# 目 录

# 第 1 章 绪论

## 1.1 课题研究的背景、目的和意义

石油是人们生活中不可或缺的资源，成品油零售点更是随处可见。随着成品油零售行业的发展，加油站的网络信息化建设也变得越来越重要。销售公司需要实时、准确的掌握成品油的“进、销、存”信息，及时了解市场的动态，并且要实现数据共享，以便总部数据中心根据反馈信息进行整合统计，对下一步的成品油供给做出调整，合理的分配资源，做到不漏空、不浪费。目前，国内的加油站正处于由独立、局部管理向互联管理转变的时期，正逐步形成全国范围加油站互联，使得成品油的供给和消费情况及时准确的反馈与共享，以对加油站操作的各个环节进行有效的监控[1]。

加油站需要集成所有的前庭设备，如加油机、液位仪等进行管理。但是在成品油零售行业，不同设备生产厂商都有自己的设备协议，这就导致了加油设备协议和接口的多样化，造成了维护困难以及加油设备的互操作性差，石油零售商在选择、更换加油设备时受制于生产厂商[2]。为了使不同厂商的设备间更容易集成，国际加油站标准论坛组织制定了统一的 IFSF（International Forecourt Standards Forum）协议来规范加油设备生产商。IFSF 协议规定了前庭设备之间通信的协议与规范，提高了前庭设备间的互操作性。在加油站设备的互联系统中，IFSF 标准站级架构[3]是一种非常先进的前庭设备连接方式，协议统一，冗余较少，易于添加设备。而由于前庭设备协议的多种多样，国内加油站互联系统采用最多的是将不同标准的加油设备连接到前庭控制器上的传统架构方式。前庭控制器通常包含针对不同厂商协议的协议转换模块，将复杂多样的协议转换成 IFSF 协议，然后再实现网络通信。这一方式有着许多的不足，不仅增加了协议转换模块，提高了复杂程度，而且使得前庭设备受控于前庭控制器，增加了故障点。

采用 IFSF 架构是石油产业中加油站管理的一种发展趋势，基于 IFSF 协

议的加油机的研究也变得非常重要。传统的加油机基于多种协议，需要通过协议转换将多种加油机协议转换成统一的 IFSF 协议，以串口通信的方式进行与前庭控制器的连接，造成了浪费；而基于 IFSF 协议的加油机不需要进行协议转换，符合 IFSF 架构要求，加油站也因前庭设备基于 IFSF 协议可以共享数据而有着很大的优势，所以，基于 IFSF 协议的加油机必将取代传统的加油机。本课题提出的是基于 IFSF 协议的加油机控制装置，并且以 Internet 网络通信方式直接与后台以及总部数据中心连接，代替了传统的串行通信，不需要进行通信协议转换，还有利于加油站信息化建设，可以对加油站的各种作业进行更有力地监控。目前，加油站互联系统正由前庭控制器连接方式向 IFSF 架构方式转变，基于 IFSF 协议的加油机控制装置有着更大的优势，具有非常高的可行性和先进性。

## 1.2 国内外研究现状综述

近年来，随着世界石油石化工业竞争的日益激烈，国内外各大石化公司以科学管理、降低成本和提高经济效益作为发展目标，加油站管理尤为重要。很多大的石油公司逐步向着网络自动化管理方向发展，以提高运营效率。

国外的加油站设备先进，加油站管理也达到了很高的水平，形成了高效的管理网络系统，并向自动控制方向发展，加油站的技术含量非常高，自助加油站也变得普遍，由于采用了加油站网络系统，对加油的各个环节也能实时监督，避免泄露、油品跑冒等问题，大大减少了损失。我国的加油机产业起步较晚，加油站的自动化管理水平比发达国家落后，加油站数量很多，加油设备却陈旧，难以使加油站网络信息化；但是随着国产加油机科技含量的不断增加，国内加油站管理水平的不断提高，加油站互连系统也在不断健全。

国内外前庭设备厂商为了能够实现自动实时采集前庭设备数据，各自定义了前庭设备的通讯协议，这导致了协议的多样化，建立统一标准的协议成了各大石油公司的共同目标，在这种情况下出现的 IFSF 协议就实现了前庭设备的统一。现在有四种加油站互联方式[4]：

1) 中控系统方式：设备的控制功能由中控机实现，其它功能由集线器完

成；

2) 卡机联动方式：加油机的控制是在加油机的卡模块上实现的，并通过集线器将加油机与后台计算机相连；

3) 前庭控制器方式：由前庭控制器集成所有设备，然后实现数据传输、协议转换和设备控制；

4) IFSF 架构方式：由于前庭设备基于 IFSF 标准，不需要进行协议转换，可以直接利用软件对前庭设备进行多点控制，并利用 LonWorks 或 TCP/IP 网络完成数据通信。

前两种方式由于协议的不统一，更换维护设备非常困难，而且由集线器实现与后台的连接，第三种方式在前庭控制器中加入协议转换模块并实现网络功能，增加了故障点且浪费了资源，而第四种 IFSF 架构方式克服了前三种方式的缺点，最适用于加油站互联应用。

国外的加油站互联方式有中控系统、前庭控制器和 IFSF 架构方式，而加油机数量庞大却协议多样的国内现有的互联方式有中控系统、卡机联动以及前庭控制器方式。国外很多国家使用液位仪和中控系统进行加油站远程管理，虽然目前我国也建立了金卡系统，但由于国内的现金消费习惯等原因，在非城市地段和跨省应用方面并不顺利，而加油站互联可以有效地弥补这一缺陷。国内加油站管理系统网络普遍采用的是基于主从模式的前庭控制器连接方案，由前庭控制器控制各前庭设备。整体架构如图 1-1：

加油站管理系统在空间上可以划分为室内和前庭两部分[5]：室内包括前庭控制器 FCC（Forecourt Controller），前台销售系统 POS（Point of Sale），后台管理系统 BOS（Back Office System）；前庭设备主要有加油机、液位仪、价格牌等。由于前庭设备协议不尽相同，设备间的互操作性差，管理人员对设备也很难进行统一管理和监控，因此要将 Non-IFSF 协议转换成 IFSF 协议，而很多前庭控制器都包含将协议转换功能，一来可以有效地集成设备，二来可以将多种协议转换为统一的协议并实现网络通信。如上所述，所有的前庭设备均通过前庭控制器来连接后台，一旦前庭控制器发生故障，就会导致后

台和前庭设备失去联系，因此存在着潜在的不可靠运行，增加了管理和维护上的困难。

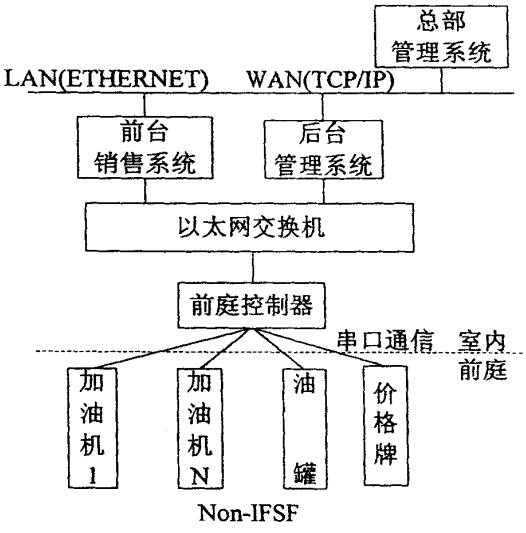



图 1-1 国内加油站管理系统整体架构

Fig.1-1 Architecture of gas station management system in China

而基于 IFSF 架构的互联系统架构如图 1-2 所示。

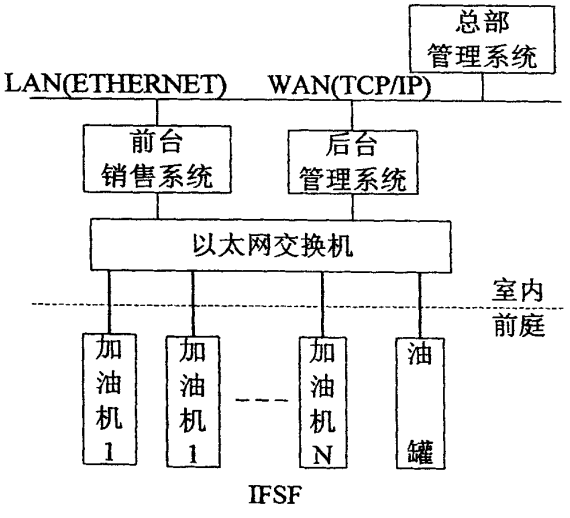



图 1-2 基于 IFSF 协议的管理系统构架

Fig.1-2 Architecture of management system based on IFSF

在此架构中，加油机等设备采用 IFSF 协议，这样就不需要协议转换；没有连接到前庭控制器，这样就避免了单点故障，减少了故障点，使错误排查

更加快捷，消除了轮询延迟，通信速率高，实时性更好，而且管理方便，更容易进行数据共享。在我国加紧加油站网络信息化管理的进程中，采用使用星型拓扑结构的基于 TCP/IP 的 IFSF 站级网络部署[6]，是一种很好的选择，基于 IFSF 协议的加油机的研究也越来越多。

## 1.3 本文研究的主要工作

我国石油消耗巨大，加油站点很多，石油的“进、销、存”资料共享非常重要，IFSF 架构方式中的设备均采用 IFSF 协议，避免了众多私有协议的开发，也避免了由于采用前庭控制器产生的单点故障[3]，是网络信息化发展的必然趋势，研究基于 IFSF 协议的加油机变得非常重要。传统加油机的通信方式通常是串行通信，而本文通过研究基于 IFSF 协议的加油机，设计了一种直接基于网络连接的加油机控制装置，替代了原来的通过串口连接的加油机，使加油机直接与总部系统数据中心通信成为可能，总部中心可以直接监控观察站点加油机的操作，对加油机进行实时、准确的监控。

本文的主要研究工作有：

1) 针对连接到前庭控制器上的需要进行协议转换的 Non-IFSF 加油机存在的弊端，研究了基于 IFSF 协议的加油机控制装置，设计了以 PHILIPS 公司高性能的 ARM7 系列芯片 LPC2378 为微处理器的加油机控制装置，采用基于嵌入式的方法，工作稳定，适合加油站环境工作。

2) 详细研究 IFSF 协议，对 μC/OS-Ⅱ[7]进行深入了解，并了解 TCP/IP 协议栈，实现 IFSF 在 TCP/IP 上的应用。

3) 进行系统测试，实现加油操作和网络通信来对装置进行调试，测试其性能。

## 1.4 本文的章节安排

第一章绪论部分介绍课题的研究背景及意义和发展现状，并在此基础上给出了本文的结构安排；第二章：介绍 IFSF 协议的相关内容，并且介绍了文章中用到的 μC/OS-Ⅱ操作系统及其移植和 TCP/IP 协议栈及其移植；第三章简要介绍加油机控制装置及其采用的硬件，包括加油机加油功能及网络通信功能的硬件实现；

第四章详细阐述了加油机控制装置的软件实现过程；第五章对加油机控制装置（包括硬件与软件）进行系统测试，证明该装置的合理性与可行性；第六章总结论文取得的研究成果，并对进一步的研究工作进行展望。

## 1.5 本章小结

本章讨论了基于 IFSF 协议的加油机控制装置研究的背景与意义，介绍了国内外研究现状，并提出了本文的主要工作和章节安排。

# 第 2 章 IFSF 协议及嵌入式操作系统与协议栈的移植

## 2.1 IFSF 协议及其实现原理

### 2.1.1 IFSF 协议简介

IFSF，即国际加油站前庭设备标准论坛（International Forecourt Standards Forum），是石油零售行业的国际标准组织，是为了在石油行业推广使用标准化的加油设备和加油站互联体系而建立的论坛[8]。由于很多设备制造商都有自己私有的前庭控制器与前庭设备的通信协议和接口，这些私有协议将石油零售商锁定在了特定的设备供应商，而这些设备商提供的设备有时不能满足石油公司对于计算机系统发展变化的要求，而且因为配置的复杂多样，对于度量检定也造成了很大的困难。IFSF 就是在这种情况下于 1993 年成立，代表了很多大的石油公司，如 BP（英国石油公司）、Kuwait（科威特石油）、ExxonMobil（埃克森美孚石油）等。为了实现国际石油零售行业协调前庭设备的互操作性和通信标准的共同目标，帮助提高石油公司加油站的营运效率，IFSF 设计开发了加油站现场通信技术标准[9]。IFSF 并不销售商业产品和服务，是非盈利性组织；其宗旨是鼓励前庭设备商向市场提供由 IFSF 认证的统一的可互操作设备，以供给石油公司使用[10]，使石油公司可以随便选择使用前庭设备。IFSF 规定了一个通用通信接口，通过它前庭设备可以和前庭控制器以及其它网络设备交换信息。IFSF 协议的出现，使得零售商购买更换设备不再局限于厂家，使得前庭设备的维护更加简便。

为了满足设备间的互操作性要求，IFSF 的设计准则有：开放的系统，国际认证，灵活；独立于设备商；购置，安装，测试和支持成本低；网络传输及时可靠。

IFSF 标准分为三部分：管理文档（Management documents）、通信规格（Communication Specifications）和设备应用程序协议（the Device Application protocols）。

IFSF 协议遵守标准 OSI 的七层协议模式，其实现如图 2-1 所示。

OSI 标准模式共七层，分别是物理层、链路层、网络层、传输层、会话层、表

示层和应用层。其中低六层在 IFSF 协议中合为通信层，有两种网络结构，分别采用的是 LonWorks 协议和 TCP/IP 协议；第七层为应用层，是由 IFSF 自己定义的设备协议，也是实现前庭设备间互相通信的最为重要的一层[11]。

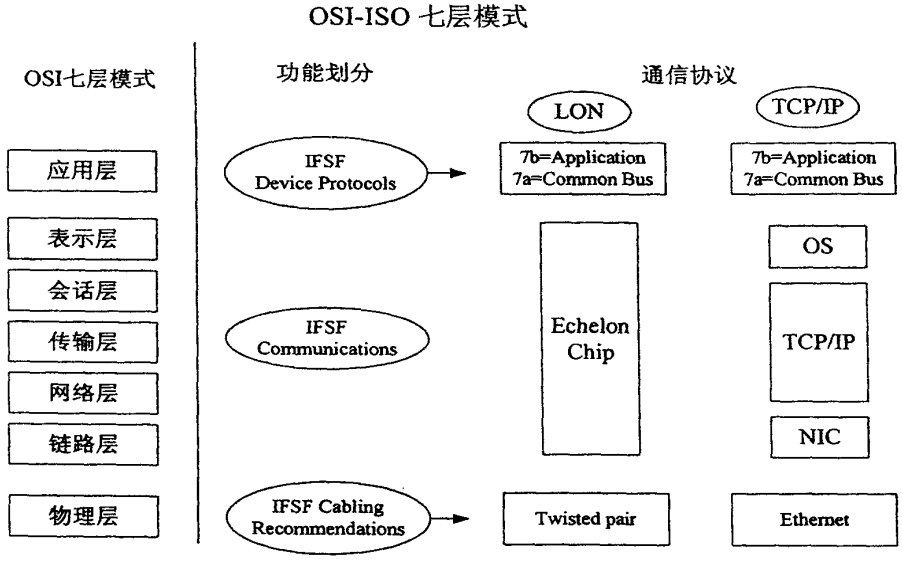



图 2-1 IFSF ISO-OSI 七层协议实现图

Fig.2-1 IFSF implementation of the ISO-OSI seven-layer model

局部操作网络[12]（Local Operating Network，LonWorks），即图 2-1 中的 LON 网络，是一种工业现场总线[13]，由美国 Echelon 公司于 1990 年推出，用于开发监控网络系统。LonWorks 网络基于 LonTalk 通信协议，遵守 ISO 的七层模型，由支持多种 IO 功能的智能节点（Node）组成，并通过神经元（Neuron）专用芯片上的硬件和固件来实现。LonWorks 网络在安全方面有着无与伦比的优势，而且它采用面向对象的设计方法，应用网络变量来提高数据传输速率和可靠性；它创立的 LonTalk 七层协议支持多种传输介质，能够提供 OSI 模型定义的全部七层服务，已占据控制领域开放标准的主流地位；LonWorks 是新一代控制网技术，有利于实现系统的统一[14-15]。

TCP/IP 是当今流行最为广泛的一种网络通信协议，与 OSI 体系结构相同，也采用分层结构，分四层：网络接口层、Internet 层、传输层和应用层。TCP/IP 可以在诸如 PC、嵌入式器件等设备上实现；许多操作系统，如 Windows、UNIX/Linux、

μC/OS-Ⅱ等，都支持 TCP/IP 应用程序编写接口，也就是“Socket API”，API 提供了可以与任何其他使用 TCP/IP 协议族的 Internet 上的设备进行通信的工具[16]，使得在 TCP/IP 之上实现 IFSF 协议更加简便。

由于 TCP/IP 是开放的系统，可以不花钱或花很少的钱就可以实现，本装置中基于 IFSF 协议设计的软件，选择了 TCP/IP 协议通信方式，可以更广泛的使用，降低成本。

### 2.1.2 IFSF 实现原理

IFSF 协议的实现主要是建立在数据库基础上的：假定每台设备上都有一个用来存取该设备相关控制数据的数据库（可以是完整的数据库系统，或是一般的表等数据结构），其它设备只需读写该数据库便能控制该设备；根据 IFSF 协议规定，先将数据库每个存取位置代表的意义规定好，以便控制方了解该位置存储信息的意义，通过读写实现对数据库的访问，而被控方对该请求做出响应，并加以处理[17]。

## 2.2 IFSF 协议的数据库格式定义

IFSF 信息分为心跳（HeartBeat）和数据消息（Data Message）两种。

HeartBeat 是每个基于 IFSF 协议的前庭设备为了证明自己在线，而每隔一段时间就在 IFSF 网络上广播的一个消息，若是其它设备在一定时间内没有收到该消息，将视该设备离线，不可通信。在 TCP/IP 上是用 UDP 广播包来实现 HeartBeat，一个 HeartBeat 即为一个 UDP 广播包。

在 IFSF 网络中，Data Message 是以数据库读写形式来访问的，有六种基本消息：读消息（Read Messages）、写消息（Write Messages）、有确认主动数据消息（Unsolicited Data Message with Acknowledge）、无确认主动数据消息（Unsolicited Data Message without Acknowledge）、回答消息（Answer Messages）、确认消息（Acknowledge Messages）[18]。六种消息的发送序列如表 2-1 所示。

