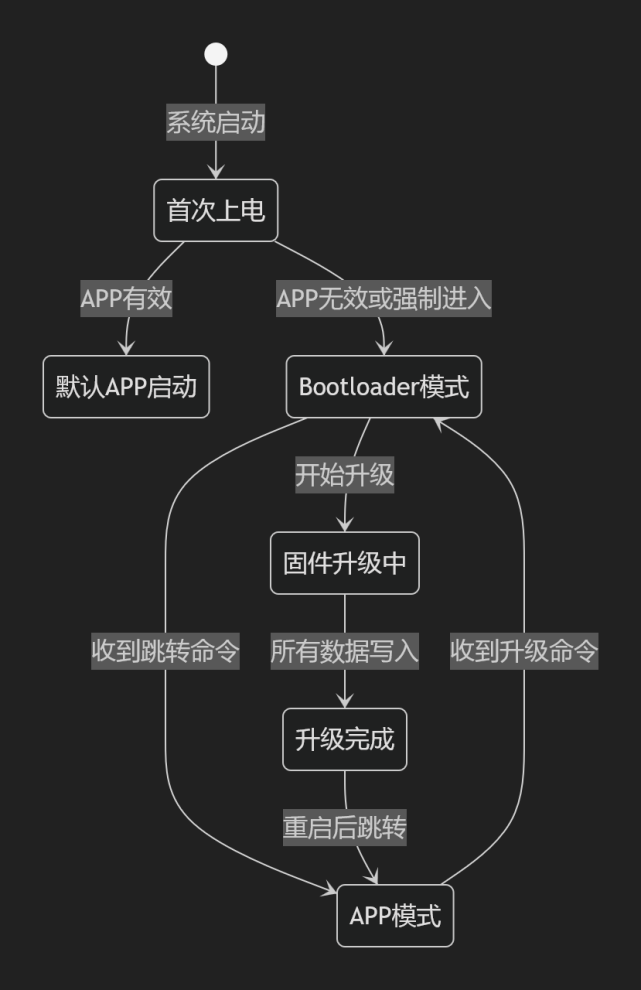
