测试时使用了三通阀进行通讯，发送移动帧0x35指令使三通阀运转，随后发送读取帧0x32指令读取三通阀的开度，最后通过串口在上位机上进行显示，见图12。

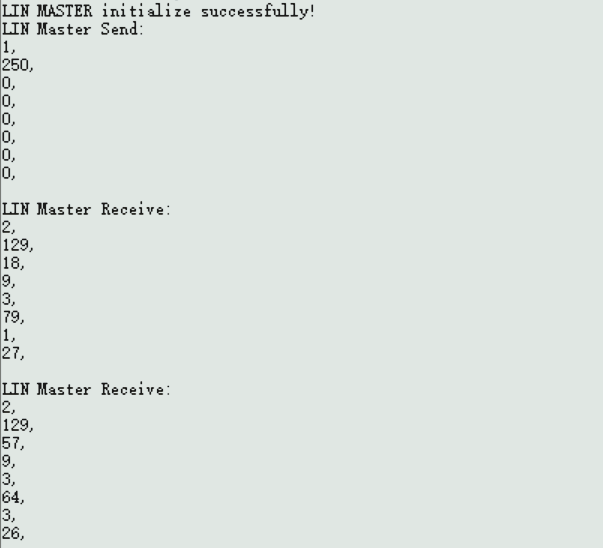


图12 LIN通讯串口反馈

### 2.2.2 FreeRTOS操作系统

项目中涉及到一些实时性能要求，例如控制系统、元件数据采集等。FreeRTOS是一个实时操作系统，其实时调度机制能够提供可预测和有保障的任务调度，确保系统能够及时响应各种实时事件，满足项目的实时性需求。

另一方面任务的管理对于系统的稳定性和性能至关重要。FreeRTOS提供了强大的任务管理机制，能够有效地划分和调度多个任务，使得系统能够更有效地利用处理器资源，提高任务并发性。这对于我们项目中可能涉及到的复杂任务处理是至关重要的。针对不同任务之间需要协同工作和共享信息。FreeRTOS提供了多种同步和通信机制，包括信号量、消息队列、互斥锁等，使得任务之间能够安全地共享资源，实现更复杂的系统功能。FreeRTOS还提供了适应多任务环境的中断处理机制，确保中断服务程序能够与系统中的其他任务协同工作，提高系统的可靠性和可维护性。

综上所述，在开发过程中为芯片移植了FreeRTOS操作系统（FreeRTOS Kernel V10.4.3）。源码存放在项目rtos文件夹中，内存管理使用了heap\_4.c方式。FreeRTOSConfig.h参考了FreeRTOS官方给出的文件进行了修改。

任务创建使用动态内存创建方法，在main.c文件中创建了一个任务（AppTaskCreate），用于创建主程序任务。项目中的主要任务有四个：

* Task\_main：在获取串口中断释放的信号量后对数据帧进行解析，如果满足正确的帧格式则进行解析，根据指令码将数据发送给消息队列或激活事件，唤醒对应的任务。在未收到信号量的时候阻塞任务。
* Task\_00：在收到Task\_main的事件激活时被唤醒，清除Task\_02使能事件，中止Task\_01的发送，中止台架的运行。未收到激活事件时被阻塞。
* Task\_01：在收到Task\_main向消息队列中写入的消息后被唤醒，对数据帧进行解包，根据数据向控制元件进行控制及调整，在发送控制数据结束后，唤醒Task\_02。在消息队列中未收到消息的时候阻塞任务。
* Task\_02：在收到Task\_01的事件激活时被唤醒，获取台架的状态信息，对状态信息进行组包，通过串口向上位机发送。在被清除使能事件或未收到激活事件时被阻塞。

表4 任务功能

|  |  |  |
| --- | --- | --- |
| 任务 | 任务指令码 | 任务主要功能 |
| Task\_main | - | 获取指令激活对应任务 |
| Task\_00 | 0x00 | 中止台架运行，停止数据传输 |
| Task\_01 | 0x01 | 根据收到的数据控制台架元件 |
| Task\_02 | 0x02 | 向上位机传输台架状态信息 |