“读消息”允许发送设备读取接收设备指定数据库中可访问的元素值，可以读取一个或多个数据元素。发送端将要读取的数据列表（包括数据标识和接收端标识）放入“读消息”中发送，然后接收端须在超时时间段内以“回答消息（包

含被请求数据标识和数据内容)”来响应“读消息”。如果接收方不能回应，也必须在超时时间段内发送“确认消息”来告知发送端不能进行回应的原因：接收端繁忙或请求数据错误等，否则认为发送失败。

表2-1 消息发送序列表

Table2-1 Message-delivering sequence

| 描述 | 发送方 | 发送方向 | 接收方 |
|---|---|---|---|
| 发送读消息请求数据 | 读消息 | → | |
| | | ← | 回答消息 |
| 发送读消息请求多个数据 | 读消息 | → | |
| | | ← | 回答消息 |
| | | ← | 回答消息 |
| | | | ... |
| | | ← | 回答消息 |
| 发送写消息写入数据 | 写消息 | → | |
| | | ← | 确认消息 |
| 发送有确认主动消息 | 主动数据消息 | → | |
| | | ← | 第一个接收方确认消息 |
| | | ← | 第二个接收方确认消息 |
| | | | ... |
| | | ← | 第N个接收方确认消息 |
| 发送无确认主动消息 | 主动数据消息 | → | |

“写消息”用来实现数据从发送设备到接收设备的传送。接收端定义了数据标识，发送端将数据标识和内容放入“写消息”中发送，接收端接收后识别数据标识并保存数据内容，并且在超时时间段内发送“确认消息”，否则认为发送失败。

“有确认主动数据消息”发送的是特定数据元素值。接收端须在超时时间段内回以“确认消息”，与“写消息”的不同在于数据元素的定义是在发送端实现的。

“无确认主动数据消息”也是用于发送特定数据元素值，与“有确认主动数据消息”区别是接收端不需要回应该消息。

“回答消息”发送的是“读消息”请求的数据元素值。

“确认消息”是由接收端发送的对于“读消息”、“写消息”以及“有确认主动消息”做出的回应。

为了完成不同设备间的通信，IFSF 定义了设备间发送消息的通用格式，即 Data Message 的通用帧格式，如表 2-2 所示。其中，M_St 定义了发送的消息类型，具体定义如表 2-3；BL 分块号是用于 LonWorks 网络的，在 TCP/IP 中，数据包长度

满足 IFSF 应用协议要求，故而不需要 BL 数据元素；IFSF_MC 用于通信层过滤接收到的消息，0 代表该消息由应用层而不是神经元专用芯片处理，1 代表心跳消息，2 代表的是通信数据库的消息，由通信服务层处理。

表2-2 IFSF中Data Message 的通用帧格式
Table2-2 General format of IFSF message

| 英文缩写 | 字节数 | 描述 |
|---|---|---|
| LNAR | 2 | 接收端逻辑节点地址 |
| LNAO | 2 | 发送端逻辑节点地址 |
| IFSF_MC | 1 | 消息类型代码 |
| BL | 1 | 分块号 |
| M_St | 1 | 消息的状态 |
| M_Lg | 1 | 消息的长度 |
| DB_Ad_Lg | 1 | 数据库地址的长度 |
| DB_Ad | 1-8 | 数据库地址 |
| Data_Id | 1 | 应用数据的标识符 |
| Data_Lg | 1 或 3 | 应用数据的长度 |
| Data_El | n | 应用数据元素内容 |

表2-3 M_St定义的消息类型
Table2-3 Message type defined by M_St

| Bit8 | Bit7 | Bit6 | Bit5-1 | 描述 |
|---|---|---|---|---|
| 0 | 0 | 0 | 序号0-31 | 读消息 |
| 0 | 0 | 1 | 序号0-31 | 回答消息 |
| 0 | 1 | 0 | 序号0-31 | 写消息 |
| 0 | 1 | 1 | 序号0-31 | 有确认主动数据消息 |
| 1 | 0 | 0 | 序号0-31 | 无确认主动数据消息 |
| 1 | 1 | 1 | 序号0-31 | 确认消息 |

表2-4 HeartBeat帧格式
Table2-4 Format of HeartBeat

| 消息字段 | 说明 | | |
|---|---|---|---|
| LANO | 发送端的逻辑节点地址，由子网号和节点号组成 | | |
| IFSF_MC | 为1，代表发送的消息是心跳 | | |
| Device_Status | 设备状态，1个字节，0-255 | Bit1 | 为0时代表设备还需要配置 |
| | | Bit2-7 | 保留 |
| | | Bit8 | 为1时代表需要软件刷新 |

HeartBeat 用于设备间互相告知是否在线，是否可以通信，在 IFSF 网络中占有很重要的地位。它的帧格式如表 2-4 所示。每隔一个心跳间隔，节点设备需要发送一个心跳消息，如果在 3 个心跳间隔内其它节点设备没有收到该设备的心跳消息就会认为该设备离线。

## 2.3 IFSF 加油机应用协议

IFSF 几乎规定了所有前庭设备的应用，如液位仪应用[19]、支付柱应用[20]、洗车[21]和加油机应用[22]等，IFSF 加油机应用协议主要给出了加油点（Fuelling Point，FP）的行为模式和加油机数据库的定义。

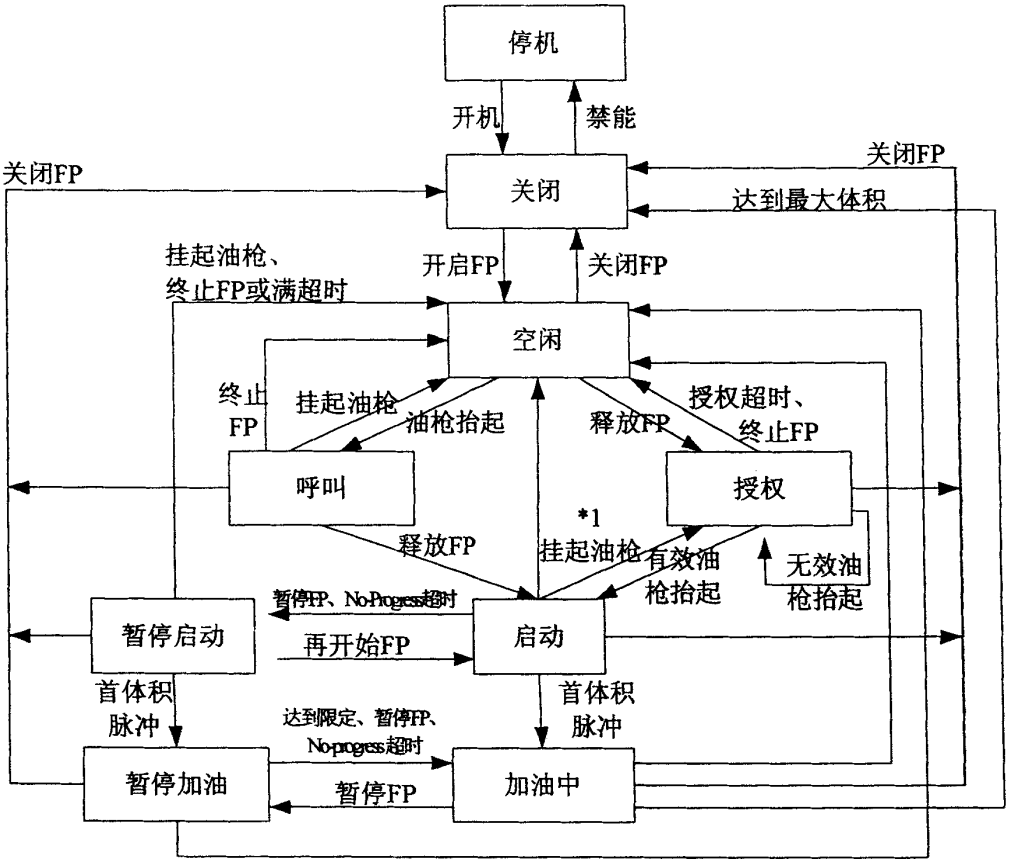



图 2-2 加油点状态图

Fig.2-2 FUELLING POINT STATE DIAGRAM

加油点状态图表如图 2-2 所示，其中包括加油点的所有状态和触发状态转换的事件。状态有 9 种，分别是：停机（INOPERATIVE）、关闭（CLOSED）、空闲（IDLE）、呼叫（CALLING）、授权（AUTHORISED）、启动（STARTED）、暂停启动（SUSPENDED STARTED）、加油中（FUELLING）和暂停加油（FUELLING SUSPENDED）。其中在启动后挂起油枪，如果状态“授权”是被允许的，就会由“启动”状态变为“授权”状态，否则就会变为“空闲”状态。

加油点未配置或是检测到重要错误时进入“停机”状态，此时是不可操作的；当基本数据配置完毕且没有检测到重要错误时，加油点进入“关闭”状态；开启 FP 进入“空闲”状态。加油有两种方式：一种是非定量加油，用户抬起油枪（不管是有效抬起还是无效抬起）后等待油泵启动，如果至少有一个交易缓冲区可用，将启动油泵加油；另一种是定量加油，被预授权先得到油泵，然后等待有效地油枪抬起，加油到预定油量后直接停止。

IFSF 协议还定义了不同状态下可能遇到的一些错误，并对交易缓冲区各个状态及其触发事件做了说明。每个加油点都会定义交易缓冲区用于未付款的加油交易，只要控制设备没有清空交易，加油点就会负责完成交易并将交易数据保存。交易缓冲区共有三种状态：清空交易（CLEARED TRANSACTION）、可付交易（PAYABLE TRANSACTION）和被锁交易（LOCKED TRANSACTION）。

## 2.4 基于 TCP/IP 的 IFSF 协议

TCP/IP 协议允许不同设备不同操作系统之间互相通信，被称作“全球互联网”或“因特网（Internet）”的基础[23]，它的分层结构中每层的协议如图 2-3 所示，应用层是用户程序，传输层最常用的协议就是 TCP 和 UDP（TCP 提供面向连接的可靠传送服务， UDP 与 IP 一样提供不可靠的传递服务），网络层包含 IP 协议，最后通过数据链路层接入以太网。

TCP/IP 协议将 IFSF 信息包在每层上封装，如图 2-4 所示，使用 UDP 协议只是将其 TCP 头部换成 UDP 头部即可。TCP/IP 提供了很多服务，而其中只有很少一部分是基于 TCP/IP 的 IFSF 所必需的。在 TCP/IP 上实现 IFSF 的最小的 IP 栈包括：IP（网际协议）、ARP（地址解析协议）、ICMP（互联网控制报文协议）、TCP（传输控制协议）、UDP（用户数据报协议）和 DHCP（动态主机配置协议，是客户端还是服务器取决于设备）。

在 TCP/IP 上实现 IFSF 的架构如图 2-5 所示，有四部分： IFSF 应用程序、IP 栈、DHCP 服务器和 IFSF 与 IP 转换器模块[24]。IFSF 应用程序部分在各个 IFSF 标准规范中定义，用于实现前庭设备各功能；IP 栈部分是网络的接口，实现各种 IP 协议，提供网络连接管理服务；DHCP 服务器用来为同一网络中的 IP 设备分配 IP

地址，可以是 IFSF 设备的一部分，也可以是一个单独的设备，但是一个网络中只能有一个 DHCP 服务器。

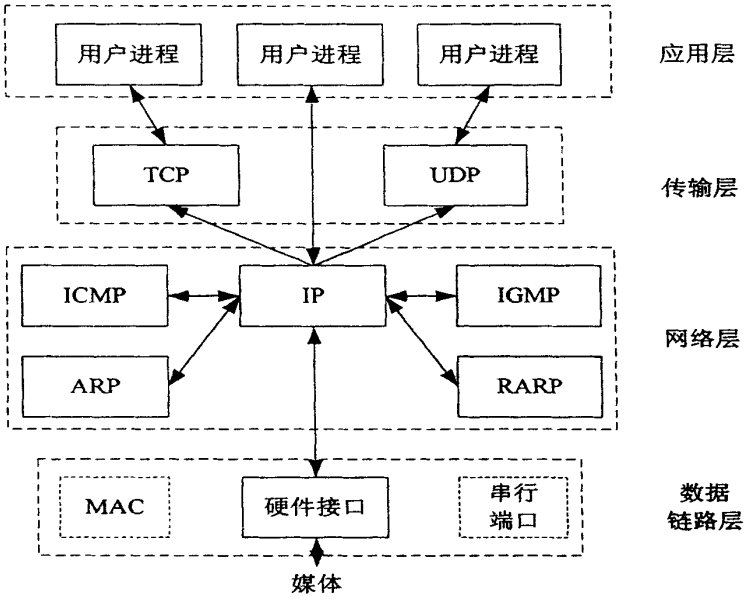



图 2-3 TCP/IP 协议栈的各层协议

Fig.2-3 Protocols of TCP/IP Protocol Stack layers

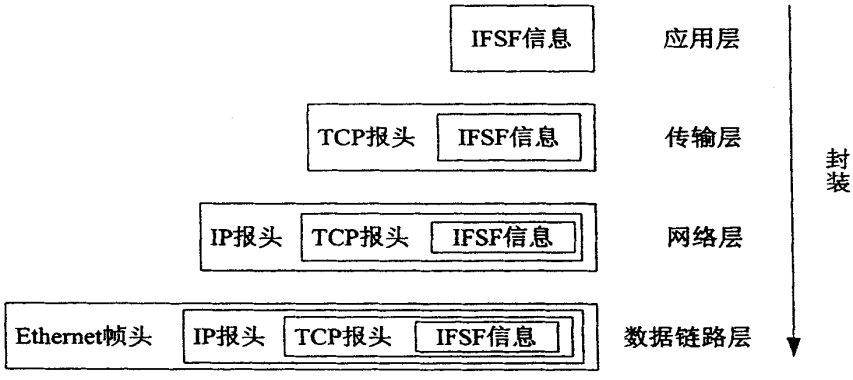



图 2-4 IFSF 信息的 TCP/IP 封装

Fig.2-4 TCP/IP encapsulation of IFSF message

IFSF 与 IP 转换器模块（IFSF to IP Converter，IIPC）就像是到本地 IFSF 应用程序的 IFSF 接口，它的作用是接收所有的 IFSF 消息，将它们封装到 IP 数据报里，通过本地局域网发送到远程设备。该模块有三个主要目标：通过心跳代理发送和

接收心跳；保持一个局域网上的所有活动连接列表；将局域网上所有的数据和控制信息封装到 TCP 数据流中。模块内部各部分说明如下：

IFSF 接口负责的是 IFSF 应用程序和其它 IP 通信服务之间的接口。它将维持所有 LNA（逻辑节点地址）及其对应的 IP/端口地址（IP 地址、协议和端口号组合成的套接字地址）的一个表格。IFSF 接口将所有的心跳发送给心跳代理，心跳外所有其他消息发送给连接控制器；IFSF 接口接收来自其它设备的心跳，若该设备不在列表中，就将该设备的 LNA 和套接字地址添加到表中，然后发送 IFSF 心跳到该接口正在托管的所有应用程序。

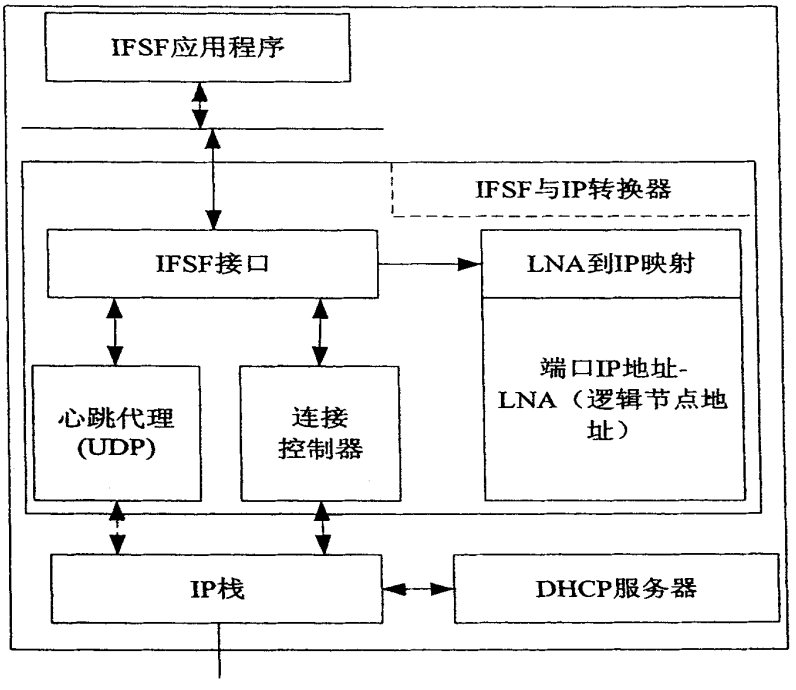



图 2-5 IFSF 的 TCP/IP 实现架构

Fig.2-5 Architecture of IFSF over TCP/IP

心跳代理是负责封包本地应用程序的心跳并使用 UDP 数据报将它们广播。传入的心跳消息经过心跳代理，然后通过 IFSF 接口发送；同时，心跳代理也将它从本地设备接收到的心跳发送到其它本地托管的设备。一个 IFSF 心跳包含 LNA 和设备状态位。为了使心跳在 IP 网络上有效，该消息需要将远程设备用来与本地 IFSF 应用程序连接的本地 IFSF 应用程序的主机 IP 地址和本地主机的端口号添加到它本

身。远程设备接收到这个消息时，它会剥掉 IP 地址和端口号信息，记录发送设备的 LNA，并将心跳消息以 IFSF 协议标准格式传给 IFSF 应用程序。远程设备从接收到的消息中提取数据，并在 LNA 到 IP 的映射表（该表作用是将 IP 地址和端口地址映射到已在网络中声明过自己的设备的 LNA）上登记。每当 IIPC 收到一个心跳，这个心跳会为该远程设备重置定时器。设置定时器的目的是如果在一段时间内未收到心跳就通知 IIPC。当 IIPC 接到该通知时会假定该远程设备已经离线并从 IP 到 LNA 映射表中删除该设备，发送一个连接断开消息到该设备，并关闭任何除了主服务连接以外的所有与该设备相连的本地连接。下一次收到从远程设备来的心跳或 TCP 连接请求后，重新登记到 IP 到 LNA 的映射表中，并且再次启动定时器。

