学校代码：10255
学　　号：2070961

多 MCU 基于数据库实现方法的研究

专　　业：检测技术与自动化装置
姓　　名：连翔宇
指导教师：唐莉萍
答辩日期：2010 年 3 月 2 日

东华大学　信息科学与技术学院
College of Information Science and Technology
Donghua University

东华大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律结果由本人承担。

学位论文作者签名：连翔宇

日期：2010年3月

东华大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权东华大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密 □，在 ____ 年解密后适用本版权书。

本学位论文属于

不保密 ☑。

学位论文作者签名：连翔宇　　指导教师签名：

日期：2010年3月3日　　日期：2010年3月3日

多 MCU 基于数据库实现方法的研究

摘 要

随着单片机技术的发展，单片机控制系统在各种场合应用越来越广泛。单片机控制系统的一般设计方法是根据用户提出的固定要求对系统外围的硬件设备和系统软件进行设计。一旦系统功能确定以后，如果用户需要对输入、输出信号、数据结构、显示页面和按钮功能等进行修改，必须重新编写源程序，系统软件的复用功能差，重新开发需要的周期长，因此系统的可维护性和可升级性能较差。

目前大型的控制系统大都采用通用工业自动化组态控制软件开发应用程序。用组态软件开发的应用程序，当现场（包括硬件设备或系统结构）或用户需求发生改变时，不需作很多修改而方便地完成软件的更新和升级。市场上流行的组态软件有国外 Wonderware 公司的 Intouch 、国内昆仑通泰公司的 MCGS 和北京亚控公司的组态王（Kingview）等。这些组态软件本身价格昂贵，需要操作系统支持。上位机使用 PC 机或高档的嵌入式系统为组态软件提供人机界面支持。因为它的成本较高，一般的单片机控制系统很少选用。

本文结合组态软件的优点和传统的单片机系统控制软件的不足，提出一种低成本动态组态单片机控制系统的解决方案。将一些常用的函数作为系统的应用程序保存在单片机的固定程序中，而页面信息、输入输出端口信息、系统的功能、流程等都通过配置文件进行组态。

文章从分析传统的单片机控制系统特点入手，提出小型动态组态

控制系统的系统架构。根据系统架构特点，分别从通用的硬件结构和软件模块两方面进行讨论。硬件模块由主控模块、人机交互界面、外部存储和 I/O 端口模块四部分组成。结合系统的硬件构成，设计单片机核心控制、屏幕显示及键盘处理、大容量外部存储模块读写、I/O 端口控制模块的通用程序。

系统的核心部分是对存放在大容量外部存储设备上的组态系统控制配置文件的操作。包括对配置文件的存放、读取、解析，并根据配置文件对系统的人机交互界面及 I/O 端口进行控制输出。系统采用标准的 XML 文档作为动态组态控制系统的配置文件。由于单片机的内存空间小，处理速度慢，无法应用现有的 XML 解析程序。本文在分析 XML 文档结构特点的基础上，设计了一种针对单片机平台使用的 XML 解析程序。通过系统自带的解析程序能方便地对 XML 程序进行查找、读取、解析及改写。

最后用本文提出的动态组态系统实现了一个实际的多路洗衣机洗液分配控制系统。

关键词：大容量存储，动态组态，控制系统、XML 解析器

A RESEARCH OF MULTI-MCU IMPLEMENTATION BASED ON DATA SET

ABSTRACT

With the development of micro controller unit (MCU) technology, MCU control systems have found more and more applications. Conventionally, MCU control system is designed upon the input/output and functional requests. Therefore, system peripheral hardware devices and the controlling software is determined. In case of version-up or functional modifications, the controlling program must be modified or re-written to cope with the change of signal arrangement, data flow, data structure, user interface modification, or button functions. System software reusability and productivity are poor.

Configuration controlled software is widely used in large-scale control systems. These configuration controlled software systems can be easily configured. Without any code-modification, system functions are even realized on-site. There are some third-part software productions, such as, Wonderware's Intouch, KunlunTongtai's MCGS, and WellinTech's Kingview. These configuration controlled software products are expensive, and need operating system such as Windows. The host is a PC or high grade embedded system. The GUI is developed to support on-screen configuration. Because of the cost and software volume, these

software can not fit for MCU control systems.

Combining the advantage of configuration controlled software to the MCU system hardware resources, a low cost dynamic configuration controlled system solution is proposed in the thesis. General functions are stored as part of system's application program. The menu control information, input and output port specifications, system functions, and processing routines are configured by the configuration files.