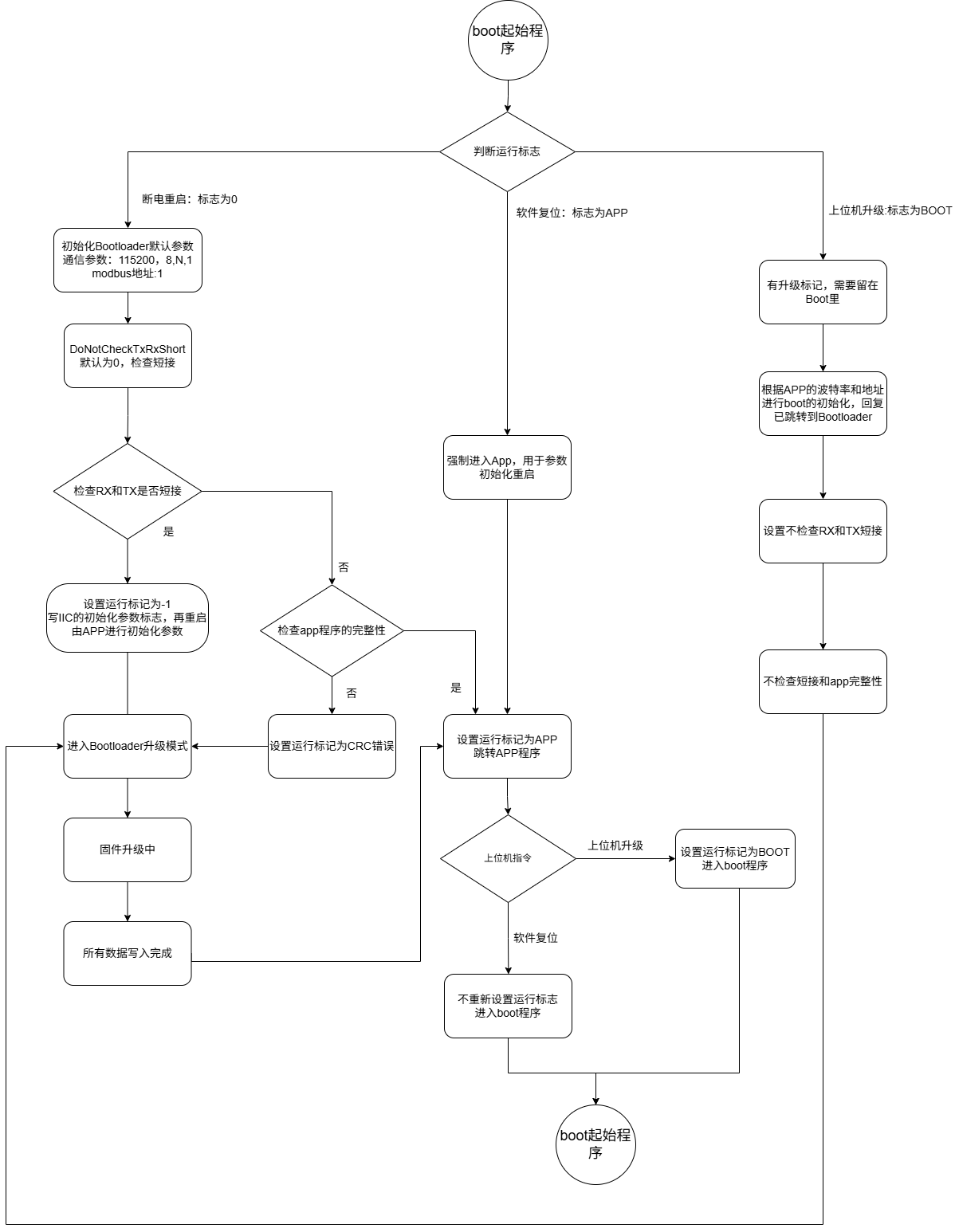
# 总体流程