连接控制器是负责管理 TCP 连接的，心跳以外的任何 IFSF 消息都通过这个接口处理。

所有的数据和控制信息，将封装在 TCP 中发送到对应的地址。该地址是根据从 IFSF 消息中提取的 LNA 信息在 LNA 到 IP 的映射表中找到对应的套接字地址来确定的。接收端将接收该消息，去掉 TCP 包装，将该 IFSF 信息传递到设备应用程序。

## 2.5 嵌入式操作系统 μC/OS-Ⅱ的移植

嵌入式系统是“软硬件可裁剪”的，满足应用系统对功能、成本、可靠性、体积和功耗严格要求的计算机专用系统[25]。它在发展之初是基于单片机的，随着嵌入式操作系统的出现与完善、处理器性能的提高，尤其是 ARM 技术的出现，嵌入式系统深入到了工业自动化、航空航天、环境工程与自然和消费类电子等领域，提供一定的显示、检测和控制等功能[26-27]。

嵌入式系统可以分三部分：嵌入式硬件平台、嵌入式操作系统和嵌入式系统应用。嵌入式硬件平台提供给嵌入式操作系统和应用程序运行，是各种嵌入式设备（如 ARM、单片机）；操作系统实现系统初始化和应用的任务调度及控制等功能，现在的嵌入式操作系统有商用型和免费型两种，免费的操作系统主要有 Linux 和 μC/OS-Ⅱ；在硬件平台搭建好，操作系统移植成功后，嵌入式系统应用会运行

于该嵌入式操作系统，完成用户所需的特定功能[28]。

对于嵌入式操作系统，本装置选用的是实时操作系统μC/OS-II。

### 2.5.1 μC/OS-II 的简介

μC/OS-II 是源代码公开的实时操作系统，是针对嵌入式应用设计的，使用 ANSI C 编写，可移植性好，而且它的内核小，易于剪裁，可以运行于 8 位、16 位、32 位甚至 64 位处理器上。μC/OS-II 是抢占式实时多任务的内核，它提供的基本服务是任务切换，总是运行就绪任务中优先级最高的；每个任务都有唯一的优先级和自己独立的堆栈。

移植，是指使一个操作系统实时内核可以在某个微处理器或者微控制器上运行。大部分μC/OS-II 源代码是用 C 语言编成的，但某些与处理器相关的代码仍需要用 C 语言和汇编语言来完成。例如，只有通过汇编语言才能实现对处理器寄存器的读/写操作。μC/OS-II 的移植相对来说还是比较容易的，因为在设计μC/OS-II 的时候就充分考虑了可移植性[29]。

要使μC/OS-II 系统正常运行，处理器必须满足以下要求：

1) 处理器的 C 编译器能产生可被多个任务同时调用的可重入代码。

2) 处理器支持并能产生定时中断 （通常在 10 到 100HZ），以实现多任务之间的调度。

3) 用 C 语言就可以打开和关闭中断。

4) 处理器支持能够容纳一定量数据（可能是几千字节）的硬件堆栈。

5) 处理器有将堆栈指针和其他 CPU 寄存器读出和存储到堆栈或内存的指令，因为寄存器的入栈和出栈是μC/OS-II 多任务调度的基础[30-31]。

ARM7TDMI-S 处理器完全满足上述要求。LPC2378 基于 ARM7TDMI-S 内核，符合μC/OS-II 移植条件。

如图 2-6，它示意了μC/OS-II 的软件体系结构及其与硬件的关系。由图可知，μC/OS-II 移植的与处理器相关的代码就在 OS_CPU.H、OS_CPU_A.ASM 和 OS_CPU_C.C 三个文件中。移植所要做的工作是在不同的处理器上用不同的汇编语言来改写与处理器有关的代码和其他与处理器特性相关的部分[32]。

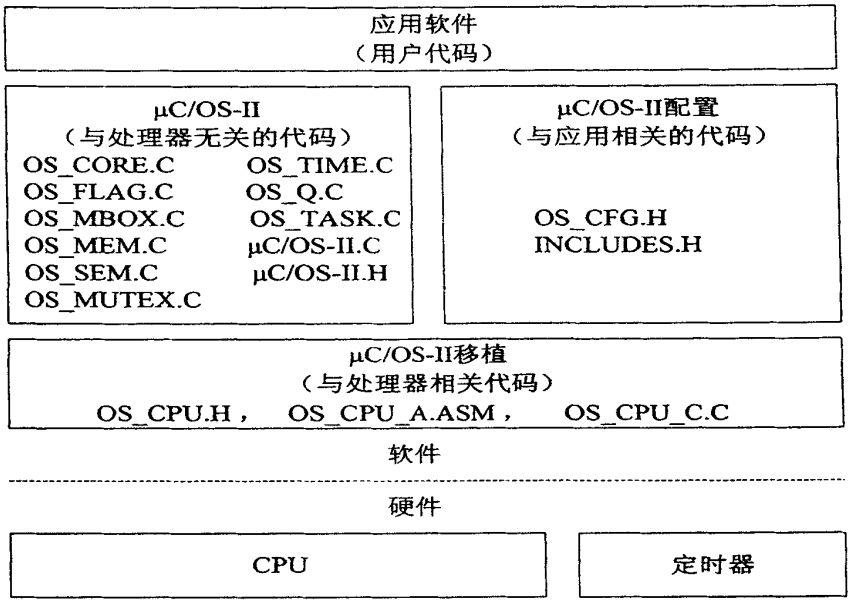



图 2-6 μC/OS-II 硬、软件体系结构

Fig.2-6 The hardware and software structure of μC/OS-II

## 2.5.2 μC/OS-II 在 LPC2378 上的移植实现

1) OS_CPU.H 包含了与处理器相关的常数、宏和类型，用#define 来定义。由于不同处理器数据字长不同，为增加可移植性与直观性而重新定义数据类型。本课题中移植过程使用的是 RealView MDK 开发套件，数据类型定义如下：

```
typedef   unsigned   char     BOOLEAN;
typedef   unsigned   char     INT8U;
typedef   signed     char     INT8S;
typedef   unsigned   short    INT16U;
typedef   signed     short    INT16S;
typedef   unsigned   int      INT32U;
typedef   signed     int      INT32S;
typedef   float      FP32;
typedef   double     FP64;
typedef   unsigned   int      OS_STK;    /*堆栈入口宽度为 32 位*/
```

其中，建立任务声明堆栈时必须只能使用 OS_STK 作为数据类型。除此之外，

还需要用#define 定义常量 OS_STK_GROWTH 的值，表明堆栈的方向。本文堆栈选择从上往下方向，将 OS_STK_GROWTH 置 1：

```
#define   OS_STK_GROWTH          1 ;
```

μC/OS-Ⅱ中，为保护临界段代码不被多任务或中断服务子程序破坏，处理临界段代码之前必须要先关中断，访问结束之后再开中断。所以，在 OS_CPU.H 中还声明了 OS_ENTER_CRITICAI( )和 OS_EXIT_CRITICAL( )两个宏来开中断和关中断。

2) 与处理器有关的用汇编语言编写的代码在 OS_CPU_A.ASM 中，共四个函数：

①OSStartHighRdy();该函数被 OSStart()调用，使就绪任务中具有最高优先级的任务开始运行，代码如下：

```
LDR       R0, OSTaskSwHook   ;调用函数 OSTaskSwHook();
MOV       LR, PC
BX        R0
LDR       R0, OSRunning      ;将 OSRunning 置 1
MOV       R1, #1
STRB      R1, [R0]
LDR       R0, OSTCBHighRdy   ;得到最高优先级任务控制块 TCB 的地址
LDR       R0, [R0]           ;得到堆栈指针
LDR       SP, [R0]           ;切换到新的堆栈
LDR       R0, [SP], #4       ;从栈中得到任务新的状态
MSR       SPSR_cxsf, R0
LDMFD     SP!, {R0-R12, LR, PC}^   ;切换到新任务开始运行
```

OSStartHighRdy ()必须调用函数 OSTaskSwHook()，因为 OSStartHighRdy ()不保存当前任务的寄存器，只恢复高优先级任务的寄存器。

②OSCtxSw();该函数为任务级的切换函数，实现的功能是将当前任务的内容入栈；保存当前堆栈指针 SP，得到最高优先级任务的 SP；从新任务堆栈中恢复该

最高优先级任务的内容，完成上下文切换。其代码如下：

```
;保存当前任务内容：
STMFD    SP!, {LR}   ; 保存返回地址
STMFD    SP!, {LR}
STMFD    SP!, {R0-R12}   ;保存寄存器内容
MRS      R0, CPSR    ;保存当前 CPSR
STMFD    SP!, {R0}
;保存当前任务的堆栈指针 SP 到当前任务控制块
LDR      R0,OSTCBCur
LDR      R1, [R0]
STR      SP, [R1]
;调用 OSTaskSwHook();
LDR      R0, OSTaskSwHook
MOV      LR, PC
BX       R0
;将就绪任务中优先级最高的任务的优先级赋给 OSPrioCur
LDR      R0, OSPrioCur
LDR      R1, OSPrioHighRdy
LDRB     R2, [R1]
STRB     R2, [R0]
;将该任务的控制块赋给 OSTCBCur
LDR      R0, OSTCBCur
LDR      R1, OSTCBHighRdy
LDR      R2, [R1]
STR      R2, [R0]
;得到该将要运行任务的堆栈指针
LDR      SP, [R2]
;恢复新的任务的内容
LDMFD    SP!, {R0}      ; 从新的任务堆栈中恢复 CPSR 值
```

```
MSR       SPSR_cxsf, R0
LDMFD     SP!, {R0-R12, LR, PC}^   ;将新的任务的内容出栈
```

③OSIntCtxSw();该函数在中断服务子程序 ISR 中被调用，完成中断级的任务切换。它与 OSCtxSw()代码大致相同，只是 ISR 中 CPU 现场已保存，不需要在 OSIntCtxSw( )中进行 CPU 寄存器的保存，具体代码如下：

```
LDR       R0, OSTaskSwHook
MOV       LR, PC
BX        R0
;将就绪任务中优先级最高的任务的优先级赋给 OSPrioCur
LDR       R0,OSPrioCur
LDR       R1, OSPrioHighRdy
LDRB      R2, [R1]
STRB      R2, [R0]
;将该任务的控制块赋给 OSTCBCur
LDR       R0, OSTCBCur
LDR       R1, OSTCBHighRdy
LDR       R2, [R1]
STR       R2, [R0]
;得到该将要运行任务的堆栈指针
LDR       SP, [R2]
;恢复新的任务的内容
LDMFD     SP!, {R0}      ; 从新的任务堆栈中恢复 CPSR 值
MSR       SPSR_cxsf, R0
LDMFD     SP!, {R0-R12, LR, PC}^   ;将新的任务的内容出栈
```

④OSTickISR();该函数是系统的时钟节拍中断服务函数。μC/OS-Ⅱ中要有一个完成时间延迟和超时功能的周期性的时钟源，即时钟节拍，频率为每秒 10-100 次。在调用 OSStart()之后，必须启动时钟节拍中断，但是 OSStart()函数不返回，所以时钟节拍的启动要在多任务启动后运行的第一个任务，即优先级最高的任务

当中。该函数完成功能为：保存 CPU 寄存器内容，调用 OSIntEnter()、OSTimeTick()和 OSIntExit()，从栈中恢复寄存器内容，最后中断返回。

3) μC/OS-Ⅱ移植要求用户编写与操作系统相关的 10 个函数，OSTaskstkInit()、OSTaskCreateHook()、OSTaskDelHook()、OSTaskstatHooK()、OSTaskSwHook()、OSTaskIdleHook()、OSInitHookBegin()、OSInitHookEnd()、OSTCBInitHook()以及 OSTimeTickHook()，在 OS_CPU_C.C 中，用 C 语言编写。其中，OSTaskstkInit()是惟一必要的函数，其它的可以不包含任何代码，但必须进行声明[33]。本文中编写的函数有：

①OSTaskstkInit();该函数作用是完成栈结构的初始化，由 OSTaskCreate()和 OSTaskCreate Ext()调用。如图 2-7 所示，OSTaskstkInit()在建立任务时，将任务堆栈初始化并返回栈顶指针指向的地址。假设堆栈从上向下递减。

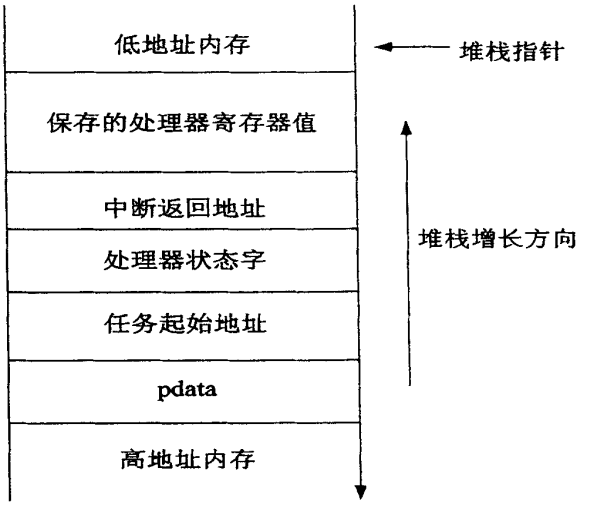



图 2-7 堆栈初始化结构

Fig.2-7 Storehouse initializing structure

堆栈初始化之后，看起来好像刚发生中断一样：pdata 会被保存到堆栈；因为任务不需要返回，所以只需要知道起始地址即可，所以任务起始地址也会被保存到堆栈中；处理器寄存器的值会按照一定的顺序保存，最后返回堆栈指针所指向的地址。

②OSTaskCreateHook();该函数在任务创建时被调用，调用期间中断是开着的，

参数是指向将要建立任务的 TCB 的指针，它可以访问 TCB 结构所有的成员。

③OSTaskDelHook();该函数在删除一个任务时被调用，参数是指向将要被删除任务的 TCB 的指针，该函数被调用时要禁止中断。

④OSTaskSwHook();该函数在任务切换时被调用，能直接访问指向将要被切换出去的当前任务的 TCB 的指针 OSTCBCur 和指向将要运行的任务的 TCB 的指针 OSTCBHighRdy。在调用此函数期间，中断是被禁止的。

## 2.6 μC/TCP-IP 协议栈的移植

嵌入式微控制技术越来越成熟，嵌入式系统的开发也越来越热门。由于嵌入式系统与网络的结合，在嵌入式实时操作系统中引入协议栈，支持嵌入式设备接入网络，已成为嵌入式领域重要的研究方向[34]，也就是要在嵌入式系统上移植 TCP/IP，这对于容量小、速度慢的低端 MCU 来说比较困难。在此基础上开发了一些适合嵌入式系统的嵌入式 TCP/IP 协议栈[35-37]，比较主流并开源的如：代码相对较多不太容易裁剪的 BSD TCP/IP 协议栈、占用 RAM 较少适合用于低端嵌入式系统应用的 LwIP[38-40]、带最小化用户接口并能应用串行链路的 μC/IP[41]、专为 8 或 16 位 MCU 设计的小代码容量的 uip[42]等。本设计中采用的是 Micrium 公司的 μC/TCP-IP[43]。

μC/OS-II 是实现任务调度与通信的内核，μC/TCP-IP 协议栈基于 μC/OS-II 开发，使用 C 语言编写，可以为其补充网络通信功能。μC/TCP-IP 可以运行于不同的 CPU 和操作系统，能够实现 TCP/IP 协议集内最核心、最重要的协议：IP、ARP、ICMP、UDP、TCP，除本身 μC/TCP-IP Sockets 外，还提供了 BSD Sockets 接口[44-45]，能够完成很多应用。

μC/TCP-IP 协议栈层次结构及其与应用程序、硬件的关系如图 2-8 所示，其中中间 4 层是 μC/TCP-IP 协议栈内容。由图可知，μC/TCP-IP 的移植，主要是要实现与硬件和操作系统有关的函数。其中与 CPU 有关的函数在 cpu_def.h、cpu. h 和 cpu_asm. s 三个文件中，cpu_def.h 中主要定义了数据对齐方式和 CPU 大小端方式；cpu. h 中包含了 CPU 中断开关函数的宏定义并定义了数据类型；而 cpu_asm. s 是对中断开关函数的定义。与 NIC 有关的函数有 NetNIC_Init()、NetNIC_IntEn()、

NetNIC_ConnStatusGet()、NetNIC_RxPktGetSize()、NetNIC_ RxPkt()、NetNIC _RxPktDiscard()、NetNIC_TxPktPrepare()和 NetNIC_TxPkt()，主要是实现网卡驱动和消息帧的收发。与操作系统有关的函数在 net_os. c 内，主要负责接收处理从网卡来的消息帧的任务 NetOS_IF_RxTask 和时间管理任务 NetOS_Tmr_Task，完善这两个函数即可完成移植。

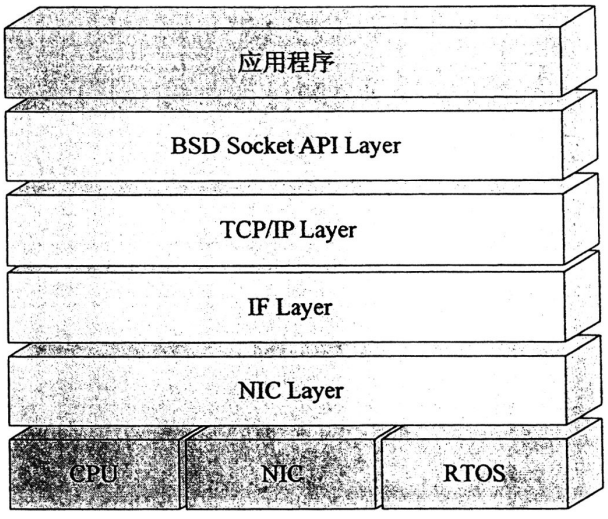



图 2-8 μC/TCP-IP 协议栈应用层次图

Fig.2-8 μC/TCP-IP protocol stack layers

## 2.7 本章小结

本章详细介绍了 IFSF 协议的定义、实现原理和数据库格式，并简介了 IFSF 加油机协议，以及 IFSF 的 TCP/IP 实现方式；说明了选择μC/OS-Ⅱ的原因，阐述了μC/OS-Ⅱ在 ARM 上的移植过程，最后简要介绍了系统中用到的 μC/TCP-IP 协议栈及其移植。