In the thesis, the characteristics of conventional MCU control system are discussed. A small-scale dynamic configuration control system architecture is proposed. According to the system architecture features, hardware architecture and software modules are discussed in detail. The hardware module is composed of four modules including, the main control module, human-computer interface, external storage and I / O port module. The common procedures of the core microcomputer control, screen display and keyboard handling, high-capacity external storage module to read and write and I / O port control modules are implemented.

The operation description files for system operation are stored in the large-capacity external storage device. The system can store, read and parse the configuration files. The human-computer interface and I / O port outputs are defined by these files. The standard XML formats are used for the dynamic configuration control system configuration files. XML parsers used in the MCU system are specially designed to meet the

requirement of MCU conditions. With limited resources, the proposed XML parser can read, write, get the command contents from XML files. Finally, the proposed dynamic configuration system is successfully used in the multi-machine distribution control system for laundry.

LIAN Xiang-yu (Examine Technology and Automatization Equipment)

Supervised by Tang Liping

KEY WORDS: large-capacity storage, dynamic configuration, control systems, XML parser

目 录

插图索引

表格索引

1 绪论

随着科学技术的高速发展，计算机的应用已经越来越广泛。单片机技术和应用也得到了迅速发展。在工业生产领域，如自控、机电、化工，民用家电等各方面，单片机均有着广泛的应用[1]-[4]。

在工业生产中，单片机的应用最为广泛。主要表现为以单片机为核心所组成控制系统，对系统进行数据采集、命令控制、输入端口信号检测和输出端口设备控制等多种操作。为了满足工业生产中各种不同应用场合，能否针对各种不同工作目标、可以方便升级的控制系统成为衡量单片机控制系统的关键因素。

1.1 选题背景、目的及意义

传统的单片机控制系统的源程序都是根据用户提出的要求和系统外围包含的硬件设备进行设计。按照一定的功能，事先定义好所有的输入、输出信号的处理方法、显示页面、按钮功能，根据系统流程信息编写系统程序。程序功能和使用环境相对固定，设备依赖性强[5]。这种方法的优点是程序执行速度快，缺点是编写程序工作量大、周期长。如果用户对某些功能或信号端口需要进行修改，则必须重新连接硬件、编写源程序。因此系统的可维护性和可升级性能差。

本课题研究的目的为了实现一种动态组态的单片机控制系统。系统只对底层的应用程序进行源程序设计和编程，外围的具体功能，在系统软件完成之后根据用户的需要，用系统的配置文件定义，通过动态组合实现。这样当系统的功能需要进行修订时，不需修改原有的软件程序和硬件系统，只需简单地修改配置文件即可实现系统功能的修改。

本文提出一种基于数据库的多 MCU（Micro Controller Unit）控制系统，以数据库文件定义来实现多 MCU 系统的结构和实际功能。在实际使用中按照用户的需要对数据库进行动态设置，定义各种硬件功能、按钮功能及页面显示功能，将定义好的硬件系统和底层应用软件进行功能组合，即可实现系统功能的重组和修改。从而免去系统软件代码的重新开发。

1.2 国内外研究现状及发展趋势

传统的单片机控制系统[6]-[8]功能具有一定的针对性。如果需要新建、扩展或变更功能时，可能要对系统的软硬件进行重新设计，这样就需要很大的开发成本和很长的开发周期。解决这类问题的方法是对系统的功能进行组态。

组态"Configuration"就是用应用软件中提供的工具、方法，完成工程中某一具体任务的过程。组态的概念最早出现在工业计算机控制中，如 DCS[9]（Distributed Control System）组态，PLC[9]可编程序控制器组态。

DCS，即所谓的分布式控制系统，或称为集散系统，是相对于集中式控制系统而言的一种新型计算机控制系统，它是在集中式控制系统的基础上发展、演变而来的。在系统功能方面，DCS 和集中式控制系统的区别不大，但在系统功能的实现方法上却完全不同。DCS 是由系统网络以及可以完全对现场 I/O 进行处理并直接实现控制功能的网络节点组成的。DCS 组态的基础和核心是系统网络，它是对 DCS 进行离线配置、组态工作及在线系统监督、控制、维护的网络节点。其主要功能是提供对 DCS 进行组态、配置工作的工具软件（即组态软件）。系统网络可以在 DCS 在线运行时实时监视 DCS 网络上各个节点的运行情况，使系统操作人员可以通过系统网络及时调整系统配置、设定系统参数，使 DCS 保持在最佳的工作状态之下。与集中式控制系统不同，所有的 DCS 都有系统组态功能，可以说，没有系统组态功能的系统就不能称其为 DCS。