### 2.2.3 下位机运行逻辑

为实现下位机与上位机的正常通信，需要对上位机和下位机的运行逻辑进行设计。下位机的运行逻辑流程图如图13所示，在FreeRTOS操作系统中运行四个任务，其中主任务与串口中断通过信号量同步，串口接收到一帧信息后通过信号量通知主任务，主任务进行解析获取指令，通过消息队列或事件组的事件激活唤醒对应的任务。唤醒的任务完成自身的主要功能并通过事件激活或阻塞其他任务。

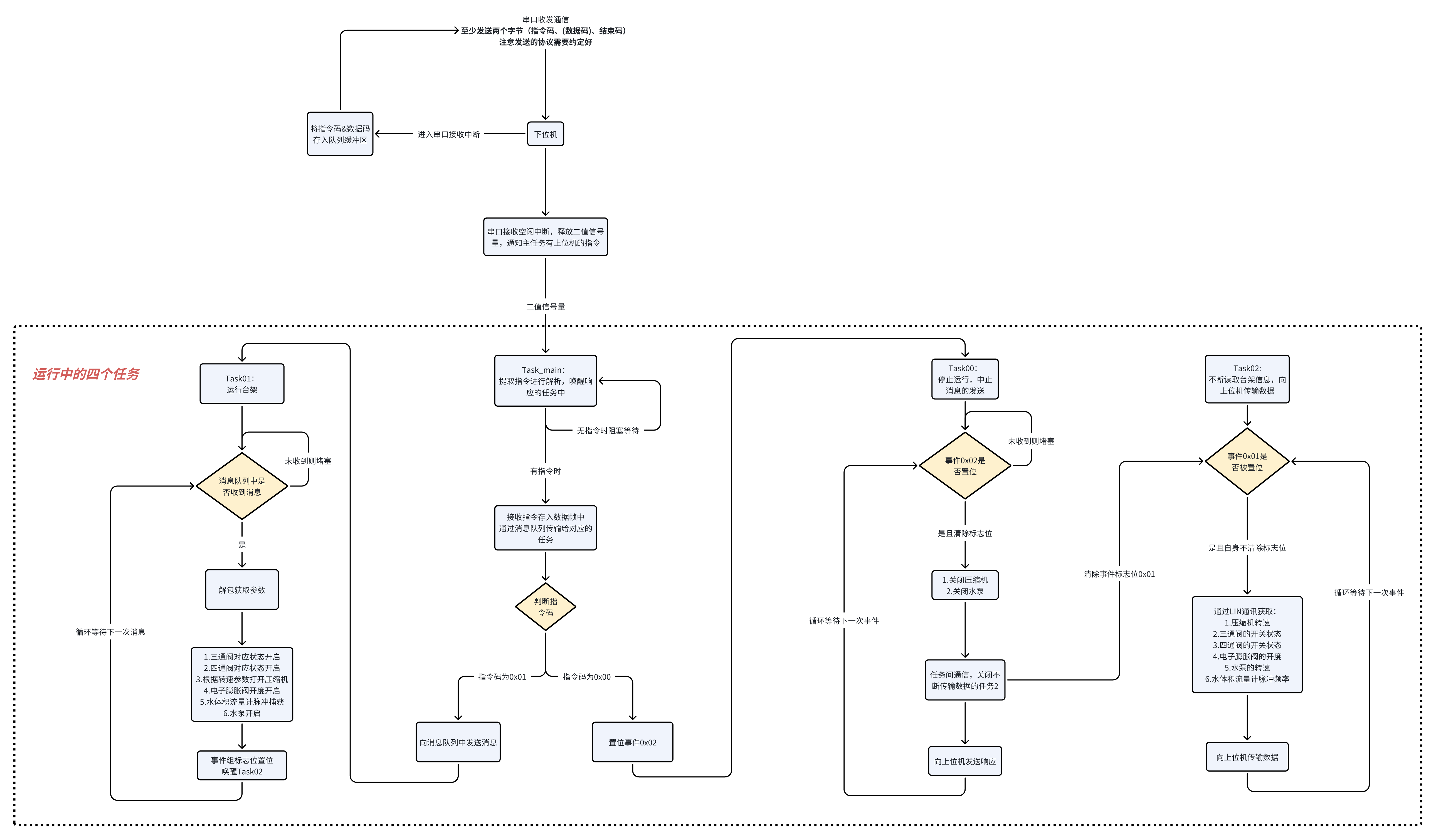


图13 下位机运行逻辑流程图

针对元件的控制编写对应的API，其声明如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 压缩机通讯 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 更改压缩机的转速，变相等于开启压缩机 \*/

uint8\_t Compressor\_Set\_Speed(uint16\_t speed, uint16\_t limit\_power);

/\* 关闭压缩机 \*/

uint8\_t Compressor\_Shutdown(void);

/\* 获取压缩机状态 \*/

uint8\_t Compressor\_Get\_info(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 电子膨胀阀通讯 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 更改电子膨胀阀的开度 \*/

uint8\_t Expansion\_valve\_Set\_Open(uint16\_t open);

/\* 获取电子膨胀阀状态 \*/

uint8\_t Expansion\_valve\_Get\_info(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 三通阀通讯 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 更改三通阀的比例开度 \*/

uint8\_t Three\_way\_valve\_Set\_Open(uint8\_t instance, uint8\_t pos);

/\* 获取比例三通阀状态 \*/

uint8\_t Three\_way\_valve\_Get\_info(uint8\_t instance);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 四通阀通讯 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 更改四通阀的开关状态 \*/

uint8\_t Four\_way\_valve\_Set\_Open(uint8\_t instance, uint8\_t mode); // mode 只有两种取值

/\* 获取比例三通阀状态 \*/

uint8\_t Four\_way\_valve\_Get\_info(uint8\_t instance);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WPTC通讯 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 开启WPTC加热，并设置温度 \*/

uint8\_t WPTC\_Set\_Temperature(uint8\_t instance, uint8\_t temperature, uint8\_t heat\_power);

/\* 获取WPTC状态 \*/

uint8\_t WPTC\_Get\_info(uint8\_t instance); // 输入的是第instance个WPTC，instance = 1 or 2

以上API分别在Task00、Task01、Task02中被调用，用于控制元件和获取元件的状态数据。实现中为防止资源的使用冲突，在LIN的发送与接收时添加了互斥锁。