# 第 3 章 硬件电路结构设计

## 3.1 加油机控制装置总体架构

加油机控制装置的功能是要实现加油操作和加油数据的实时采集，将加油信息及时的反馈到后台管理中心，并迅速把后台及其他前庭设备发出的控制指令传达给加油机各部分。本课题所研究的加油机控制装置基于 IFSF 协议，不需要进行协议转换，采用网络通信，可以直接连接到后台管理中心和总部管理中心，使得管理人员可以及时掌握成品油的销售情况，有效地防止空罐、油品滞销等问题，以便合理经营，提高效率；除此之外，经营商还可以方便的更换加油机，不用受制于设备厂商，只要通信速率稳定，在条件允许的情况下，可随意增删加油机，加油站规模不受限制。

加油机控制装置的总体结构框图如图 3-1 所示，该装置包含了六部分：中央控制模块、执行部件、网络通信模块、显示模块、数据采集模块和供电模块。嵌入式处理器有 3 种：MPU（微处理器，如 ARM[46]）、MCU（微控制器，如 51 单片机）和 DSP（数字信号处理器）。在设计硬件电路时，中央控制模块选择的处理器是目前非常流行的 ARM7 系列处理器，选择的芯片是 PHILIPS 公司的 LPC2378。

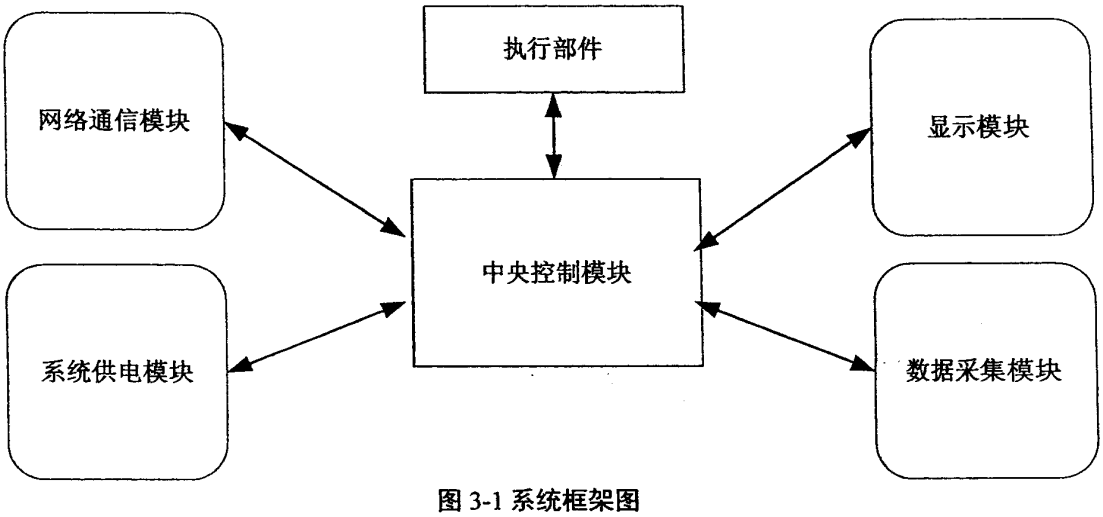



图 3-1 系统框架图

Fig.3-1 System diagram

系统的整体硬件结构示意如图 3-2 所示：数据采集信号包括键盘输入信号、流量计脉冲计数信号和油枪开关信号，由 ARM （LPC2378）控制，输出油泵与电磁阀开关信号，将油量显示到液晶显示器，并通过以太网接口与后台管理中心及其它 IFSF 设备通信，要求能够快速联网，并且要有很高的通信速率。

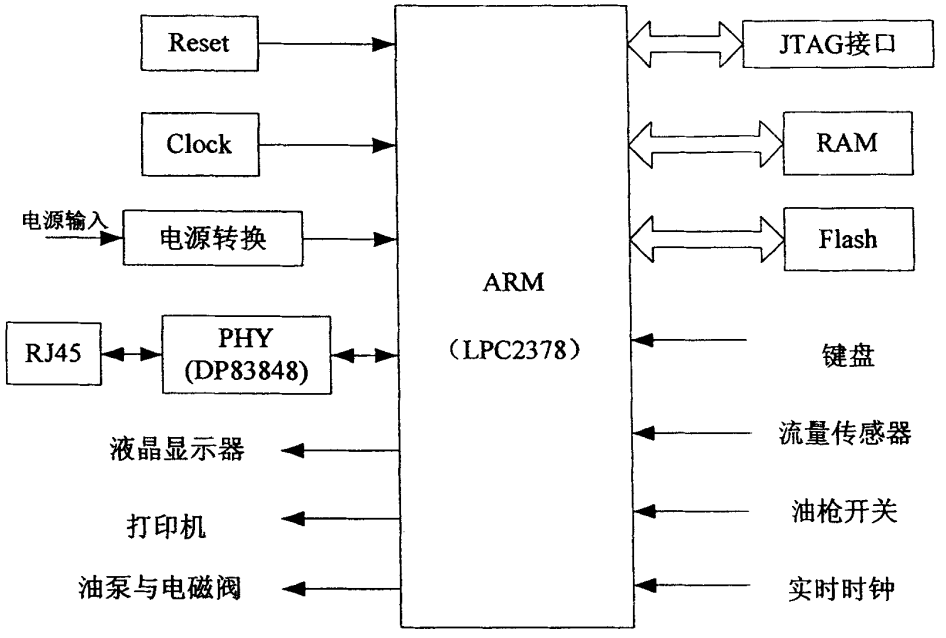



**图 3-2 硬件结构示意图**

Fig.3-2 Hardware structure diagram

加油机开机后初始化，有顾客加油时，选择定量加油或非定量加油，加油流程为：抬起油枪，油枪开关向控制模块发出请求加油信号，控制模块接收到该信号后作出指示，打开电子阀和油泵，使加油机处于加油状态；开启油枪，油液将通过油枪输出，当流经流量计时，流量计内部的活塞会在油液的推动下做往复运动，每次往复运动都会排出一定体积的油液，并带动流量计转动轴上的传感器分度盘，以此产生油液计数脉冲信号。控制模块接收该计数脉冲信号并进行处理，将加油数据通过显示模块显示出来；当油液体积达到预定数值或者接收到油枪挂起信号时，控制模块发送控制信号关闭电磁阀和油泵，加油结束[47]。在加油过程中，加油机产生的数据信息存储到 FeRAM（铁电存储器）中，并通过网络传送到后台管理中心，以便查询和监控。

以上是整个加油机控制装置的总体结构及加油流程，实物图见附录 A。

## 3.2 中央控制模块

中央控制模块主要包括 ARM 处理器、RAM、Flash、JTAG 和复位芯片。

1) ARM 处理器通过 IO 口对其它器件和模块进行控制。

ARM 公司成立于 1990 年 10 月，主要出售芯片设计技术给各大半导体公司。它专门设计开发基于 RISC（Reduced Instruction Set Computer，精简指令集计算机）技术的芯片，与传统的基于 CISC（Complex Instruction Set Computer，复杂指令集计算机）的架构相比有着许多的优点，如指令格式精简、程序执行速度提高等。各大半导体生产商购买 ARM 公司设计的 ARM 微处理器核，加上不同的外围电路，形成自己的 ARM 微处理器芯片，推向市场[48]。

ARM 处理器系列有通用处理器系列 ARM7-ARM11、高安全应用处理器 SecurCore 系列和 Intel 公司的 Xscale 系列等。ARM7 微处理器系列有 ARM7TDMI、ARM7TDMI-S、ARM720T 和 ARM7EJ 四种类型的内核。ARM7TDMI-S 是 32 位嵌入式 RISC 处理器，支持 16 位 Thumb 指令集，使用三级流水线来提高指令流的处理速度，应用非常广泛[49-51]。

PHILIPS 公司的 LPC2378 是基于 ARM7TDMI-S 内核的 32 位微控制器，可在高达 72MHz 的频率下工作，支持 10M/100M Ethernet，包含 USB 2.0 全速接口和两路 CAN 通道，功能强大成本低；具有高达 512KB 的片内 Flash，高达 58KB 的 SRAM，10 位 A/D 和 D/A 转换器带 8 个管脚输入复用，可作系统时钟的 4MHz 的内部 RC 振荡器，104 个通用的 I/O 管脚，满足对多个模块的控制需求；此外，它内嵌一个功能齐全的以太网 MAC（媒体访问控制器），支持精简的媒体独立接口 RMII（Reduced Media Independent Interface）和带缓冲的 DMA（直接存储器访问）接口，可在半双工或全双工模式下进行操作[52]。因其只包含了以太网 MAC 控制器，却并未提供物理层接口，所以以太网模块要使用支持 MII 或 RMII 协议的片外 PHY 来与片内的 MIIM（媒体独立接口管理）串行总线进行连接，以实现网络功能。

LPC2378 的以太网模块包含主机寄存器模块、连接到 AHB（高性能总线）的 DMA 接口、以太网 MAC 和附属的 RMII 适配器、发送通道和接收通道；通过与

物理层器件的通信来控制网络功能的实现。

2) 本装置中外扩了 RAM 和 Flash，为的是将没有及时发送到后台的加油机数据存储，以方便查看与监督。这里选择的 RAM 芯片是 RAMTRON 公司的铁电存储器 FM28V020，Flash 选择的是 ATMEL 公司的 AT45DB041D。

FM28V020[53]是一款 256K 的非易失性铁电存储器，在断电的情况下数据也不会丢失，能快速访问，无延时写入，读写次数几乎无限，非常适用于实时性要求比较高且需频繁写入的非易失性的嵌入式操作系统。对于加油机来说，数据读写比较频繁且数据存储要求非易失，所以选择 FM28V020，其引脚描述如表 3-1 所示。根据读写时序和引脚功能，设计其接口电路如图 3-3 所示。

表 3-1 FM28V020 引脚描述

Table3-1 FM28V020 Pin Descriptions

| 引脚名称 | 类型 | 引脚说明 |
|---|---|---|
| A(14-0) | 输入 | 地址输入 |
| /CE | 输入 | 芯片使能 |
| /WE | 输入 | 写使能 |
| /OE | 输入 | 输出使能 |
| D(7-0) | 输入/输出 | 数据输入/输出 |
| VDD | 电源 | 电源电压 |
| VSS | 电源 | 地 |

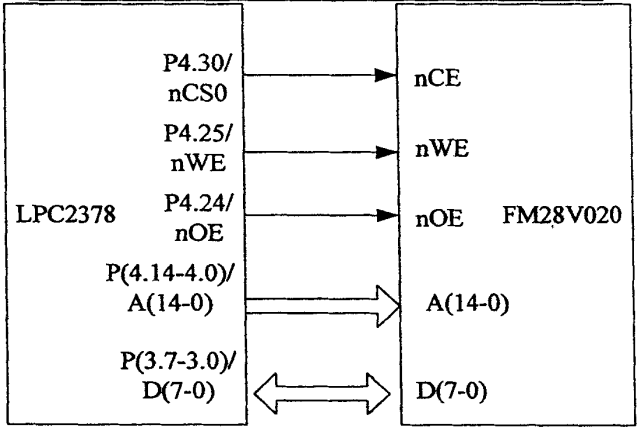



图 3-3 FM28V020 接口电路示意图

Fig.3-3 FM28V020 interface circuit diagram

AT45DB0410D[54]是 4M 的 Flash 芯片，支持 RapidS 串行接口，可实现高速应用，包含一个主存储器，分 2048 页，每页 256 或 264 字节；两个 256/264 字节的

SRAM 数据缓冲区；通过片选引脚 CS 使能，通过串行输入 SI、串行数据 SO 进行数据通信，串行时钟 SCK 对读写进行控制。AT45DB041D 主要是用来备份加油机数据。其接口电路原理图如图 3-4 所示。

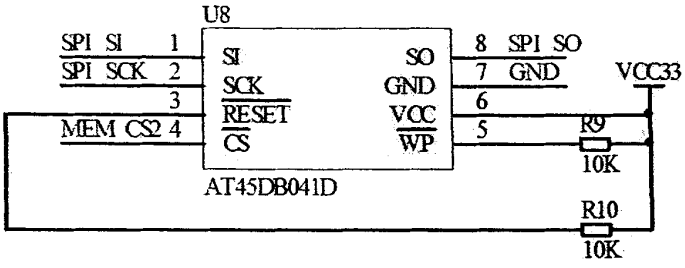



图 3-4 AT45DB0410D 接口电路示意图

Fig.3-4 AT45DB0410D interface circuit diagram

3) 复位芯片选择的是周立功公司专用的低功耗微处理器电源监视芯片 sipex708s[55]，该芯片包括看门狗电路、uP 复位模块和手动复位模块，其监视功能满足 ARM 对电源稳定性的要求。JTAG 接口电路用于系统测试和仿真，本系统采用的是标准的由 ARM 公司提供的 20 脚仿真接口。

## 3.3 网络通信模块

为了使加油站互联系统由传统站级结构向 IFSF 标准架构转变，加油机满足基于 TCP/IP 的 IFSF 要求、能够通过 Internet 通信就提供了非常大的便利。加油机直接与后台管理中心进行网络通信，不仅可以将加油机数据及时的传送到后台，更可以使设备连接不受限。所以，网络通信模块的研究在整个加油机控制装置研究中占有很重要的地位，以太网接口能够更好的体现加油机嵌入式系统的价值。网络通信部分主要由媒体访问控制器 MAC 和物理层接口 PHY（Physical Layer）两大部分构成。LPC2378 已经包含了 MAC 控制器，所以只需要在 LPC2378 外接一个物理层器件即可。这里选用的以太网物理层接口芯片是 National SemiConductor 公司的 DP83848C，它是一个 10/100Mb/s 单端口低功耗的物理层器件，提供了 MII 和 RMII 接口，可以很方便的与只支持 RMII 接口的 LPC2378 连接。

DP83848C 提供几种智能降功耗模式，可以提高产品的整体可靠性；它有 25MHz 时钟输出，很容易通过外接变压器和双绞线媒体相连；支持 MII 和 RMII，

确保了设计的易用性和灵活性；集成了子层来支持 10BASE-T 和 100BASE-TX 以太网协议；功耗小于 270mW、3.3V MAC 接口；48 引脚 LQFP（7mm×7mm）封装，只需要很小的空间。因此，DP83848C 广泛应用于高端外围设备、工业控制和工厂自动化操作以及通用的嵌入式应用领域[56]。

LPC2378 与 DP83848C 连接简便，直接通过 RMII 连接即可，然后再通过 RJ45 接入以太网。DP83848 支持 MII，RMII，以及 10M SNI（串行网络接口）的 MAC 接口方式，配合 LPC2378，本方案中选择 RMII 方式，通过 6 引脚和 39 引脚进行配置。LPC2378 与 DP83848C 的连接引脚数目比较少，数据传送速率每次是两位，在 DP83848C 的 34 引脚上需要加 50MHz 的振荡器。DP83848C 电路原理图如图 3-5 所示。

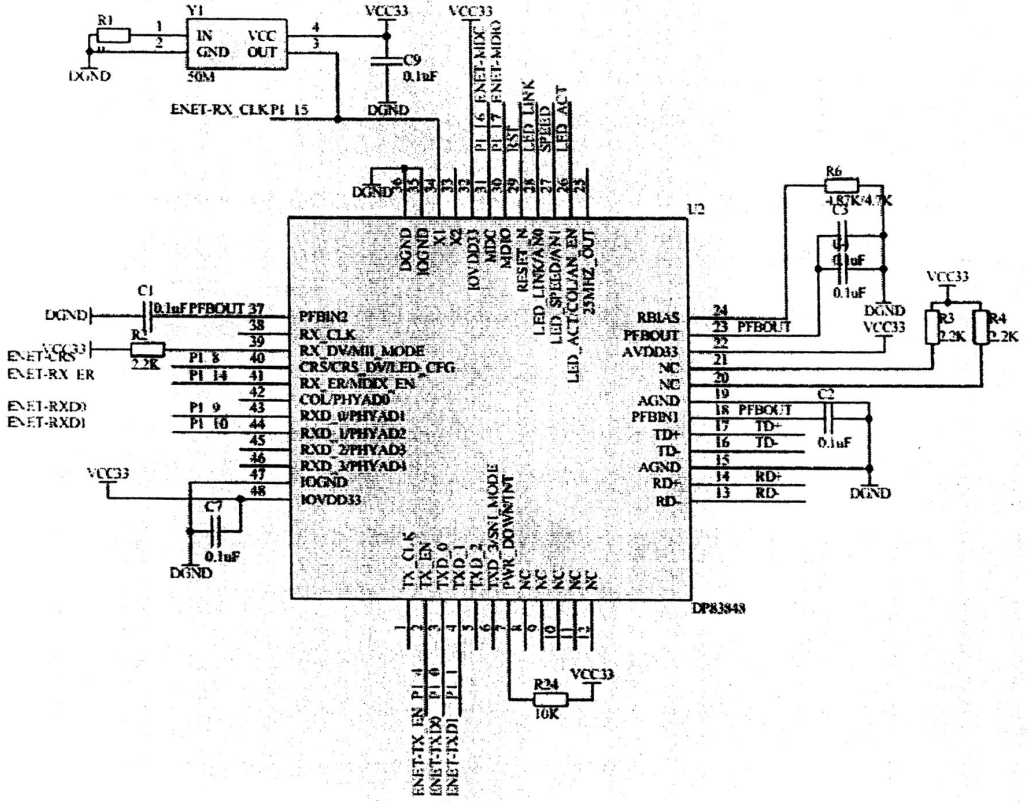



图 3-5 DP83848C 接口电路原理图

Fig.3-5 DP83848C interface circuit schematic

## 3.4 显示模块

加油机控制装置中显示模块包括两部分：采用字符液晶显示屏用于提示加油机状态信息；采用 8 段码液晶显示屏展示加油单价、加油升数和加油金额。加油机开启之后，会将加油机的操作状态显示到字符液晶显示屏上；而 8 段码液晶显示屏是由税控模块控制的，依照国家标准税控燃油加油机定型鉴定大纲 JJF 1060-1999，根据单价和采集到的实时加油升数计算出加油金额，将其保存并显示到 8 段码液晶显示屏上，以便顾客观看和国家检查。

字符液晶显示屏选用的是 LCD12232-1 点阵液晶显示模块，由 LPC2378 I/O 口 P1.18-1.29 控制。液晶显示模块引脚描述如表 3-2 所示，E1 和 E2 引脚为读写使能引脚，A0 选择是指令还是送显的数据，若为高电平，则将数据显示；若为低电平，则设置显示器。