PLC 可编程序控制器是微机技术与传统继电器控制技术相结合的产物。它克服了继电器控制系统中机械触点的接线复杂、可靠性低、功耗高、通用性和灵活性差的缺点，充分利用了微处理器的优点。PLC 的程序编写，不需要专门的计算机编程语言知识，而是采用一套以继电器梯形图为基础的简单指令形式，使用户程序编制形象、直观、方便易学。PLC 的调试与查错也都很方便，用户在购到所需的 PLC 后，只需按说明书的提示，做少量的接线和简易的用户程序编制工作，就可灵活方便地将 PLC 应用于生产实践。

组态软件指以组态形式组成的面向监控与数据采集（Supervisor Control And Data Acquisition，SCADA）的软件平台工具[10]。组态软件以其丰富的项目设置，

灵活的使用方式，强大的功能，稳定可靠的工作性能，良好的人机界面，方便的硬件配置以及简单的编程等特点而获得广泛的应用[11]。

现在市场上主要流行的组态软件有 Wonderware 公司的 Intouch[12]、昆仑通泰公司的 MCGS[13]和北京亚控公司的组态王（Kingview）[14]等。

美国 Wonderware 公司的 Intouch，堪称组态软件的"鼻祖"，率先推出的 16 位 Windows 环境下的组态软件，在国际上曾得到较高的市场占有率。Intouch 软件的图形功能比较丰富，使用较方便，但控制功能较弱。其 I/O 硬件驱动丰富，但是使用 DDE 连接方式，实时性较差，且驱动程序须单独购买。

昆仑通泰公司的 MCGS（Monitor and Control Generated System）是一套基于 Windows 平台的、用于快速构造和生成上位机监控系统的组态软件系统。能够完成现场数据采集、实时和历史数据处理、报警和安全机制、流程控制、动画显示、曲线和报表输出、企业监控网络以及高性能、高可靠性、低成本的嵌入系统等功能。使用 MCGS，用户无须了解计算机编程的知识，只要通过其提供的上位机工具软件进行简单开发，就可以轻易完成一个稳定，成熟，具备专业水准的计算机监控系统。

北京亚控公司的组态王，是国内较早出现的组态软件产品之一。早期的组态王是对 Intouch 的仿造，只有单机接口，程序构架不合理、网络功能较为薄弱，支持不了真正意义上的分布式系统。现在已经具有了较强大的组态功能。

上述的组态软件本身价格昂贵，且需要操作系统（例如 Microsoft Windows 95/ 98/ ME/ NT/ 2000[15] 或 WinCE ）的支持，上位机使用 PC 机或高档的嵌入式系统为组态软件提供人机界面支持，这些组态软件适用于大型的控制系统的组态。对于小型的单片机控制系统，目前组态的研究还是在初步阶段，为用户提供低成本、无需操作系统支持的小型动态组态单片机控制系统，是单片机控制系统的一种趋势，即实现组态系统的小型化[16]。

1.3 主要研究内容

首先从传统的单片机控制系统特点入手，研究小型动态组态控制系统的系统

架构。根据系统架构特点，分别确定硬件结构和软件模块。由主控模块、人机交互界面、外部存储和 I/O 端口模块四部分组成系统的硬件框架。根据系统的硬件构成，给出单片机核心控制、屏幕显示及键盘处理、大容量外部存储模块读写、I/O 端口控制模块的通用程序的设计思想和实现方法。

系统的配置数据库方案采用标准的 XML 文件格式。针对单片机的内存空间小，处理速度慢，无法应用现有的 XML 解析程序的特点。在分析 XML 文档结构特点的基础上，设计一种针对单片机平台使用的 XML 解析程序。通过系统自带的解析程序能方便地对 XML 程序进行查找、读取、解析及改写。

完成用动态组态的单片机控制系统实现多洗衣机洗液分配器的控制功能。

1.4 文章结构安排

1、分析当前单片机控制系统及动态组态思想的特点，提出小型单片机动态组态系统的系统构架。并针对该系统构架，给出合理的硬件结构及软件设计思路和方法。

2、根据提出的单片机控制系统的硬件结构，进行功能模块的细化设计。对每个功能模块的主要器件根据系统要求，进行选型，比较可选硬件的特点，确定合适的元件，构建硬件系统。