# bootloader程序

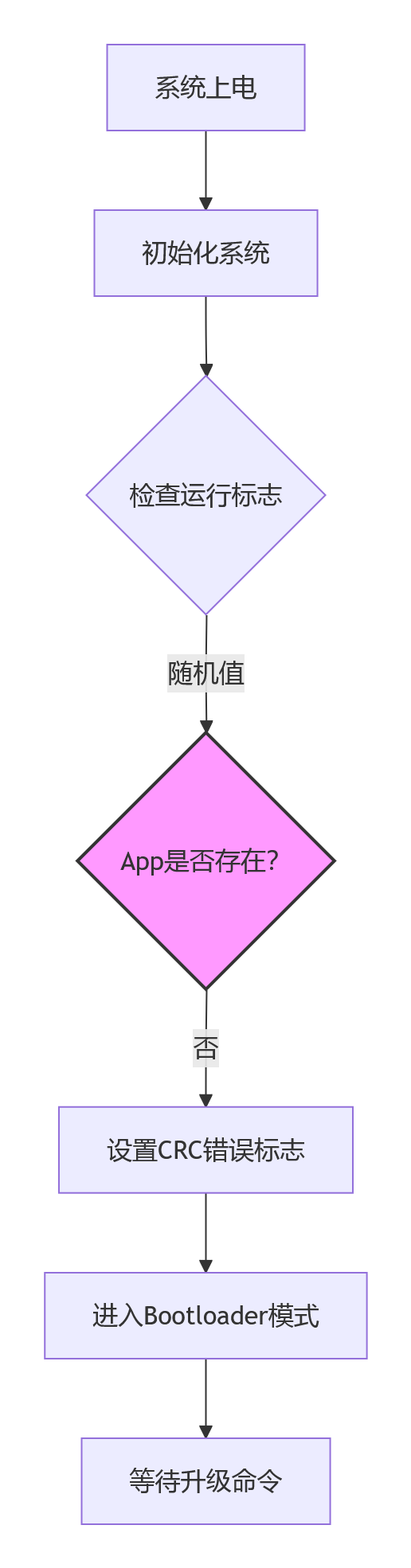
## 首次上电使用



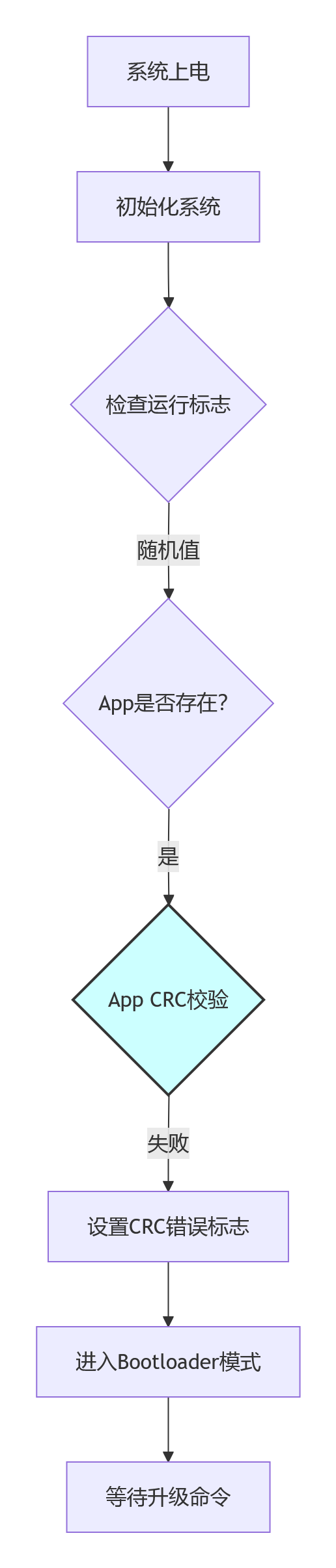


## 各种场景的流程

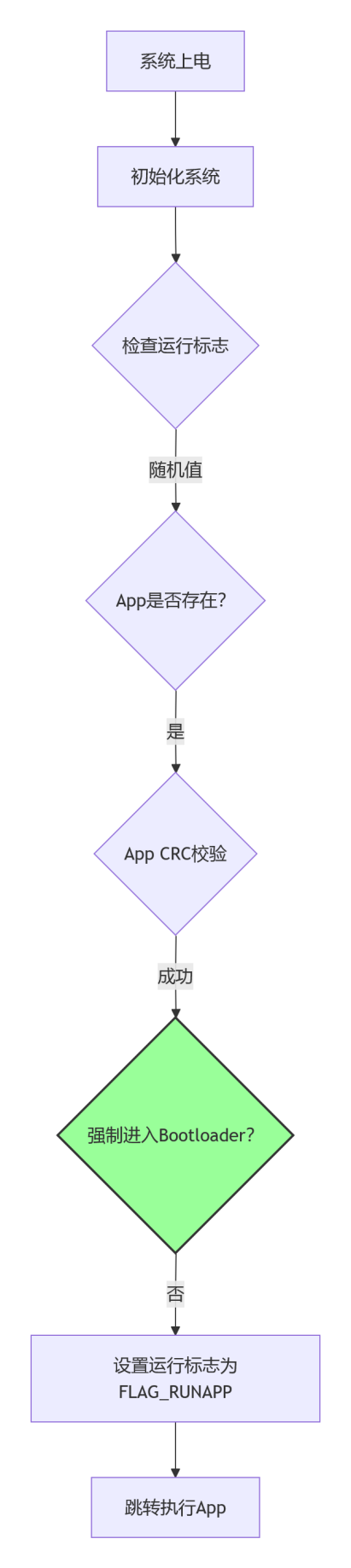
### 上电且无有效App（场景1）



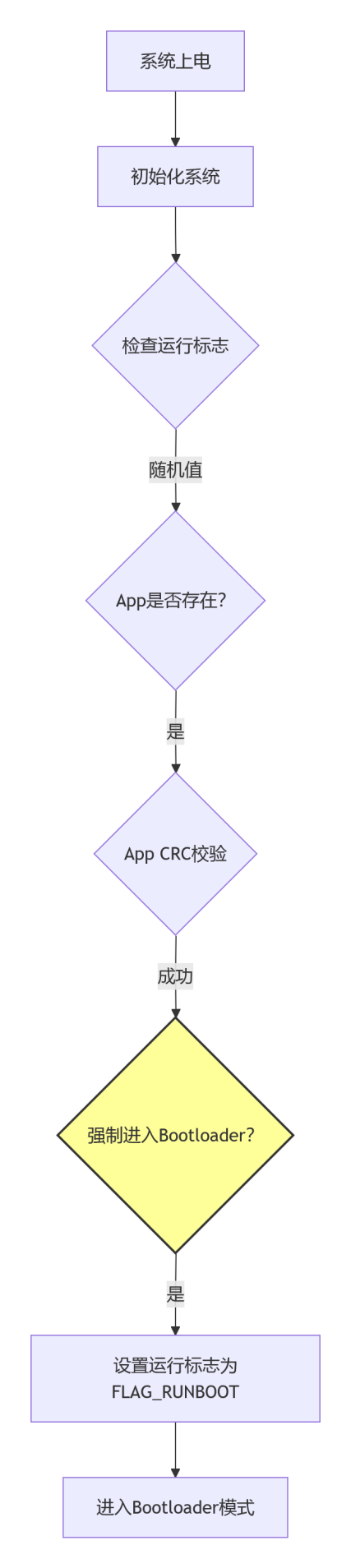
### 上电且App不完整或校验失败（场景2）



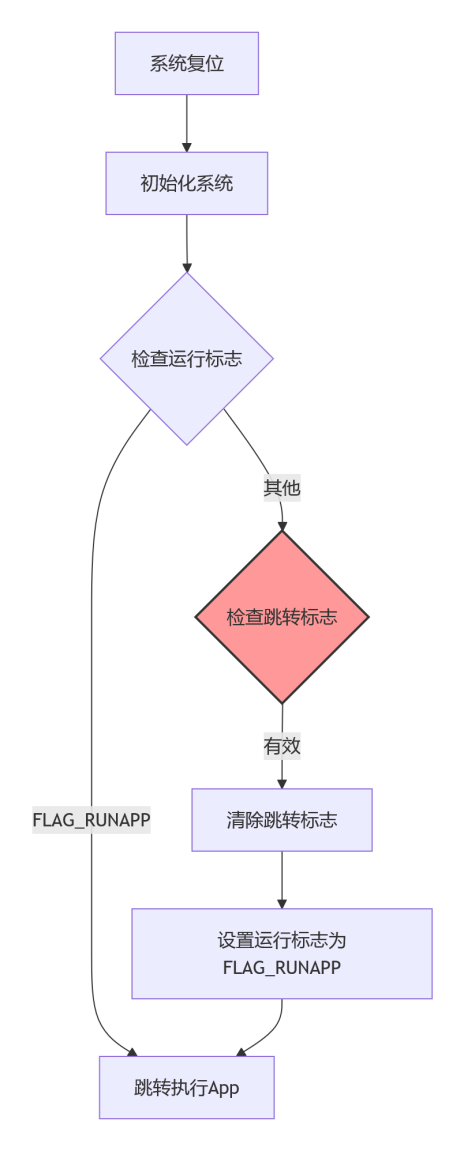
### 上电且App完整有效且无强制标志（场景3）



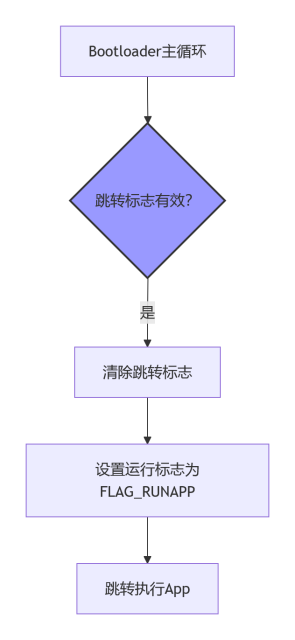
### 上电且App完整但强制进入Bootloader（场景4）



### 复位且跳转标志有效（场景5）



### 正常工作时手动设置跳转标志（场景6）



# Bootloader 启动与升级流程详解

## 引言

该Bootloader设计用于在目标MCU上实现固件升级和可靠启动。核心思想是：

通过运行标志决定启动模式（App或Bootloader）（在热启动复位时候通过SRAM进行判断）