表 3-2 LCD12232-1 引脚描述

Table3-1 LCD12232-1 Pin Descriptions

| 引脚号 | 1 | 2 | 3 | 4 | 5-6 | 7 | 8 | 9-16 | 17-18 |
|---|---|---|---|---|---|---|---|---|---|
| 引脚名称 | VDD | VSS | Vo | /RESET | E1/E2 | R/W | Ao | D0-D7 | LED+/LED- |
| 引脚说明 | 电源 | 地 | LCD驱动电源 | 复位端 | 读写使能 | 读/写选择 | 数据/指令选择 | 数据输入输出 | -- |

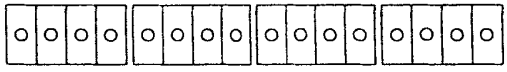
8 段码液晶显示屏由税控模块控制，税控模块主要由 8031 单片机、两输入非门 74HCT02N、静态移位寄存器 MC14562BCP 和两输入复用器 74HC157 组成。

## 3.5 数据采集模块、执行部件与电源模块

数据采集模块需要采集传感器脉冲信号、按键信号和油枪的开关信号。按键直接由 LPC2378 控制，传感器脉冲信号采集时需要进行光电隔离，采集的信号送入 LPC2378 I/O 口。

执行部件包括电磁阀和油泵开关。当数据采集模块采集到数据后，系统对数据进行分析处理，然后发出加油命令到执行部件，命令开关电磁阀和油泵。

电源模块引出 31 个引脚，引脚整体接线如图 3-6 所示，其中 7、8 引脚接电磁阀，20、21 引脚接电池。本方案中应用的是 AC380V 接法。



29 28 27 26 25 24 2322 21 2019 18 1716 15 14

AC380V接法：1，3，5脚接380V火线，空接380V零线；
2,4,6脚为输出，分别接电机的三根控制线；外置固态继电器控制正端接27脚，负端接28脚

AC22V接法：5脚接220V，空接AC零线，无固态继电器。

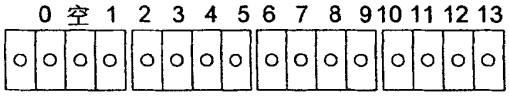



图 3-6 电源模块端子接线示意图

Fig.3-6 Terminal of the power supply module wiring diagram

## 3.6 本章小结

本章首先介绍了加油机控制装置的整体架构和整体硬件结构，然后介绍了装置每部分的器件选择和电路设计。

# 第 4 章 加油机控制装置的软件实现

## 4.1 加油机控制装置软件整体架构

本系统在 Keil μVision4 Realview MDK（Microcontroller Development Kit）[57]环境下，使用 ANSI C 语言[58]编程，利用 Socket API [59]设计开发加油机控制装置软件。加油机的主要功能包括加油流程及加油操作的实现和加油机与后台管理中心之间以及 IFSF 设备之间的 TCP/IP 通信。IFSF 加油机应用协议主要包括了加油机工作状态、加油机操作和数据库的定义，可以用于解析通信帧，以便实时监控加油进程；TCP/IP 通信的实现，主要是完成其它节点对本加油机数据库的读、写访问。加油机的内部结构框图如图 4-1 所示，其中，TCP/IP 提供网络服务，IFSF socket API 用来实现 IFSF 心跳和数据消息的发送和接收；控制及数据缓冲模块完成应用服务。

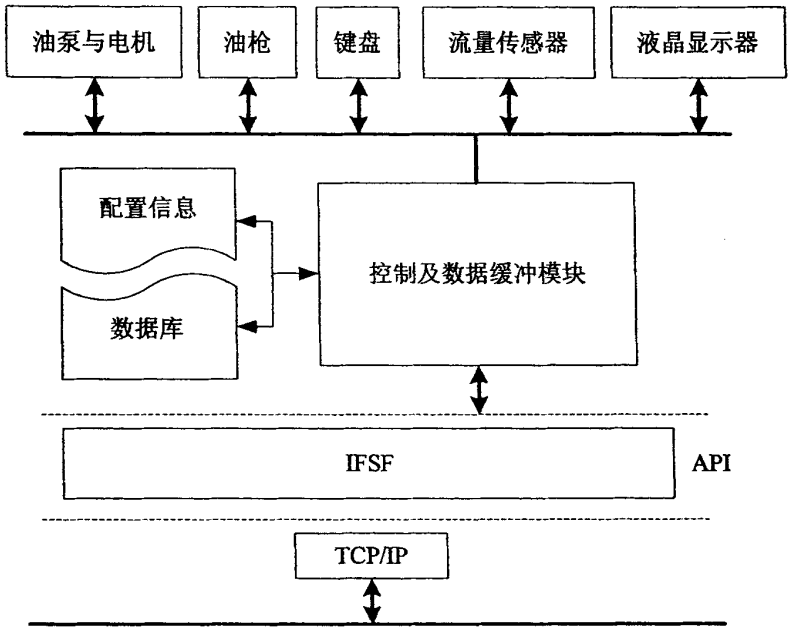



图 4-1 加油机内部结构框图

Fig.4-1 Internal block diagram of dispenser

基于 IFSF 的加油机控制装置可以通过高速以太网远程访问其他的 IFSF 设备，

后台管理中心也可以通过以太网访问和控制加油机，实现对加油机的实时监控。本加油机控制装置软件要实现的功能是加油操作，并将实时采集的数据加以处理，以 IFSF 包的格式封装到 TCP/IP 中发送到后台管理中心及其他设备，同时将通过 API 接口接收到的数据进行处理存储。加油机控制装置软件分为两部分，一部分是实现加油功能和加油操作，对加油交易的时间、数量、油品、枪号等信息进行管理；另一部分是实现网络通信，将得到的数据进行封装传送。在μC/OS-Ⅱ上以 7 个任务实现以上两部分功能，如图 4-2 所示，任务优先级从左向右依次降低，任务间通信是通过创建的两个消息队列 TCPCommMbox 和 TCPSendMbox 实现的。

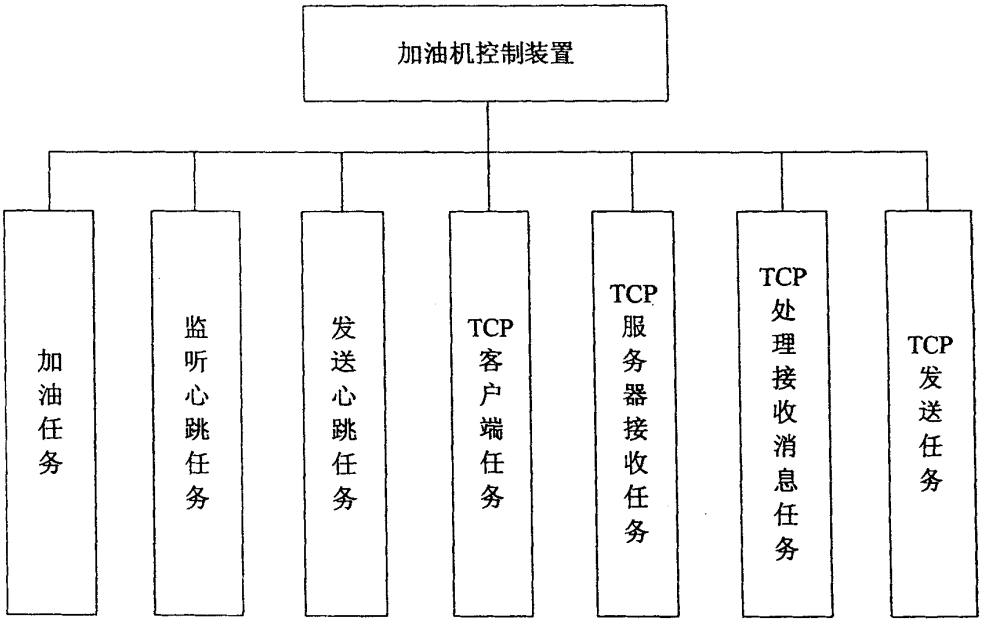



**图 4-2 加油机任务**

Fig.4-2 Dispenser tasks

加油机控制装置初始化流程设计如图 4-3 所示：初始化硬件配置时主要是初始化有关 LPC2378 的基本功能，然后初始化时钟节拍；初始化 TCP/IP 时会设置网卡 MAC 地址和本地 IP 地址；初始化 tcp 表，设置本机心跳信息；初始化加油机流程时，要初始化所有与加油有关的硬件如按键、液晶显示器、数据采集接口、铁电存储器等和加油机数据库以及加油机系统参数（包括加油点状态、油品单价等），然后等待正确开机信号来临时开机，将数据库中加油点状态 FPState 设置为“关闭”

状态，即加油机变为“关闭”状态。

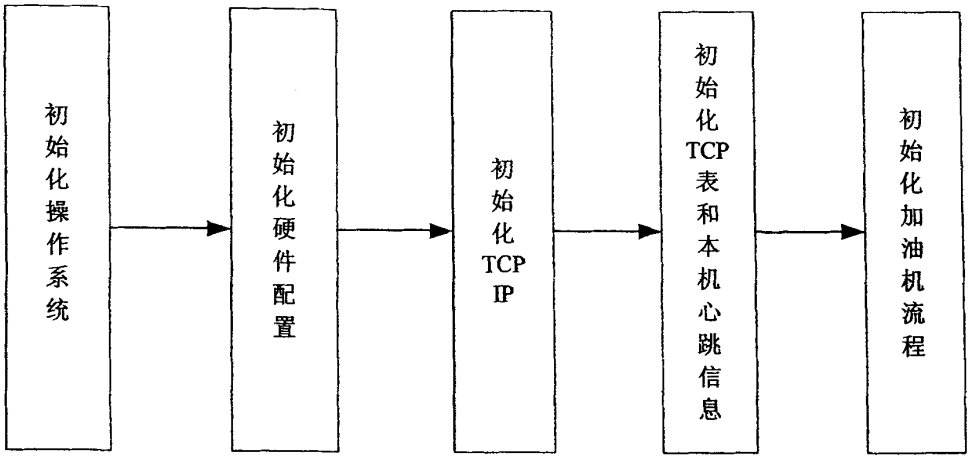



图 4-3 加油机初始化框图

Fig.4-3 Dispenser Initialize Diagram

## 4.2 加油机控制装置软件数据结构设计

根据 IFSF 协议实现原理，每个 IFSF 设备都有一个数据库，对加油设备的控制即为对数据库的访问读写。对加油机数据元素的访问是由数据库地址 DB_Ad 和数据标识 Data_Id 共同完成的。Data_Id 是数据库中数据元素的唯一标识，而这个数据库是由 DB_Ad 标识，即 DB_Ad 确定数据库的位置，Data_Id 确定数据库中数据元素的位置，由它来访问数据库中的数据元素。

### 4.2.1 IFSF 协议的主要数据结构

IFSF 信息包括 HeartBeat 和 Data Message。心跳消息的数据结构定义如下，因为实时性的要求，心跳消息要尽可能的小，用于更新 tcp 表。

```
struct HeartBeat
{
  long host_ip;//主机 IP
  uchar port[2];//端口号
  uchar lnao[2];//发送心跳地址
  uchar ifsf_mc;//消息类型，为 1 代表心跳
  uchar status;//设备的状态：bi8 =1（需更新）bit1=0（不具备运行的正确测试）；
```

```
};
Data Message 的数据结构定义如下：
typedef struct
{
  uchar lnars;//接收子网号
  uchar lnarn;//接收节点号
  uchar lnaos;//发送子网号
  uchar lnaon;//发送节点号
  uchar m_mc; //消息码
  uchar m_st;//发送消息类型
  uchar m_lg[2];//消息长度
  uchar db_ad_lg;//数据库地址长度
  uchar *db_ad;//数据库地址
  uchar *ifsf_data;//数据地址
}IFSFPacket;
```

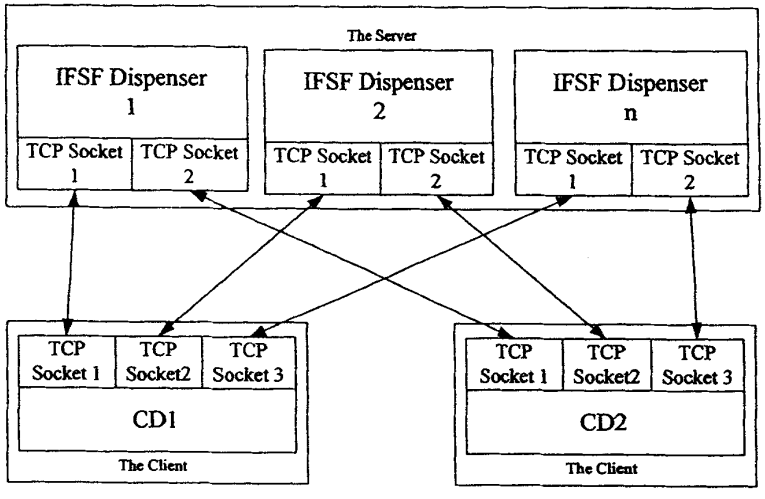



图 4-4 IFSF 设备间 TCP 连接方式

Fig.4-4 TCP connections between IFSF devices

IFSF 设备之间 TCP 连接方式如图 4-4 所示，本连接方式中一个应用程序对应一个 TCP 连接，响应迅速，只是占用资源较多。加油机控制装置的 tcp 表 tcp_table

定义如下。通过 IFSF 心跳在 tcp 表中添加 IFSF 设备时，根据发送心跳设置定时，只要 rest_time 大于 3 倍的心跳间隔，就说明该设备已离线，可以将该设备信息删除。每当 IFSF 加油机接到其它设备的心跳信息，就会更新 tcp 表中的在线标志位 flag 参数，在线将其置 1。IFSF 加油机控制装置要发送数据，先要建立 TCP 连接，TCP 连接建立后 send_sockedfd 和 rec_sockedfd 非 0。Node_Num 是 IFSF 节点数，更改 Node_Num 的值，可以相应增删 IFSF 加油机。

```
struct ifsf_tcp
{
  long ip;//IFSF 设备的 IP 地址
  int port;//IFSF 设备通迅的端口号
  uchar lnao[2];//发送心跳地址
  int send_socketfd;//网络连接设备描述符
  int rec_socketfd;
  uchar send_lan;
  uchar rec_lan;
  long rest_time;//未接到心跳消息时间
  long interval_time;//超时间隔
  uchar flag;//在线标志位
};
struct ifsf_tcp    tcp_table[Node_Num];
```

除此之外，还定义了无确认主动数据消息的格式，主要用于发送实时加油数据，有加油机状态主动消息、实时加油主动消息、交易缓冲区状态主动消息和交易错误四种主动消息。

### 4.2.2 加油机控制装置数据库设计

现在，嵌入式设备的内存和各种永久性存储介质的容量在不断增加，意味着嵌入式系统内部数据处理量的增加；而相对于嵌入式系统体积小的特点，传统的数据库技术体积大、延时长，不能满足嵌入式系统的开发[60]。而在本系统中，利

用数据结构 struct 实现加油机数据库，与μC/OS-II操作系统相配合，可以直接由应用程序访问而不需要单独运行数据库引擎。

在加油机软件中，根据 IFSF 设备协议，定义加油机数据库如下：

```
struct Dispenser_DB
{
 struct  COMMUNICATION_SERVICE_DATABASE  CService_Db;
 struct  CALCULATOR_DATABASE  Calculator_Db;
 struct  Fuelling_Point_Database FP_Data;          //21-24
 struct  Product_Database P_Db;                    //41-48h
 struct  Meter_Database  M_Db;                     //81h-90h
 struct  Product_per_Fuelling_Mode_Database Pro_Fue_Mode[10];
 struct  Logical_Nozzle_Database Log_No_Data;      //可以不用配置
};
struct Dispenser_DB DDB;
```

其中，COMMUNICATION SERVICE DATABASE，DB_Ad=00H，包括通信协议版本号、本地节点地址、接收地址表、心跳间隔和最大块长度。

CALCULATOR DATABASE，DB_Ad=01H，主要包括产品号、加油模式、加油点号、最大授权时间、最小加油量、最小显示体积和油品单位价格等。

Meter Database，DB_Ad=81H-90H，包括仪表类型（若为 0，表明未配置；若为 1，表明正常速度；若为 2，表明为高速）、连接到仪表的脉冲发生器产生的每一脉冲的体积（以十分之一毫升为单位）、仪表刻度因数和单脉冲计量总数。

Product Database，DB_Ad=41H~48H，参数为油品编码。

Product per Fuelling Mode Database，DB_Ad=PR_DAT (61H) + Prod_Nb (00000001-99999999) + FM_ID (11H-18H)，包括模式名称、产品价格、最大加油量等。

Fuelling Point Database，DB_Ad=21~24H，主要包括加油点名称、加油点状态、加油点的逻辑油枪数、默认模式等。

Logical Nozzle Database，DB_Ad=FP_ID (21H-24H) + LN_ID (11H-18H) ，主

要参数有产品标识（可以取值 0-8，0 代表未分配产品，1-8 分别代表 41-48H 为地址的数据库中的产品）、物理喷嘴标识、第一种基本产品标识、各逻辑喷嘴总体积、各逻辑喷嘴总额以及由该逻辑喷嘴提供的总额。

除此之外，还定义了 Fuelling Transaction Database、Error Code Database。为了方便记录和查询，将 IFSF 加油机协议中 Fuelling Point Database 的一部分单独定义为当前交易数据，定义如下：

```
struct CURRENT_TRANSACTION_DATA
{
  uchar Current_TR_Seq_Nb[2];// 29   (1DH)
  uchar Release_Contr_Id[2] ;   //30   (1EH)默认设置 00
uchar Suspend_Contr_Id[2];   // 31   (1FH) 默认设置 00
  uchar Current_Amount[5]; // 34   (22H)
  uchar Current_Volume[5]; // 35   (23H)
  uchar Current_Unit_Price[4]; // 36   (24H)
  uchar Current_Log_Noz; // 37   (25H)，逻辑枪号 1
  uchar Current_Prod_Nb[4];// 38   (26H)，油品编码 00000001
  uchar Current_TR_Error_Code;// 39   (27H)
  uchar Running_Transaction_Message_Frequency[2];   //60(3CH)
  uchar Date[4];              //202，自己增加 Data_ID
  uchar Time[3];              //203，增加 Data_ID
  uchar Pump_Total[7];
  uchar Open_FP; //60 (3CH)
  uchar Close_FP; //61(3DH)
  uchar Release_FP;// 62(3EH)
  uchar Terminate_FP;// 63(3FH)
  uchar Suspend_FP;// 64(40H)，保留
  uchar Resume_FP;// 65(41H)，保留
};
```

struct CURRENT_TRANSACTION_DATA Cu_Tr_Data;