3、根据单片机控制系统的硬件结构，在满足系统功能动态组态为目的的前提下，设计及编写控制系统的底层应用软件。

4、确定系统配置的数据文件库文件格式。设计系统的 XML 解析模块，实现对系统配置文件的各种操作，从而达到系统动态组态的目标。

5、应用本文提出的动态组态单片机系统实现一个洗衣机洗液分配器控制系统。给出控制页面组态的 XML 数据库文件的格式，以及数据库动态组态页面实现的方法。针对系统的硬件设备特点进行优化设计。

2 系统设计的思想

2.1 概述

传统的单片机控制系统一般由以下两大模块组成，如图 2-1 所示：

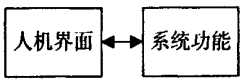


图 2-1 传统控制系统结构图

1、操作人员能够查看系统当前操作信息、运行状态及设定系统控制功能的人机界面模块。

2、实现、响应操作人员输入的控制功能的系统功能模块。

这种单片机控制系统结构的特点是：系统的功能在硬件设计、软件程序编写完成之后就已经确定下来，若系统功能需要更改或修订，则硬件设计需要变更，程序需要重新编写，系统的灵活性和可维护性低下。

2.2 系统三层架构的提出及设计

针对传统单片机系统功能不易修改以及现有大型组态控制本高的不足，本文提出一种针对小型控制系统的动态组态单片机控制系统结构。在设计系统时，对系统的硬件结构必须充分考虑，以满足用户对系统硬件的扩展需求。系统的软件部分，在设计的初期只需要考虑最基本的功能，即对不同的硬件模块的底层应用程序进行编程。用户提出需要实现的具体控制目标功能需求，可以通过人工配置数据库信息，和系统预定义的基本应用程序进行功能组态，实现特定的控制系统目的。在系统使用过程中，如果用户的需求发生更改，系统无需修改硬件连接、软件程序部分，只要对系统功能的变更调整数据库信息即可，这样就可以大大节约系统变更时重新开发的时间，极大提高系统的灵活性。

根据动态组态的思想，单片机控制系统采用三层系统结构：即在传统的单片机控制系统中，添加一层数据库信息模块。如图 2-2 所示：

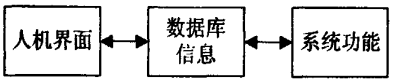


图 2-2 动态组态控制系统构架

为了能够实现对软硬件进行动态组态，必须以一种耦合程度低的方式，设计系统的软硬件，这样才能方便实现系统根据数据库信息进行动态组态系统的功能。

模块化的设计符合系统耦合度低的要求。所谓模块化，就是从系统的观点出发，考虑系统功能的具体类型，将系统功能以最小功能结构的思想进行分解，建立起模块的体系，最后运用模块组合的方式，建立系统的过程。模块化不是研究和解决某一个孤立系统的设计或构成问题，而是解决某类系统的最佳构成形式问题。它的主要方法是系统的分解和组合。系统的分解和组合的技巧和运用水平，是模块化的核心问题。模块化过程的特点有以下几点[17]：

（1）具有目的性。没有明确的目标，模块化就失去了方向，就无从组织和管理模块化全过程。

（2）综合性。模块化的目标是建立一个模块系统和模块化产品系统，这是一个综合的、复杂的系统工程，需要多个学科人员通力合作才能很好的完成。

（3）动态性。模块化过程是一个动态过程，一个模块化的产品系统不仅有其形成、完善和成熟的过程，而且还有其老化和更新的过程，所以必须及时淘汰老化的模块，不断补充新的模块，才能使这个模块化系统保持生机，取得尽可能大的效益。