通过App完整性（CRC）校验保证代码安全（冷启动在断电时候通过读取内部flash的app的size和crc校验值进行判断）

通过升级模式支持上位机远程固件写入与校验

通过跳转标志支持动态切换执行App或Bootloader

## 关键变量与标志说明

| **变量/标志** | **地址** | **值** | **含义** |
| --- | --- | --- | --- |
| RunAPP\_Flag | 0x20000000 (RAM) | 0x55555555 (FLAG\_RUNAPP) | 强制跳转运行App |
|  |  | 0xAAAAAAAA (FLAG\_RUNBOOT) | 强制停留Bootloader，等待固件升级 |
|  |  | 0xFFFFFFFF (FLAG\_CRC\_ERROR) 或其他 | App完整性校验失败标志，停留Bootloader等待升级 |
| Boot\_Para | 0x20000004 (RAM) | 波特率+ID等参数 | 升级过程相关参数 |
| 跳转标志 | RAM或特定寄存器 | 1（跳转App） | 标记当前需要跳转运行App |

## 启动流程逻辑详细说明

### 上电或复位初始化阶段

1、关闭中断，设置中断向量表地址指向Bootloader。

2、系统时钟和延时初始化。

3、读取 RunAPP\_Flag 以判断启动模式。

### 根据 RunAPP\_Flag 判断启动模式

| **RunAPP\_Flag 状态** | **处理逻辑** |
| --- | --- |
| FLAG\_RUNAPP（0x55555555） | 直接跳转执行App，无条件运行。适用于重启后强制运行App的场景。 |
| FLAG\_RUNBOOT（0xAAAAAAAA） | 初始化外设（串口、Modbus、I2C等），回复上位机进入Bootloader状态，进入升级等待模式。 |
| 其他（含CRC错误标志等） | 进入App完整性检测流程，判断App是否有效。 |

### App完整性检测

调用函数 Bootloader\_CheckApp()：

检查App起始地址是否合理（首地址有效）

计算App区域CRC与存储CRC比对。

结果分两种：

| **结果** | **处理逻辑** |
| --- | --- |
| App完整且校验通过 | 如果没有强制进入Bootloader标志，设置 RunAPP\_Flag = FLAG\_RUNAPP 并跳转执行App。 |
| App不完整或CRC错误 | 设置 RunAPP\_Flag = FLAG\_CRC\_ERROR，停留Bootloader，等待固件升级。 |