## 4.3 加油任务实现

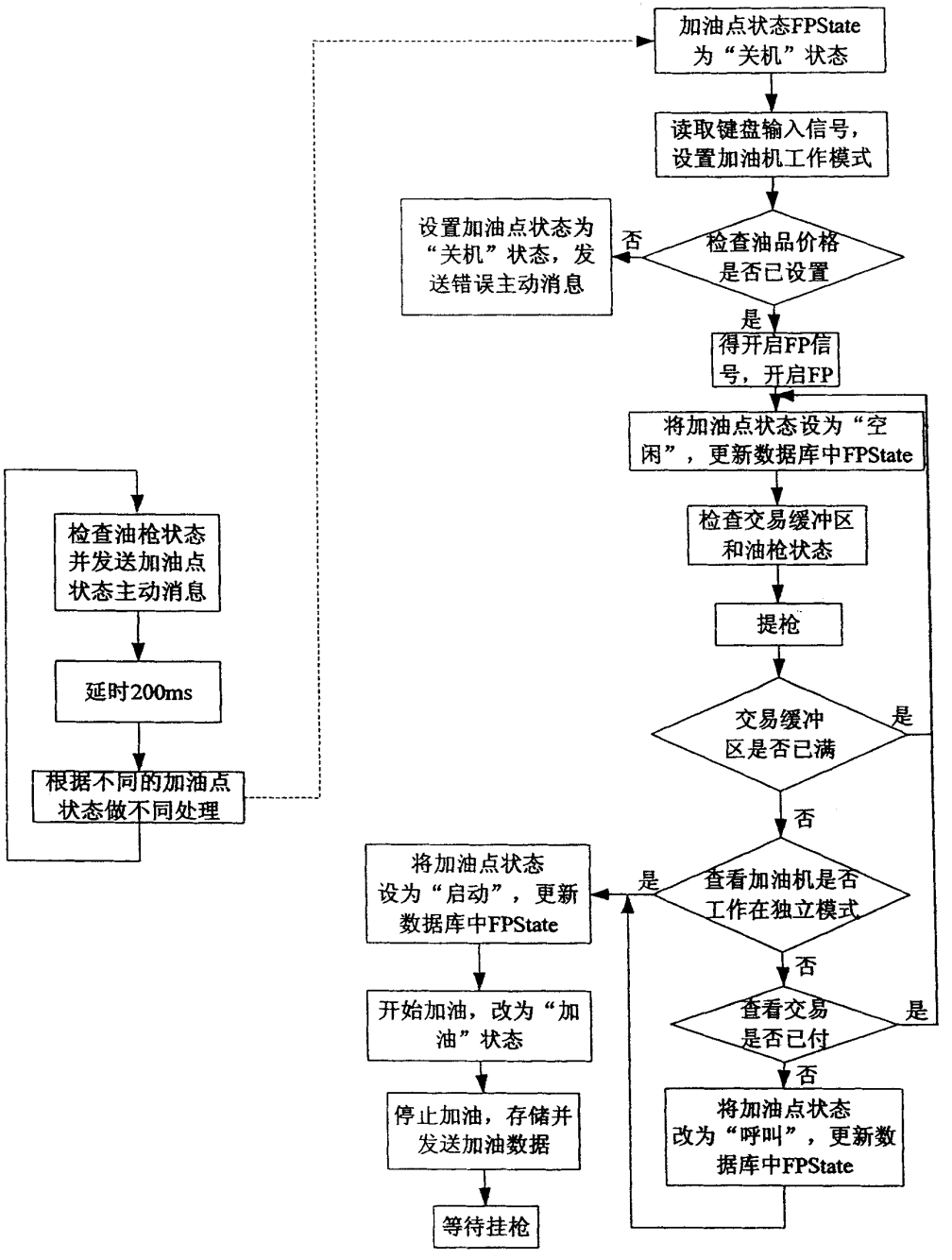



图 4-5 加油流程图

Fig.4-5 Fueling Flowchart

本装置中，根据国内石油零售业现状取消了加油点的“暂停加油”和“暂停启动”两个状态，其它的状态不变，加油任务流程设计框图如图 4-5 所示，加油任务的优先级为 1。加油机最初状态是“关机”，得到“Power on”信号时，发送开机命令；开机之后，加油点状态为“关闭”；“关闭”状态下，装置的字符液晶显示屏显示“加油机暂停工作”，读取键盘的内容，设置加油点工作在独立模式还是授权模式，检查交易缓冲区是否已满和油枪状态，在此期间，员工在编辑状态下可以查询油品、枪号等信息，等待开启 FP 信号来临，将加油点状态设为“空闲”；提枪后，如果加油机工作在独立模式，并且交易区未满，加油点状态将变为“启动”，等待开启油枪加油，如果加油机工作在授权模式，并且交易区未满，交易未预付，加油点状态改为“呼叫”，否则停止加油，等待挂枪，并返回“空闲”态；在“呼叫”状态下，如果设置了加油量，并打开了油泵，改为“启动”态；打开电磁阀，开启油枪，开始加油，将状态改为“加油中”；在加油过程中，发送状态变化主动数据消息，在定量加油结束后，发送加油数据，创建交易记录，检查交易缓冲区，等待挂枪。其中，要循环检查加油点的状态，在每一状态之后都要修改数据库中的加油点状态 FPState，并且在显示器上都会有相应状态提示；若是出现错误要发送错误主动消息。

## 4.4 TCP/IP 通信任务实现

TCP/IP 服务器和 UDP 服务器是可以同时实现数据收发的，在本设计方案中，将收发过程分开，分别建立任务，更加易于理解，便于调用。

加油机控制装置软件在设计时调用了 BSD Socket API 接口函数，UDP 和 TCP 各函数的功能及通信流程分别如图 4-6 和 4-7 所示。

UDP 客户端使用 sendto()通过套接字发送数据报给服务器，服务器用 recvfrom()等待，直到数据报到达，Recvfrom()向服务器返回客户端的信息，以方便服务器对请求作出响应并回复；TCP 是可靠性连接，服务器启动后，客户端使用 connect()与服务器使用 accept()创建的新的 socket 经三次握手建立连接，成功后向服务器发送请求，服务器处理该请求并回复，直到客户端发送特殊通知关闭连接，服务器会关闭该连接，等待新的连接建立。

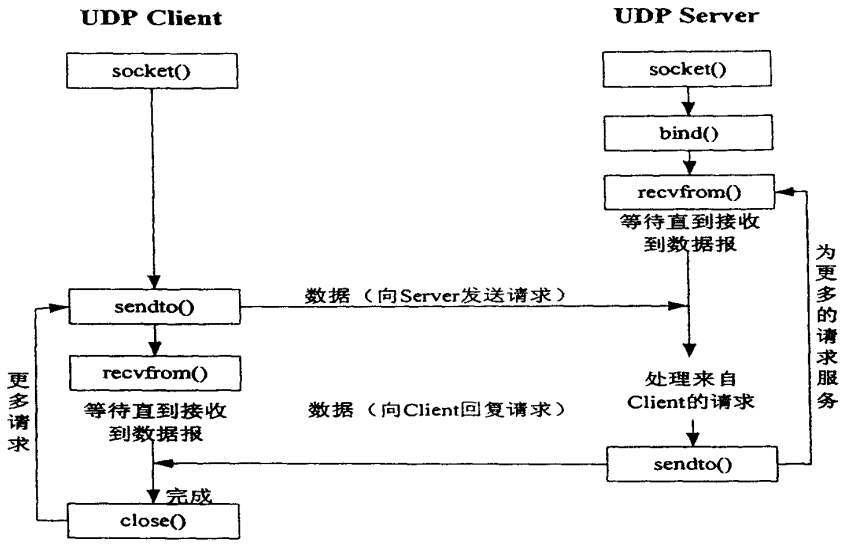



图 4-6 UDP Server-Client 通信模式

Fig.4-6 UDP Server-Client communication mode

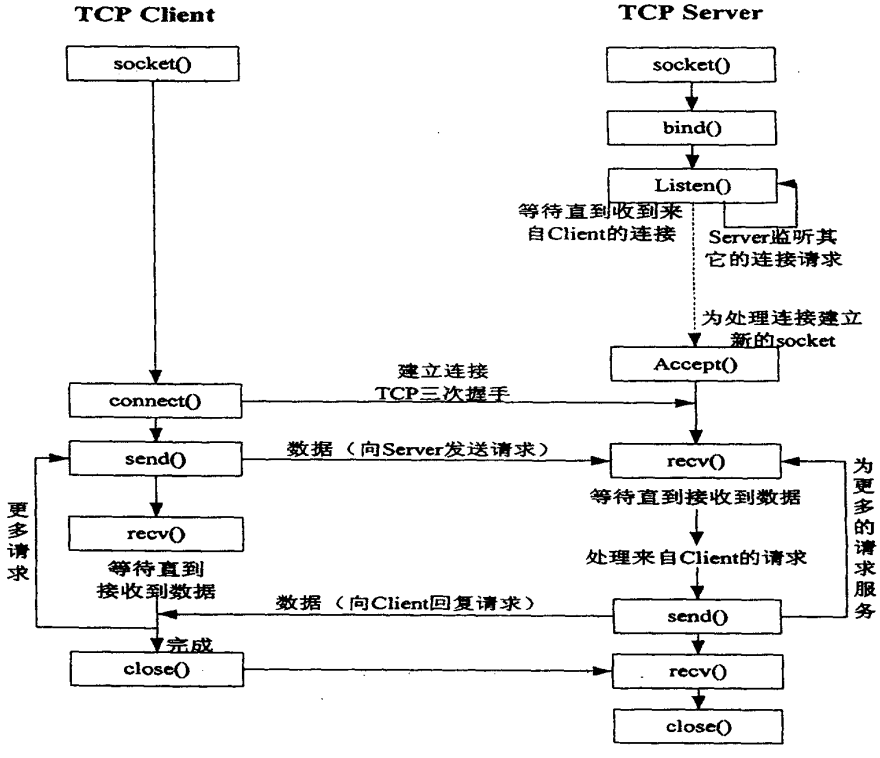



图 4-7 TCP Server-Client 通信模式

Fig.4-7 TCP Sever-Client communication mode

加油机软件 TCP/IP 通信模块按任务分，包括 TCP Server、TCP Client、UDP Server 和 UDP Client 四部分：以 TCP Server 实现数据消息的发送；以 TCP Client 实现数据消息的接收；以 UDP Server 实现心跳消息的发送；以 UDP Client 实现心跳消息的接收。通信模块的实现包括 6 个任务：发送心跳任务（相当于 UDP Client，任务优先级为 11）、监听心跳任务（相当于 UDP Server，任务优先级为 10）、TCP Client 主动建立连接并接收 Data Message 任务（TCPClient，任务优先级为 12）、TCP 发送 Data Message 任务（TCPSend 任务，任务优先级为 15）、TCP Server 监听建立连接并接收 Data Message 任务（TCPServerRec 任务，任务优先级为 13）、TCP 处理 Data Message 任务（任务优先级为 14）。CPU 调度转换任务，实现多任务运行。μC/OS-II 是抢占式实时多任务内核，在抢占式多任务处理过程中，对任务进行调度，每段时间内只服务于一个任务；在任务切换时要进行前一个任务的现场保护，并恢复下一个要执行任务的寄存器内容[61]。

本模块中， IFSF Data Message 和 IFSF HeartBeat 使用 TCP/IP 协议族来传输，Data Message 使用 TCP，HeartBeat 使用 UDP，通过 Socket API 来访问。

1) 监听心跳任务

心跳的监听是本模块的一个非常重要的部分，因为它决定着 tcp 表 tcp_table 的更新，决定着通信是否能够畅通。UDPServer 任务如图 4-8 所示，UDP Server 绑定 socket 等待接收来自其它设备的心跳消息，接收到 UDP 数据报后要查看是否是心跳消息：如果不是心跳消息，继续等待心跳消息的到来；如果是心跳消息，就查看发送该心跳消息的设备是否在 tcp_table 中已记录，如果没有记录就在 tcp_table 中找到空闲位置记录该设备的信息，然后将对应的在线标志位 flag 置 1。原本在收到心跳消息时，就应该启动定时设置，以判断没有接收到该设备心跳的时间是否超出了 3 倍的心跳间隔，但是由于心跳的接收没有规律，如果每接收到心跳消息就启动定时，非常难以管理。而发送心跳总是非常有规律的，加油机控制装置每隔 10s 就会广播一次心跳，将定时功能的实现放在发送心跳 UDPClient 任务中会非常简便，也不会影响系统功能。

2) 发送心跳任务

发送心跳任务如图 4-9 所示，每隔 10s 发送一次心跳消息，j 的定义是为了实现定时设置：在心跳间隔这 10s 中，每延时 1s 将 tcp_table 中在线设备（即 flag=1 的设备）的 rest_time 加 1 并与 interval_time 比较；初始化时，将 interval_time 设为 30，即 3 个心跳间隔 30s 内若没有接到来自该设备的心跳消息则视该设备已离线，不能与其通信，将该设备的信息从 tcp_table 中删除，也就是以查询的方式实现设备在线状态的检测，当 rest_time 大于超时间隔 interval_time 时，删除该设备信息。具体 C 语言实现如下：

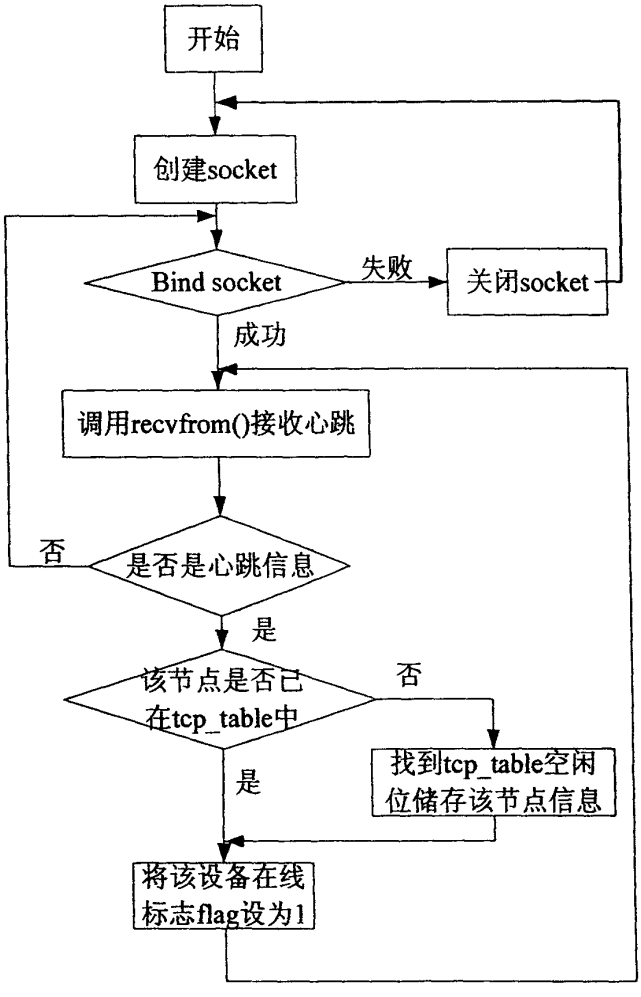



图 4-8 监听心跳任务

Fig.4-8 HeartBeat-Listening Task

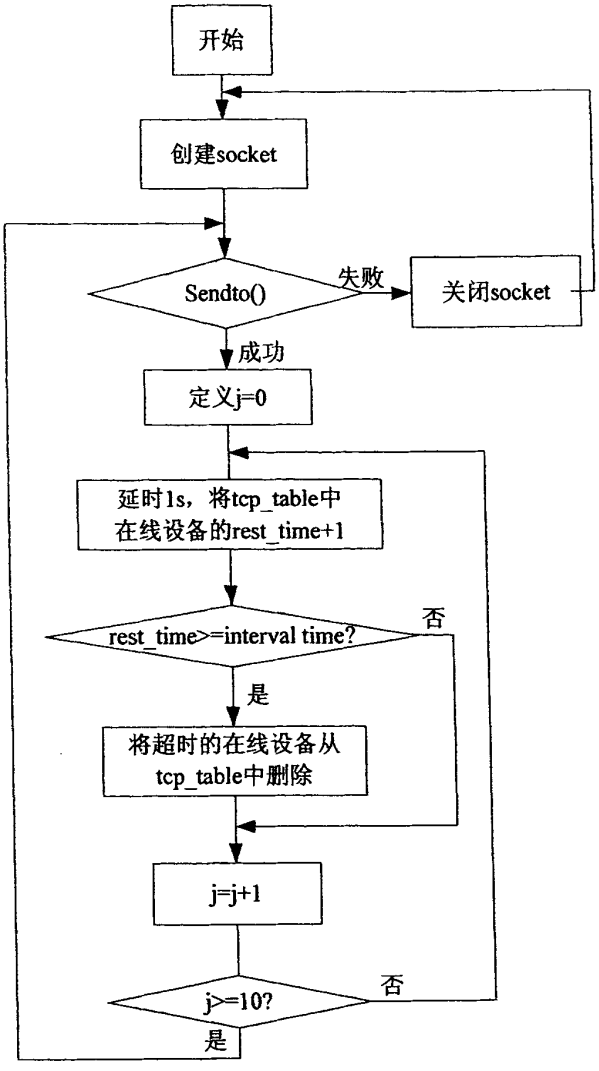



图 4-9 发送心跳任务

Fig.4-9 HeartBeat-deliverring Task

```
CPU_CHAR hb_intervaltime=10;
for(j=0;j<hb_intervaltime;j++)
    {   OSTimeDlyHMSM(0,0,1,0);
        for(i=0;i<Node_Num;i++)
        {   if(tcp_table[i].flag==1)
          {   tcp_table[i].rest_time++;
              if(tcp_table[i].rest_time>=tcp_table[i].interval_time)
```

```
{ tcp_table[i].flag=0;
  tcp_table[i].rec_socketfd=0;
 tcp_table[i].send_socketfd=0;
  tcp_table[i].port=0;
  tcp_table[i].lnao[0]=0;
  tcp_table[i].lnao[1]=0;
  tcp_table[i].ip=0;}   }   }   }
```