（4）超前性。模块化过程应及时预测和追踪技术发展动向，及时采用新技术，以确保模块化系统技术上的先进性。

从上面几点可以看出，相对于传统设计方式，采用模块化设计，因其低耦合性使每个单独模块具有各自具体的目的性，通过对模块化设备的动态组态，即能实现系统的功能。

2.3 系统硬件平台的总体设计

根据提出的三层系统构架思路，设计的系统硬件结构如图 2-3 所示。

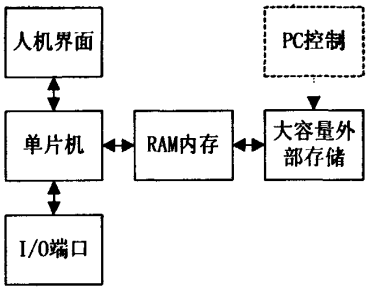


图 2-3 系统硬件设计框图

（1）单片机：作为系统的核心部分，与其它各硬件模块相连，负责协调其它模块的工作。

（2）人机界面：提供一个系统和用户之间交互的界面。通过人机界面，用户可以查看系统运行信息，实时数据，并可以对系统进行各种操作和控制命令设定。

（3）RAM 内存：该模块作为一个中间媒介，负责临时保存单片机模块中需要输出的或者需要调用的临时数据。

（4）大容量外部存储设备：用于存储用户根据需求预先设计的数据库信息。由于单片机自带 RAM 内存较小，系统运行时所有的数据库信息无法都保存在单片机 RAM 内存中，故需外部大容量存储设备支持。使用大容量外部存储设备时，用户所需功能的数据库信息可以通过 PC 机对外部存储设备进行输入和修改，为数据库的修改提供便利。

（5）I/O 端口：系统的输入控制信号和输出驱动信号。为了扩展输入、输出端口引脚数，对单片机的部分 I/O 端口设计采用端口复用技术。所有的输入、输出端口都采用光电隔离技术，以提高系统的抗干扰能力。

其中，单片机构成系统的控制模块、RAM 内存及大容量外部存储设备共同构成系统构架中数据库信息模块。单片机控制模块实现数据库信息的读取、解析及更改，是组态系统的核心部分。I/O 端口部分是具体功能输入输出的实现部分。

2.4 系统软件平台的总体设计

系统软件的主要功能是检测、处理按键事件的响应，根据用户需求查找对应的数据库，检查系统输入端口状态，按组态信息进行各种功能及输出端口的控制。

系统结构功能图如图 2-4 所示：

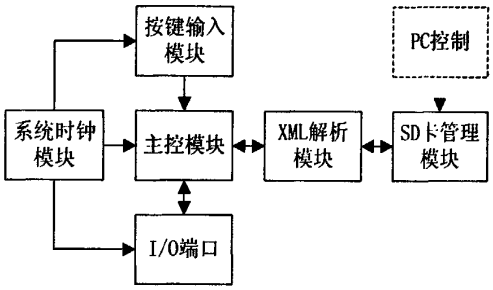


图 2-4 系统软件设计框图

（1）主控模块：对应于硬件的单片机模块的实际功能，管理系统的命令收发，信息处理，数据处理、键盘扫描输入、串行通讯、接口扩展等。

（2）人机界面模块：作为人机交互界面的具体实现，担当着系统信息的显示和键盘的设定。显示信息包括：系统当前工作状态、内部参数、外部 I/O 端口状态等。键盘实现系统工作参数设定、显示页面切换等功能。

（3）XML 解析模块：用于对 SD 卡中的 XML 文档进行解析，实现对 XML 文档的读写、修改、添加、删除等操作。既能从 SD 卡中读出系统的配置文件共主控模块使用，又能将主控模块对配置文件修改的控制信息写入 XML 文档。

（4）大容量外部存储设备管理模块：为了方便管理存储、管理用户根据需求预先设计的数据库信息，在大容量外部存储设备上搭建文件系统格式，便于大量文件信息的检索查找。

（5）I/O 端口管理模块：I/O 端口模块需要识别主控模块发来的用于识别系统的请求查询状态的命令，对端口进行查询后，以一种可以识别的信号方式返回给主控模块。同样，对于主控模块发来的需要输出的信号的命令要求，I/O 端口管理模块识别命令要求，查找到对应的系统端口进行输出。

3 系统的硬件设计

3.1 硬件系统设计的一般原则

硬件设计是以满足系统所需要的功能为前提，在设计的过程中可以遵守以下几条原则[18]：

（1）合理采用新技术，在硬件设计中采用新型而且功能完善的器件。新型而且功能完善的器件往往功能更强大，具有更高的精度和可靠性。选用新型而且功能完善的器件可以简化系统电路设计，并能够在满足系统功能和可靠性的前提下为以后系统升级留有余地。

（2）采用开放式的硬件架构，在设计中充分考虑系统未来的升级要求。硬件的升级相对于软件的升级而言要复杂得多，因此，在硬件设计之初应该尽可能留有余地，并尽量使电路做到通用，以便将来的修改和扩充。