# 三、上位机方案及进展

## 3.1上位机开发环境

上位机使用C# 开发，开发环境为Visual Studio 2022。

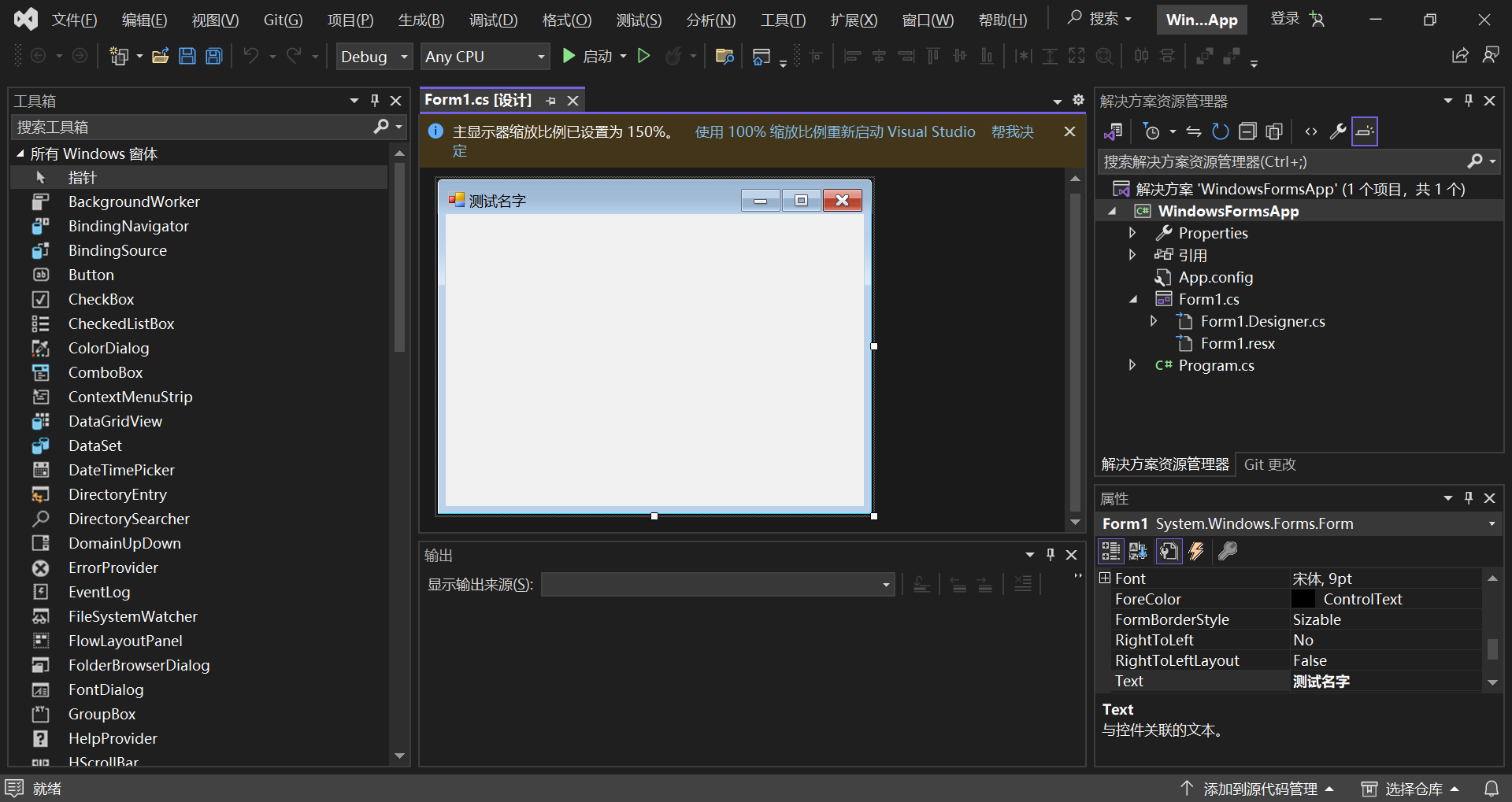


图14 上位机开发环境界面

## 3.2上位机与下位机通讯数据帧定义

上位机与下位机传输时，由于指令需求的不同，数据帧是非定长的，故在字节流中需要判断出每一帧的起始和结束位置。

串口发送帧格式规定：由 帧头 + 指令码 + 数据码 + 数据长 + 帧尾 组成。

约束串口通信的帧头帧尾格式如下：

* 帧头：0xFE
* 帧尾：0xFF



图15 串口数据通讯帧定义图

上位机与下位机在通讯过程中根据该定义进行数据的组包和解包。

## 3.3上位机运行逻辑与界面

上位机的运行逻辑设计如下：不断获取串口数据进行解包，根据数据帧的定义进行解包并进行显示，刷新显示框内的内容以及图表。操作员通过点击上位机界面上的按钮向下位机发送指令，再通过观察数据框和图标得到反馈和台架的状态数据。