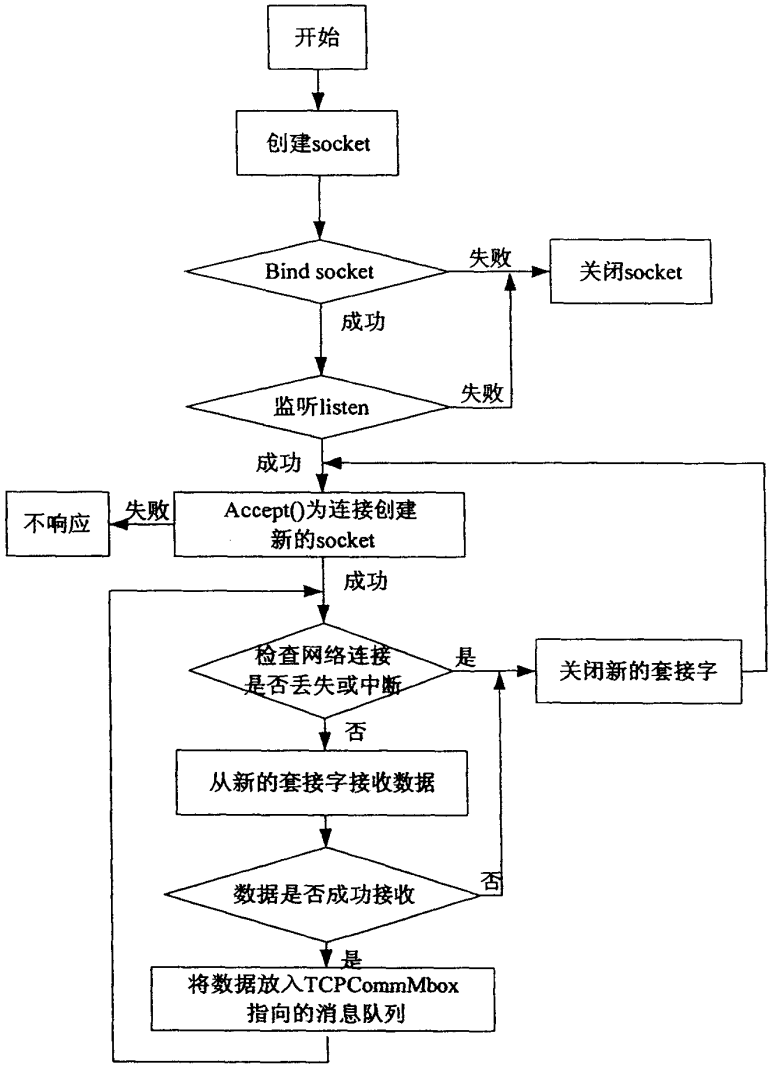



图 4-10 TCPServerRec 任务

Fig.4-10 TCPServerRec Task

3) TCPServerRec 任务

TCPServerRec 任务如图 4-10 所示，加油机（TCP Server）监听来自其他设备（TCP Client）的连接，并调用 accept()为该连接创建新的 socket，在网络连接没有中断的情况下，从新的 socket 接收数据并将接收到的数据送入 TCPCommMbox 所指的消息队列中，等待处理。

4) TCP 处理消息任务

Data Message 处理流程如图 4-11 所示，将在 TCPServerRec 任务接收的消息即 TCPCommMbox 指向队列中的数据进行校验，校验无错则加以处理，处理完毕后将需要发送的回复消息传给 TCPSendMbox 指向的队列，每间隔 1s 查询一次是否有数据到来。

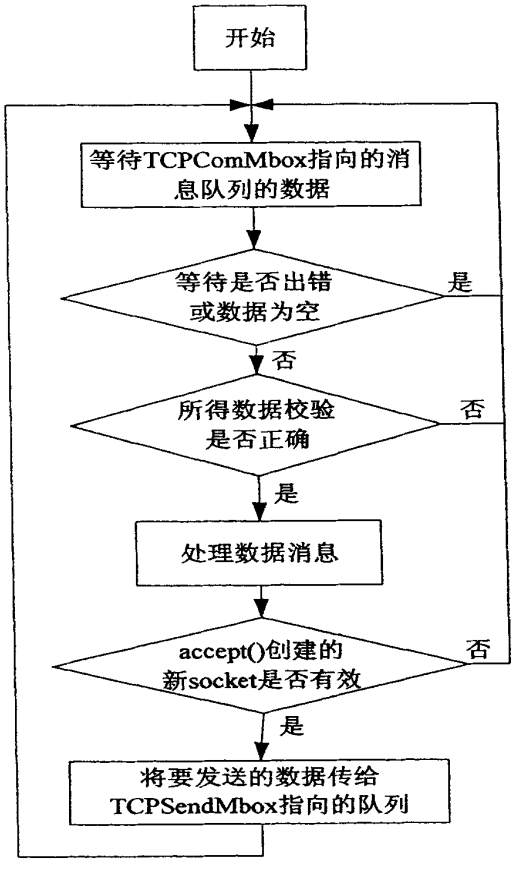



图 4-11 TCP 处理消息任务

Fig.4-11 TCP handle Data Message Task

5) TCPClientSend 任务

TCPClientSend 任务如图 4-12 所示，实现的是将数据通过 TCP 发送出去。从 TCPSendMbox 指向的消息队列中提取数据，若该数据有效，则通过已建立的连接发送出去。在发送之前，要先检查 tcp_table 中非独立工作模式标志 send_lan 是否置位，若置位，说明工作在授权模式下，TCP/IP 连接已建立，可以发送数据。

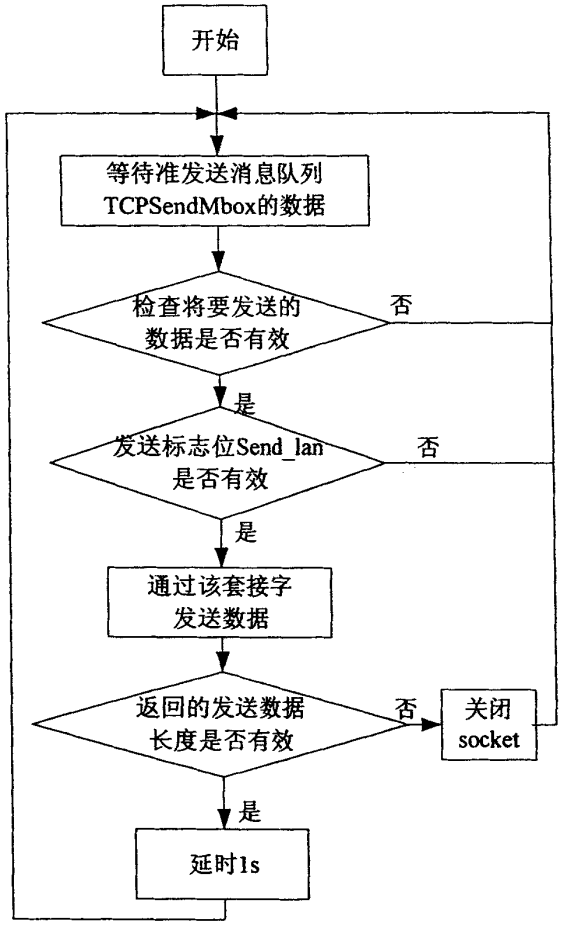



图 4-12 TCPClientSend 任务

Fig.4-12 TCPClientSend Task

6) TCPClient 任务

TCPClient 任务如图 4-13 所示。先检查设备是否在线，如果不在线，延时 50ms 后再查看，直到该设备在线，向该设备（相当于服务器）创建连接，看网络连接

是否中断，若没有，就从该套接字接收来自该设备的数据消息以便进行处理。

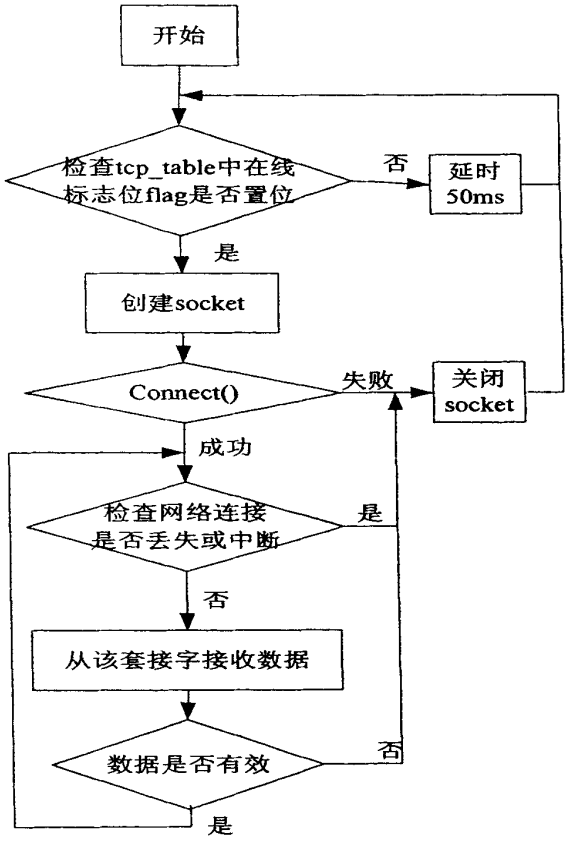



图 4-13 TCPClient 任务

Fig.4-13 TCPClient Task

## 4.3 本章小结

本章在之前介绍的 IFSF 协议的 TCP/IP 实现的基础上，结合实际给出了软件具体的实现流程，包括加油流程和 TCP/IP 通信流程，并列出了 IFSF 协议主要的数据结构并进行了深入的分析，给出了软件实现的基本方案。

# 第 5 章 系统测试

## 5.1 测试连接与测试方法

为测试加油机控制装置的性能，在完成嵌入式软件开发后，在实验室内搭建简易平台，将系统机和加油机控制装置通过 Internet 互联，完成加油操作及通信测试。测试连接示意图如图 5-1 所示。系统机向加油机控制装置发送心跳消息和数据消息，加油机控制装置接收后返回消息；加油机控制装置完成加油操作，向系统机发送主动消息。

在系统机上利用 C++ Builder 6.0 软件开发设计加油机控制装置测试软件，实现通过以太网收发数据包[62]。其界面如图 5-2 所示，UDP 受控机器文本框填写受控加油机控制装置 IP；本机（即系统机）IP 文本框填写本地 IP ；端口号文本框输入受控加油机心跳端口号，以实现心跳测试功能；界面第三排第一个文本框是命令窗口（以十六进制显示，能够方便知道命令构成），点击每条命令，该命令会被复制到 HEX 发送文本框内，点击 HEX 发送按钮，将数据发送；第三排第二个文本框是显示心跳的收发文本框；第五排的接收文本框内主要是实时数据接收显示；可以校时，可以读时间，并能清空记录。发送的测试数据完全遵照 IFSF 协议规定的格式。

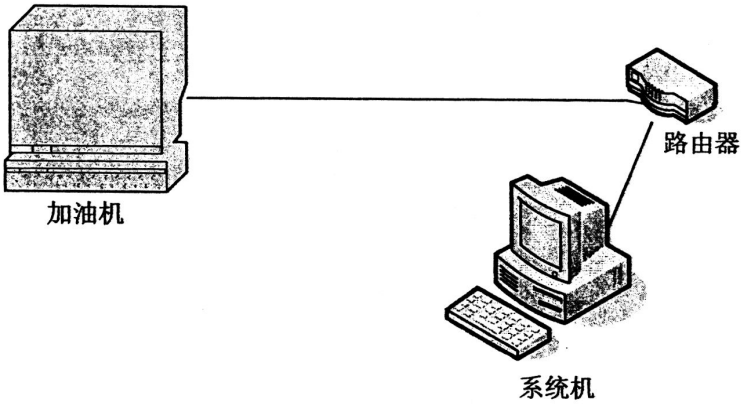



图 5-1 测试连接示意图

Fig.5-1 Connection of test system

将加油机控制装置和系统机连接到路由器后，IP 设置如表 5-1 所示，加油机 IP 是在加油机程序刷新时设置的。

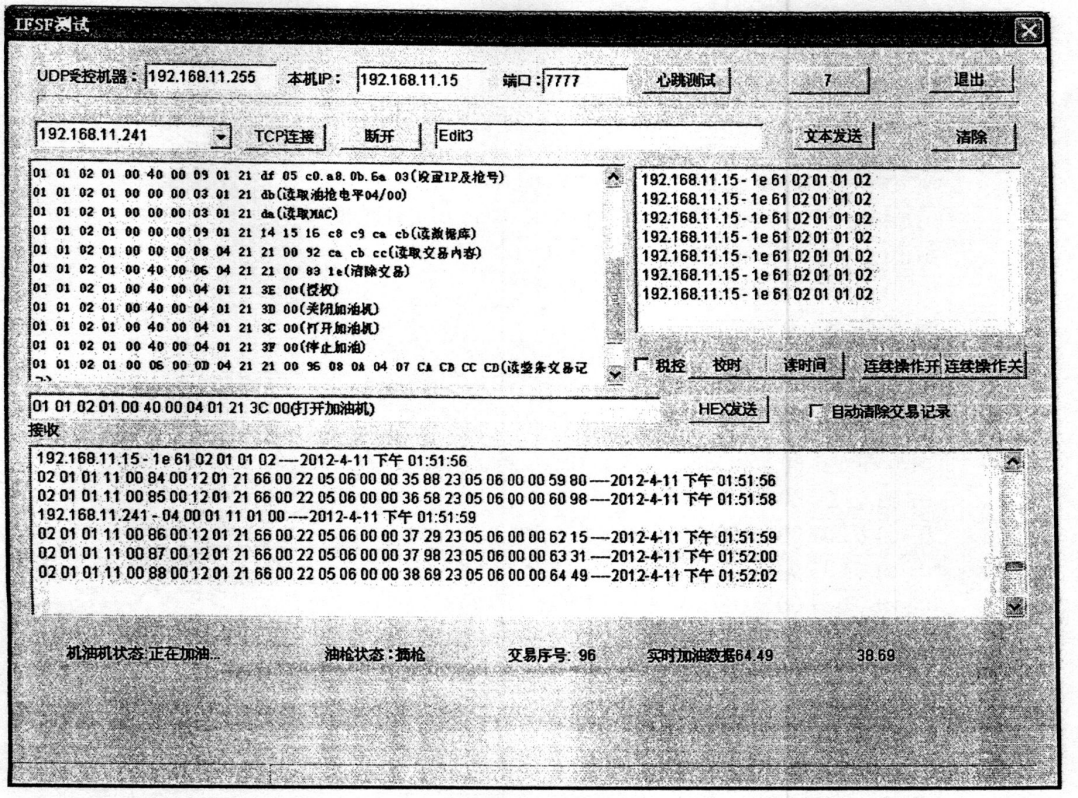



图 5-2 加油机测试软件界面

Fig.5-2 The interface of dispenser test software

表5-1 加油机与系统机IP设置

Table5-1 The IP config of dispenser and system computer

| 系统机 | | 加油机IP | |
|---|---|---|---|
| IP | 端口号 | IP | TCPClient端口号 |
| 192.168.11.15 | 7777 | 192.168.11.241 | 1024 |

IP 设置好后，点击心跳测试，会每隔 10s 发送一次心跳，心跳的收发文本框内会显示发送心跳内容，具体说明如表 5-2 所示。

表5-2 心跳帧格式说明

Table5-2 The description of HeartBeat Frame Format

| 帧内容 | 192.168.11.15 | 1e 61 | 02 01 | 01 | 02 |
|---|---|---|---|---|---|
| 说明 | 主机IP（c0 a8 0b 0f） | 主机端口号（7777） | 本地节点地址（子网号02+节点号01） | ifsf_mc（心跳） | 可以连接 |

如果网络通信畅通，心跳发送成功后，会接到加油机控制装置返回的心跳信

息，并且显示客户端连接成功。点击 TCP 连接，建立 TCP 连接后就可以与加油机控制装置通信了。系统机可以对加油机控制装置操作，发送的命令如表 5-3 所示。发送命令完全按照 IFSF 通用 Data Message 格式。

表5-3 发送命令格式

Table5-3 The delivering command format

| 序号 | 消息帧（十六进制） | 命令说明 |
|---|---|---|
| 1 | 01 01 02 01 00 40 00 08 01 41 02 04 00 00 00 02 | 设置油品 |
| 2 | 01 01 02 01 00 40 00 0D 06 61 00 00 00 02 11 02 04 04 00 06 80 | 设定油价 |
| 3 | 01 01 02 01 00 40 00 06 01 21 16 02 00 00 | 设置联机 |
| 4 | 01 01 02 01 00 40 00 06 01 21 16 02 ff ff | 设置脱机 |
| 5 | 01 01 02 01 00 40 00 03 01 21 c8 | 提取脱机交易 |
| 6 | 01 01 02 01 00 40 00 07 01 21 ed 03 a1 a2 a3 | 启动isp |
| 7 | 01 01 02 01 00 40 00 0a 01 21 da 06 00-1d-12-e0-1f-0f | 设置MAC |
| 8 | 01 01 02 01 00 40 00 05 01 21 db 01 04 | 设置油枪电平 |
| 9 | 01 01 02 01 00 40 00 09 01 21 df 05 c0 a8 0b 6a 03 | 设置IP及枪号 |
| 10 | 01 01 02 01 00 00 00 03 01 21 db | 读取油枪电平 |
| 11 | 01 01 02 01 00 00 00 03 01 21 da | 读取MAC |
| 12 | 01 01 02 01 00 00 00 09 01 21 14 15 16 c8 c9 ca cb | 读数据库 |
| 13 | 01 01 02 01 00 00 00 08 04 21 21 00 92 ca cb cc | 读取交易内容 |
| 14 | 01 01 02 01 00 40 00 06 04 21 21 00 83 1e | 清除交易 |
| 15 | 01 01 02 01 00 40 00 04 01 21 3E 00 | 授权 |
| 16 | 01 01 02 01 00 40 00 04 01 21 3D 00 | 关闭加油机 |
| 17 | 01 01 02 01 00 40 00 04 01 21 3C 00 | 打开加油机 |
| 18 | 01 01 02 01 00 40 00 04 01 21 3F 00 | 停止加油 |
| 19 | 01 01 02 01 00 06 00 0D 04 21 21 00 96 08 0A 04 07 CA CB CC CD | 读整条交易记录 |

以第 17 条打开加油机命令为例，具体说明如表 5-4 所示，M_St 前三位代表消息类型，后五位代表消息的序号；消息长度为几，其后就跟有几个字节；数据库地址长度为 n，则其后 n 个字节为数据库地址，该命令里数据库地址为 21H，即加油点数据库；而 Data_Id 代表数据库中 3CH 位置的元素，为 Open_FP；该消息是“写消息”，对加油机控制装置数据库进行写操作来操作加油机控制装置。