（3）充分利用现有的成熟的技术，尽可能选择典型电路，为硬件系统的标准化、模块化打下良好的基础。采用成熟的技术可以保证电路的正确性和可靠性，并且可以缩短设计时间。

（4）系统扩展与外围设备的配置水平应充分满足应用系统的功能要求，并留有适当余地，以便进行二次开发。

（5）硬件结构应结合应用软件方案一并考虑。硬件结构与软件方案会产生相互影响，考虑的原则是：软件能实现的功能尽可能由软件实现，以简化硬件结构。

（6）可靠性及抗干扰设计是硬件设计必不可少的一部分，包括器件选择、去耦滤波、印刷电路板布线、通道隔离等。

（7）在设计中考虑系统的成本，使性价比达到最优。

同时，硬件产品的选择需要综合考虑开发成本和开发难度两个方面。在成本方面，应该尽可能的充分利用芯片的资源，以最小的经济代价换取最大程度的功能实现；在开发难度方面，尽可能选择有成熟的开发环境和方便的开发、调试方式的硬件器件。

3.2 系统硬件各模块设计

硬件设计中采用模块化的设计方法，不仅可以使设计简单，便于电路调试和故障检测，而且可以方便后续新硬件功能模块的添加和原有硬件功能模块的删除，增强系统的灵活性。

3.2.1 主控模块确定

控制系统是基于数据库的多 MCU 控制系统，可分成主控部分及外部功能部分的两类单片机模块。下面首先讨论主控单片机模块选型。

作为动态组态系统单片机控制系统的核心模块，主控部分的单片机模块需要具有以下特点：

1、较高的系统工作频率，以能够快速处理各种命令。

2、系统工作时需要同时处理人机界面显示信息和大容量存储模块读取写入处理，这都需要大内存的支持。同时系统还需要处理包括程序中所使用的各种变量，因此实际所需要的内存大于 2K，需要较大的系统内存容量以保证系统的正常运行。

3、较多的系统 I/O 端口，能够方便系统扩展控制各种硬件模块。

4、拥有较为丰富的 A/D，D/A 转换器，以适应各种不同的处理环境。

5、能够方便与外部大容量外设进行访问控制的特殊功能。

6、有能够与外部设备进行信息交换的通信端口。

此外还需要考虑核心模块的性价比和易开发性等因素。所以在单片机选择上，需要综合以上几点进行考虑。

目前，最为流行的单片机系统结构有单片机系统及嵌入式系统。嵌入式系统是 32 位或者更高位的微机处理系统，其核心一般为 ARM、DSP 或 FPGA 等，功能强大，一般情况下需要其他的操作系统的支持，可以用于大中型动态组态控制系统中。对于小型动态组态控制系统，其所有资源无法得到充分利用，形成资源浪费。

与嵌入式系统相比，虽然单片机的处理速度低，内存容量小，但是它和外围

电路连接简单，不需要操作系统的支持，成本也比嵌入式系统低的多，适合于小型动态组态控制系统。故 MCU 的选取在单片机芯片中进行考虑。

现在单片机系统大致分为传统意义上的单片机及新型的 SoC（System on Chip）片上系统单片机。

传统意义上的单片机 80C51 具有运算速度慢、功耗大、内部资源少等不足，所以限制了其使用范围。

型片上系统指的是在单个芯片上集成的一个完整系统，将中央处理器（CPU）、存储器、以及外围电路等集中在一块芯片上，在使用中只需通过控制程序即可以实现系统集成硬件的调用和控制。将外部各部件集成在一起，这样可以避免设计一个各部件独立、需要占用大量单片机端口进行通信、信息访问复杂的系统，大大方便了使用，系统也非常灵活、高效[19]。

Silicon Labs C8051F 系列中的 C8051020[20][21]单片机是 SoC 最典型的代表。它具有以下主要特性；

（1）高速流水线结构设计，与 CIP-51 控制器内核兼容，与标准的 8051 结构相比，指令执行速度有很大的提高。标准的 8051 单片机执行一个单周期指令需要 12 个系统时钟周期，而 C8051F MCU 执行一个单周期指令只需要一个系统时钟周期。C8051F 峰值速度可达 25MIPS，程序运行方便，运行速度快。

（2）含有 4352（4096+256）字节的片内 RAM，可以保证系统能够同时处理大量数据信息时的内存需要。

（3）拥有 64 个数字 I/O 引脚，可以利用其进行方便的系统扩展。

（4）8 位 500ksps 的 ADC，带 PGA 和 8 通道模拟多路开关。两个 12 位 DAC，具有可编程数据更新方式。自带多个 A/D、D/A 转换器，能够使系统处理各种外来的模拟信号或数字信号，而无需占用其他单片机通用接口，即可进行 A/D、D/A 芯片的扩展连接。