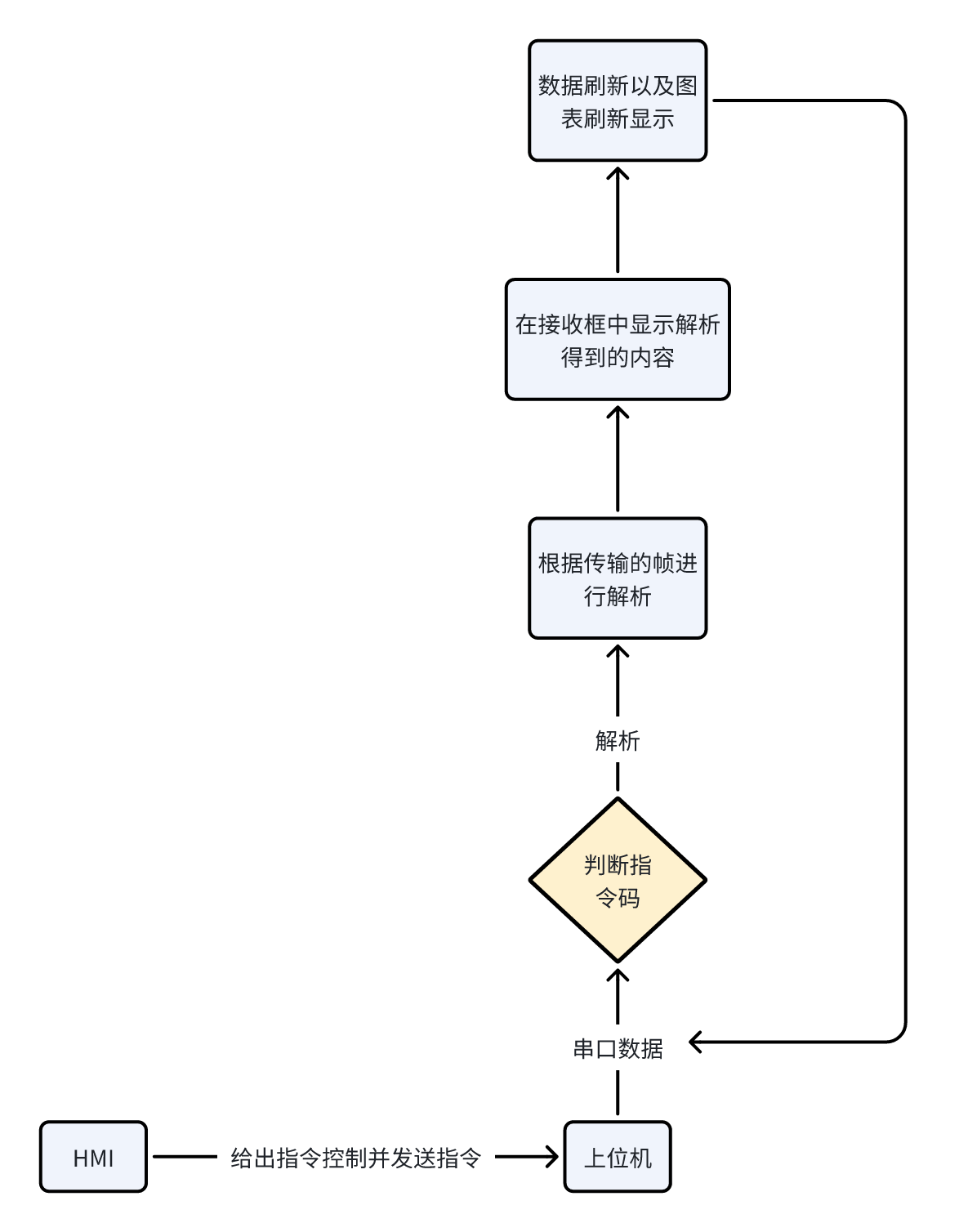


图16 上位机运行逻辑

上位机的初版界面如下所示：

控制元件的所有状态在界面中实时显示，并能够通过选择框输入进行控制，下位机的调试语句和控制元件的状态数据同样也会在左边的显示框中显示。



图17 上位机界面