表5-4 打开加油机命令格式说明

Table5-4 The description of the ‘open dispenser’ command format

| 01 02 | 02 01 | 00 | 40 | 00 04 | 01 | 21 | 3C | 00 |
|---|---|---|---|---|---|---|---|---|
| 接收节点地址 | 发送节点地址 | ifsf_mc由应用层处理 | M_St写消息、序号为0 | 消息长度为4 | 数据库地址长度为1 | 数据库地址 | Data_Id数据库中位置 | 数据长度为0 |

在测试的命令中，第 1-9 和 14-18 条是“写消息”，通过改变数据库元素值来操作加油机控制装置，其中，启动 isp 命令是要等待系统机对加油机控制装置软件

进行刷新；第 10-13 和第 19 条是“读消息”，加油机控制装置接收到该“读消息”后从相应数据库中读取数据并发送给系统机，返回的数据严格依照 IFSF 协议规定。

如图 5-3 所示，是在结束加油过程中的数据采集图。在授权状态下，提枪并开启油枪后开始加油，加油机控制装置创建实时加油主动消息发送给系统机，如接收文本框前两条所示；在挂枪时，加油机控制装置创建加油机交易缓冲区状态主动消息，如接收文本框第 4 条消息所示，然后发送加油机状态主动消息，为“空闲”态，如接收文本框最后一条所示。分别进行定量加油和定额加油，所得交易数据如表 5-5 所示，其中油价设为 6.8 元/L，定量加油分别加油 10L、50L 和 56L，定金额加油分别为 100 元、250 元和 255.5 元时采集的数据，第 1 个 06 00 后跟的 3 个字节是加油金额，第 2 个 06 00 后跟的是加油升数，以十进制读取，3 个字节中最后一个字节代表小数部分。

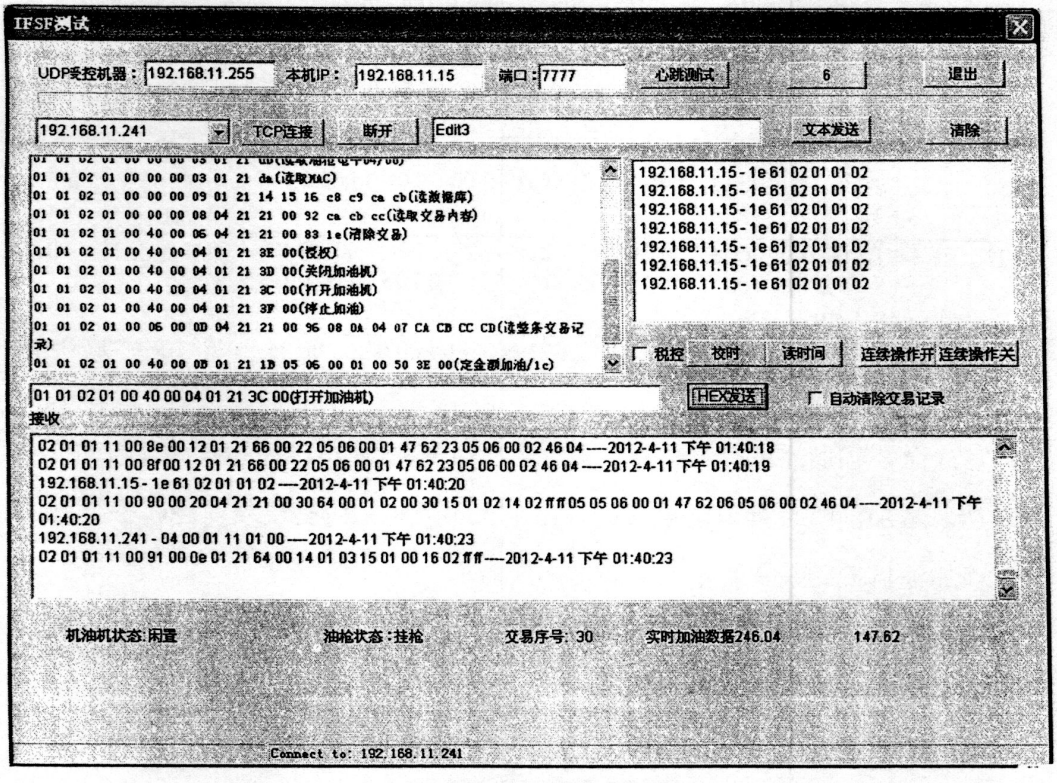



图 5-3 挂枪时加油机发送消息

Fig.5-3 The Data Message Dispenser delivered when Nozzle down

## 5.2 测试结果与系统性能分析

心跳测试，根据实时时间显示，每隔 10s 就收到一次加油机控制装置发来的心跳，而且系统机本身每隔 10s 发送一次心跳。当系统机发送命令使加油机控制装置工作在独立模式，实时加油数据存储在未发送交易存储区，发送命令可以查询到交易内容；当工作在联机模式下，加油过程正常实现，加油机控制装置加油过程的状态变化以及交易数据会主动发送给系统机接收。

表5-5 加油交易数据

Table5-5 The Fuelling Transaction Data

| 体积(L)/金额(元) | 加油交易数据 | 说明 |
|---|---|---|
| 10L | 02 01 01 11 00 82 00 20 04 21 21 00 35 64 00 01 02 00 35 15 01 02 14 02 ff ff 05 05 06 00 00 68 00 06 05 06 00 00 10 00 | 第35笔交易，加油10L，共68元 |
| 50L | 02 01 01 11 00 86 00 20 04 21 21 00 37 64 00 01 02 00 37 15 01 02 14 02 ff ff 05 05 06 00 03 40 00 06 05 06 00 00 50 00 | 第37笔交易，加油50L，共340元 |
| 56L | 02 01 01 11 00 8a 00 20 04 21 21 00 39 64 00 01 02 00 39 15 01 02 14 02 ff ff 05 05 06 00 03 80 80 06 05 06 00 00 56 00 | 第39笔交易，加油56L，共380.8元 |
| 100元 | 02 01 01 11 00 8e 00 20 04 21 21 00 41 64 00 01 02 00 41 15 01 02 14 02 ff ff 05 05 06 00 01 00 00 06 05 06 00 00 14 71 | 第41笔交易，加油金额100元，共14.71L |
| 250元 | 02 01 01 11 00 92 00 20 04 21 21 00 42 64 00 01 02 00 42 15 01 02 14 02 ff ff 05 05 06 00 02 50 00 06 05 06 00 00 36 76 | 第42笔交易，加油金额250元，共36.76L |
| 255.50元 | 02 01 01 11 00 96 00 20 04 21 21 00 44 64 00 01 02 00 44 15 01 02 14 02 ff ff 05 05 06 00 02 55 50 06 05 06 00 00 37 57 | 第44笔交易，加油金额255.5元，共37.57L |

发送命令到加油机控制装置，会收到加油机控制装置返回的数据，TCP/IP 通信成功。本加油机装置现场连续运行二十天不死机，没有出现问题。根据测试结果，该加油机控制装置基本能够满足用户需求。

由此可见，本设计方案在实际应用中比较合理，具有可行性，对中国加油站管理向 IFSF 架构发展有着重大意义。

## 5.3 本章小结

本章主要介绍了对系统的测试，说明了测试方法，给出了测试结果，并对系统性能进行了分析评价。

# 结论

本文对基于 IFSF 协议的加油机控制装置进行研究，该装置能够使加油站的网络信息化管理更加简便，论文的主要研究工作如下：

1) 分析了加油站互联系统现有的四种方式，我国主要应用的是前庭控制器连接方式，研究了该连接方式的弊端。IFSF 架构的加油站互联系统是加油站信息管理化发展的一个方向，因此提出了基于 IFSF 协议的加油机控制装置的设计方案，方案的实行使 IFSF 架构的推广成为可能，设计的加油机控制装置不但基于 IFSF 协议，而且能够实现网络通信，具有先进性。

2) 设计了嵌入式加油机控制装置的硬件结构；根据系统要求选择芯片，提出以微处理器 LPC2378 为核心芯片，以 DP83848 为物理层芯片的网络通信硬件设计方案。

3) 仔细研究了 IFSF 协议中的加油机协议和通信协议，根据加油机协议中的加油点状态图，总结加油机工作流程、加油机数据结构和数据库，以及 IFSF 的 TCP/IP 实现，为后续软件实现打下基础。

4) 为了实现 TCP/IP，体现 ARM 的优势，选择在 ARM 上移植操作系统，使得编程更加简便。因为 TCP/IP 通信对实时性要求比较高，所以选择实时性内核的 μC/OS-Ⅱ操作系统，并完成了 μC/OS-Ⅱ和 TCP/IP 协议栈在 LPC2378 上的移植。

5) 进行嵌入式软件设计，包括数据结构、加油操作的实现和通信任务的设计，并进行测试，主要是 IFSF 的 TCP/IP 实现，验证了本方案的可行性。

采用基于 IFSF 架构的加油站互联系统是石油零售产业互联系统未来发展的方向，基于 IFSF 协议的前庭设备的研究很重要。在嵌入式技术兴盛和高速发展的今天，以 ARM 为核心、采用嵌入式操作系统并实现网络通信的嵌入式系统的研究给了我们一种机遇。IFSF 协议在国际上应用广泛，我国加油站互联多采用前庭控制器连接方式，将多种多样的加油机协议转换成 IFSF 协议，并在前庭控制器部分实现网络通信。目前很多人对 IFSF 架构的加油站互联系统进行了分析，而基于 IFSF 协议的前庭设备的研究并不是很多。由于基于 IFSF 的前庭设备便于管理和维护，

虽然将非 IFSF 加油机全部换成基于 IFSF 的加油机会是一笔不小的支出，但是投入使用后会大大降低维护成本。

在本论文完成期间，由于时间紧迫，加油机数据库很多，并不能完全进行分析，只实现了加油机数据库的主要参数。对于数据采集部分和 TCP/IP 协议栈的移植研究不深，主要实现了网络通信功能和 μC/OS-II 的移植。再者，如果提升 ARM 内核的级别，处理速度上会更上升。而且，由于模拟环境与现实环境的差别，软件在使用过程中还需要进行不断地改善。

# 参考文献

[1] 王爱学.加油站设备通信协议转换技术的研究[D].硕士学位论文,重庆大学,2009.

[2] 许涛,朱斌,马军强.加油机协议转换为 IFSF 协议方案的设计与实现[J].中国信息界,2011,(6):56-58.

[3] 马晓鸿,和冬梅,程晓春. IFSF 在中国石油加油站管理系统中的应用探讨[J].石油规划设计,2010,21(3):28-30.

[4] 蔡丽娟.基于 IFSF 的加油站互联系统研究与设计[D].硕士学位论文,中国石油大学,2010.

[5] 任旭虎,蔡丽娟,杨磊.加油站前庭控制器研究与实现[J].工业仪表与自动化装置,2010,(3):37-42.

[6] International Forecourt Standards Forum. Controller Device [S]. V3.10, 2006-07.

[7] 张静,张凯.实时操作系统μC/OS-Ⅱ在 ARM7 上移植的研究与实现[J].计算机工程与应用,2004,(4):100-102,153.

[8] International Forecourt Standards Forum. IFSF Management Overview[S]. 2005-01.

[9] 向婕.嵌入式加油站前庭控制器研究与设计[D]. 硕士学位论文,重庆大学,2008.

[10] 刘运基.IFSF 和中国石油零售行业.加油站服务网,2007-07-03.

[11] International Forecourt Standards Forum. IFSF Management Introduction[S]. 2004-05.

[12] International Forecourt Standards Forum.COMMUNICATION SPECIFICATION OVER LONWORKS [S]. VERSION 1.91,2006-03.

[13] 阳宪惠.现场总线技术及其应用[M].北京:清华大学出版社,2000.11-160.

[14] 俞洪.LonWorks 技术与控制网络[J].TECHNOLOGY WIND,2011,(20):16-17.

[15] 黄河,郭海涛.LonWorks 总线的智能建筑一体化控制系统探究[J].河南科技,2011,(20):45.

[16] 罗军舟,黎波涛,杨明等.TCP/IP 协议及网络编程技术[M].北京:清华大学出版

社,2004. 6-9,168-209.

[17] 石永财.基于 IFSF 协议的加油站前庭控制器设计研究[D].硕士学位论文,重庆大学,2007.

[18] International Forecourt Standards Forum. Communications Specification[S]. Version1.80, 2000-02.

[19] International Forecourt Standards Forum. TANK LEVEL GAUGE APPLICATION[S]. VERSION 1.28, 2007-05.

[20] International Forecourt Standards Forum.PRICE POLE APPLICATION[S]. VERSION1.14, 2005-07.

[21] International Forecourt Standards Forum.CAR WASH APPLICATION[S]. VERSION1.32, 2003-12.

[22] International Forecourt Standards Forum.DISPENSER APPLICATION[S]. VERSION2.25, 2007-11.

[23] W. Richard Stevens.TCP/IP 详解卷 1:协议[M].范建华,胥光辉,张涛等译. 北京:机械工业出版社,2007.1~14.

[24] International Forecourt Standards Forum. Communications Specification over TCP/IP[S]. Version1.01,2002-02.

[25] 邱铁.ARM 嵌入式系统结构与编程[M].北京:清华大学出版社,2009.1-7.

[26] Filman Robert E. Embedded Internet Syetems Come Home[J]. IEEE Internet Computing, 2001,5(1):52-53.

[27] Guan Mo, Han Guangjie, Zhang Wenbo, etal. WebitX: A Real-Time Operating System for Embedded Internet Applications[J]. Journal of Northeastern University, 2004,25(7):649-652.

[28] 孙秋野,孙凯,冯健.ARM 嵌入式系统开发典型模块[M].北京:人民邮电出版社,2007.3-7.

[29] Jean J. Labrosse. MicroC/OS- II :The Real-Time Kernel(Second Edition) [M].LSA:CMP Books,2002.1-20.

[30] 姚天祥,李华贵, 孔若英, 樊丽丽. μC/OS-II在 LPC2148 上的移植[J].电子元器件应用,2007, 9(7):15-16,20.

[31] 马立国.嵌入式实时操作系统μC/OS-Ⅱ在 ARM 上的移植[D].硕士学位论文,吉林大学,2006.

[32] 丁国超.μC/OS-Ⅱ实时操作系统在 ARM 微处理器上的移植[D].硕士学位论文,哈尔滨理工大学,2005.

[33] 陈是知.μC/OS-II 内核分析、移植与驱动程序开发[M]. 北京:人民邮电出版社,2007.127-150.

[34] 杨晔.实时操作系统μC/OS-Ⅱ下 TCP/IP 协议钱的实现[J].单片机与嵌入式系统应用,2003,(7):80-83.

[35] Haque Tariqut, Urbaniak Michael. Step by Step Implementation of Ethernet and TCP-IP Protocols on an embedded System[J]. Computers in Education Journal.2004,14 (4):70-87

[36] Oh Soo-Cheol, Janq Hankook, Chung Sang-Hwa. Analysis of TCP/IP Protocol Stack for a Hybrid TCP/IP Offload Engine. 5th International Conference, Singapore, 2004:406-409.

[37] Suwartadi Eka, Gunawan Candra, Steijadi P. Ary, etal. First Step toward Internet Based Embedded Control System. 2004 5th Asian Control Conference, Melbourne Australia, 2004:1226-1231.

[38] Adam Dunkels. Design and Implementation of the LwIP TCP/IP Stack. http://www.sics.se/ ~adam/ lwip/doc/lwip.pdf,2001-02.

[39] 邱书波,陈伟.基于ARM的轻量级TCP/IP协议栈的研究及移植[J]. 计算机应用与软件,2009,(8):90-92.

[40] 杨辉.基于 μC /OS-II 的 LwIP 网络协议移植[J]. 中国科技信息,2010,(11):99-100.

[41] Adam Dunkels. The μC /IP Project. http://ucip.sourceforge.net/

[42] Adam Dunkels. The μIP Embedded TCP/IP Stack. http://www.sics.se/~adam/download/?f=uip-1.0-refman.pdf,2006-06.

[43] Micriμm. μC/ TCP-IP User's Manual. V1.91, 2007-10-24.

[44] 范道威,曲波,杨晔.基于 ARM9 和μC/OS-Ⅱ的 μC/ TCP-IP 协议栈移植[J].苏州大学学报(工科版),2010,(4):49-53.

[45] 孙静.基于以太网控制器 LAN91C111 的 μC/ TCP-IP 网络接口通信实现[J].电脑学习,2010,(6):52-54.

[46] ARM Limited. ARM Architecture Reference Manual. 2005.

[47] 周立功.ARM 与嵌入式基础教程[M].广州:广州周立功单片机有限公司,2004.1-70.

[48] Seqars Simon. ARM7TDMI Power Consumption[J]. IEEE Micro,1997, (17)12-19.

[49] 曹智军.一种 IC 卡自助加油机的设计[J].SCIENCE & TECHNOLOGY INFORMATION, 2011, (23):99-100.

[50] 张绮文,谢建雄,谢劲心.ARM 嵌入式常用模块与综合系统设计实例精讲[M].北京:电子工业出版社,2007. 2-5.

[51] Wang Yingchun, Ji Lijiu, Han Shu, etal. A high Speed Microprocessor Core for the Embedded System-PKURS001. 4th International Conference on ASIC Proceedings, Shanghai, 2001:748-751.

[52] NXP Semiconductors.LPC23XX User manual.Rev.02, 2009-02-11

[53] RAMTRON.FM28V020 Preliminary data sheet. Rev.1.1, 2009-09.

[54] ATMEL.AT45DB041D User Mannul. 2009-09.

[55] 周立功.SP708 技术手册. http://www.zlgmcu.com.

[56] National SemiConductor.DP83848C User's Manual. 2005-09.

[57] KEIL.ARM Development Tools. μVision Help.

[58] 谭浩强.C 语言程序设计（第三版）[M].北京:清华大学出版社,2005.69-130.

[59] Jones, M.T..嵌入式系统 TCP/IP 应用层协议[M].路晓村等译.北京:电子工业出版社,2003. 258-295.

[60] 王金华,李允俊.嵌入式数据库系统的研究[J]. 信息技术,2012,(3):73-74.

[61] Albert S.Woodhull.操作系统设计与实现(第三版)(上册)[M]. 陈渝,谌卫军译.北京:电子工业出版社,2007. 20-70.

[62] LIU Yuling, Wang Huiran, TIAN Junfeng.TCP/IP Feature Reduction in Intrusion Detection[J].Wuhan University Journal of Natural Sciences,2007,1(12):151-154.

# 附录 A：加油机控制装置实物图