（5）更重要的一点是一般单片机本身存储都不是特别大，当需要扩展外部存储设备时，需要专门的外设接口。C8051F020 包含有 SMBus（兼容 I^2C）及 SPI 接口两种常用的接口，通过各自协议，可以与多种外部存储进行方便通信。

（6）C8051F020 包含有两个 UART 串口。可以通过 UART 串口组建控制网络，和其它控制系统单元模块进行通信，实现多 MCU 控制的功能。

同时，C8051F020 的 FLASH 存储器还具有在系统重新编程能力，可用于非易失性数据存储，并允许现场更新 8051 固件。使用 JTAG 调试时，系统支持观察和修改存储器和寄存器，支持断点、观察点、单步及运行和停机命令，所有的模拟和数字外设都可全功能运行。C8051F 最为独特的是增加了“Digital Crossbar”数字交叉开关。它可将内部数字系统资源定向到 P0、P1 和 P2 端口 I/O 引脚，也可通过设置 Crossbar 开关控制寄存器将定时器、串行总线、外部中断源、AD 输入转换、比较器输出等定向到 P0、P1、P2 的 I/O 口。这就允许用户根据自己的特定应用选择通用 I/O 端口和所需数字资源的组合，使用起来非常方便。

综上所述，采用 C8051F020 单片机作为系统主控模块。

3.2.1.1 系统供电电路的设计

C8051F020 的供电电压为 2.7V - 3.6V，采用低压供电。相对于其他需要 5V 供电的单片机，在系统运行时，功耗更小，更节能。

系统常规供电为 5V，采用 LM1117[22]低压差线性调压器变压芯片对系统进行调压。经过调压芯片调压可以将 5V 输入电压，转换成 3.3V 进行输出。

3.2.1.2 单片机晶振频率的选择

C8051F020 的系统时钟由晶振电路的振荡频率确定。选择晶振频率时，要从系统的稳定性和处理速度两个方面综合考虑。通常较低频率的晶振能使系统工作更为稳定，但系统的处理速度较慢；较高频率的晶振能加快系统的处理速度，但系统的稳定性相对可能差一些。

C8051F020 的系统时钟可以取自内部振荡电路、外部振荡电路。内部时钟振荡器可以为其提供振荡频率，但是内部时钟振荡器的精度不是很高（最大误差达±20%）。由于不同器件内部振荡器的离散性较大，所以不能用于产生波特率，应该外接标准晶体，为系统提供稳定工作频率，并保证产生稳定的波特率。

在晶振选择时还需要考虑单片机所能支持的晶振范围。C8051F020 支持的晶振工作范围最大为 30MHZ。在此晶振工作范围内，单片机系统均可以正常工作。

本文设计的系统中，工作瓶颈部件为单片机对大容量外部存取设备读写操作，针对这个特点，需要选用较高频率的晶振。提高系统工作频率，可以保证系统整体工作有较高的效率。选择晶振除了要保证系统的高效工作，还必须考虑保证系统传输信息时有较低的误比特率。

若系统中采用晶振频率为 22.1184MHz，C8051F020 单片机执行一个单周期指令只需一个系统时钟周期，即为 45ns。在此频率下，改写 5070Byte 的 XML 页面配置文档大约耗时 2500ms。因本系统考虑为小型非高速控制系统，这个时间在可以接受的程序范围之内。若采用更低晶振频率，改写程序的耗时则会增加。

在尽量保证为系统提供较高工作频率的前提下，选择使用22.1184MHz晶振，还能为串口通信提供从 600 到 115200 的波特率，保证通讯无误码。因此决定选取 22.1184MHZ 晶振为单片机提供系统时钟信号。

3.2.1.3 UART 串口和 SPI 控制端口交叉开关的配置

Crossbar（交叉开关）是 C8051F 系列单片机的一大特点，它实际上是将内部功能单元（例如 UART、SPI 等）与外部 I/O 管脚相连的一个可配置开关矩阵。用户可以按照一定的优先级顺序将用到的内部功能单元配置到 I/O 管脚上，剩下没有被内部功能单元占用的 I/O 管脚作为通用 I/O 口使用。

控制系统中需要使用 UART 串口与 I/O 端口模块进行信号传输，需要 SPI 控制端口对 SD 卡读写操作。优先权交叉开关译码器按优先权顺序从 P0.0 开始，可以一直分配到 P3.7。它为数字外设所分配的端口引脚的优先顺序是按系统默认的顺序，即：串行通信 UART0 具有最高优先级，TX0 和 RX0 分别被分配到 P0.0 和 P0.1；串行通信 SPI 具有次高优先级，被分配到 P0.2；SCK、MISO、MOSI 和 BSS 分别被分配到 P0.3 P0.4 和 P0.5。详细的端口引脚的优先分配顺序表请参考参看文献[23]。