### 强制进入Bootloader检测

通过外部条件（如I2C EEPROM标志、开关状态）调用 Bootloader\_Check\_Force() 判断是否强制进入Bootloader。

如果强制进入：

设置 RunAPP\_Flag = FLAG\_RUNBOOT。

初始化外设，进入升级模式。

如果不强制，且App完整，则跳转App执行。

## Bootloader 升级等待及处理流程

### 停留在Bootloader时刻意等待

当 RunAPP\_Flag 是 FLAG\_RUNBOOT 或 FLAG\_CRC\_ERROR 时，程序不跳转App。

初始化UART、Modbus、DMA等外设，开启通信。

进入主循环，调用 ModBus\_Slave\_Process() 监听上位机命令。

### 上位机交互支持的命令

| **命令功能** | **作用描述** |
| --- | --- |
| 读取硬件ID | 识别目标设备硬件信息 |
| 读取/写入App程序长度 | 升级前准备App大小 |
| 读取/写入App CRC校验 | 校验升级数据完整性 |
| 擦除App区域Flash | 擦除旧固件空间 |
| 写入程序Flash | 分块写入固件数据 |
| 校验升级 | 比较写入CRC和预期CRC |
| 跳转运行App | 完成升级后启动新的App |
| 跳转回Bootloader | 保持升级模式或重启升级 |

### 固件写入流程简述

上位机发送升级信息（大小、CRC、硬件ID等）。

Bootloader解密数据块，调用 Bootloader\_ProgramBlock() 写入Flash。

写入过程中可能返回错误，反馈上位机。

写入完成后，上位机发校验命令确认写入正确。

校验成功后，发送跳转命令启动App。

## 跳转到App执行

### 通过跳转标志控制跳转

Bootloader主循环不断检查跳转标志。

当上位机发送跳转App命令，Bootloader设置跳转标志。

主循环检测到跳转标志后：

清除跳转标志。

设置 RunAPP\_Flag = FLAG\_GO\_APP

调用 Bootloader\_RunAPP() 跳转执行App。

## 总结各种典型启动情况及其处理

| **场景描述** | **详细情况** | **处理流程** | **备注** |
| --- | --- | --- | --- |
| 上电且无有效App | App区域为空或Flash无固件 | CRC检测失败，进入Bootloader，等待升级 | 典型出厂或Flash擦除后状态 |
| 上电且App不完整或校验失败 | App程序有残缺或CRC校验不匹配 | 设置CRC错误标志，进入Bootloader，等待上位机升级 | 防止运行损坏固件，确保安全 |
| 上电且App完整有效且无强制标志 | 正常工作状态 | 跳转执行App | 设备正常运行 |
| 上电且App完整但强制进入Bootloader | APP通信异常上位机不能使得程序进入Boot模式 | 进入Bootloader，等待上位机升级 | 用于紧急升级或维护 |
| 复位且跳转标志有效 | 上位机刚刚发送跳转命令，Bootloader收到标志 | 立即跳转执行App | 无需再次检测App完整性 |
| 正常工作时手动设置跳转标志 | 设备在线升级完成后准备跳转 | 通过标志跳转App | 升级流程正常完成 |

## 设计思路与优势

**安全性**：通过CRC校验阻止错误或损坏的App运行。

**灵活性**：支持强制启动Bootloader与App，适应多种维护场景。

**远程升级**：Modbus协议实现远程升级，无需物理接入。

**易维护**：状态标志清晰，跳转逻辑简单，方便调试与扩展。

**扩展性**：支持多种升级命令及数据加密保护。

## 代码关键流程伪代码示例

void main(void)

{

DisableIRQ();

InitSystem();

RunAPP\_Flag = ReadRunAppFlag();

if (RunAPP\_Flag == FLAG\_RUNAPP) {

JumpToApp();

}

else if (RunAPP\_Flag == FLAG\_RUNBOOT) {

InitPeripherals();

WaitForUpgrade();

}

else {

if (CheckAppIntegrity() == OK) {

if (CheckForceBootloader()) {

InitPeripherals();

WaitForUpgrade();

} else {

RunAPP\_Flag = FLAG\_RUNAPP;

JumpToApp();

}

} else {

RunAPP\_Flag = FLAG\_CRC\_ERROR;

InitPeripherals();

WaitForUpgrade();

}

}

while(1) {

ProcessModbus();

if (JumpFlagSet()) {

ClearJumpFlag();

JumpToApp();

}

}

}