3.2.1.4 系统通信方式的选择

单片机的通信方式分为串行和并行两种方式。其中并行接口是指数据的各位同时进行传送，其特点是传输速度快，但当传输距离较远、位数又多时，通信线路复杂且成本提高。串行接口是指数据逐位顺序传送，其特点是通信线路简单，

一般只要一对传输线就可以实现双向通信，成本较低，特别适用于远距离通信，但传送速度相对较慢。因本系统为小型控制系统，传输的数据信息量不是很大，所以对速率要求不高。但是被控对象可以离主控模块较远，且需要多机相互通信，因此要求保证一定的传输距离，所以系统采用串行口通信方式进行信息、命令传送。

在选择具体使用的串口通信方式，也需要考虑传输距离和多机通信的问题。现在广泛使用的串行通信方式有 RS-232[24]、RS-422[25]及 RS-485[26]，表 3-1 是三种通信方式的比较[27]：

表 3-1 三种串口通信方式对比

项目	RS-232	RS-422	RS-485
操作方式	单端	差分	差分
单线上的驱动和接收器的总数（RS-485 网络一次一个驱动器有效）	1 个驱动器 1 个接收器	1 个驱动器 10 个接收器	32 个驱动器 32 个接收器
电缆最长长度	50 英尺（2500pF）	4000 英尺	4000 英尺
最大传输速率（对 RS-422/RS-485 40 英尺-4000 英尺）	20Kb/s（特定情况下更高）	10Mbits/s	10Mbits/s
工作方式	全双工	全双工	半双工
连接方式	点到点	点到多点	多点到多点

考虑本文设计的控制系统，可能需要扩展多个 I/O 端口模块，同时控制多台从设备，且无法预知用户对设备的分布情况。为了保证可靠的数据通信和传输距离，采用 RS-485 通信协议进行通信。RS-485 通信只需要两条传输线即可实现数据的高速传输，线路架设方便。

设计的电路中，采用了 Maxim 公司的 MAX485 芯片[28]作为 RS-485 串口工作芯片，其与单片机接口电路如图 3-1 所示：

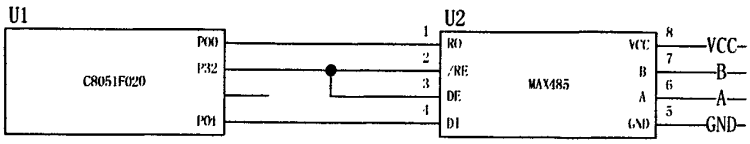


图 3-1 MAX485 芯片连接图

A、B 两端口连接到系统的传输线路上，进行差分信号的传输。DE 为 MAX485 芯片的接收、发送使能端，用于控制半双工模式下的信号发送或是接收。RO 和

DI 分别接单片机的 TX0（P0.0）和 RX0（P0.1）端口。

当 DE、/RE 引脚置低，单片机处于接收状态。当 A-B 的差值>+0.2V 时，RO 为高电平；当 A-B 的差值<-0.2V 时，RO 为低电平。当 DE、/RE 引脚置高时，A、B 引脚的差值将不影响 RO 上的电压值，其呈现为高阻态。如图 3-2 所示。

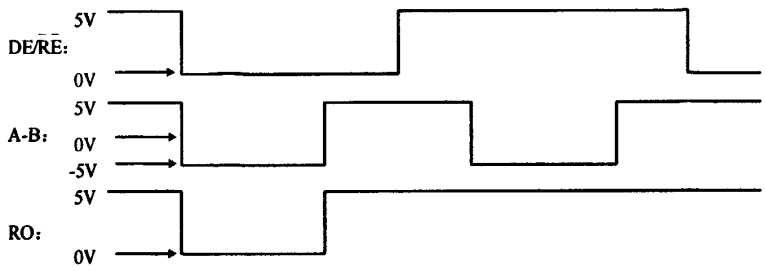


图 3-2 单片机接收时序图

当 DE、/RE 引脚置高，单片机处于发送状态。此时 A 与 DI 有同向电压，B 为 DI 的反向电压。当 DE、/RE 引脚置低时，A、B 引脚上的电压呈现为高阻态。如图 3-3 所示。

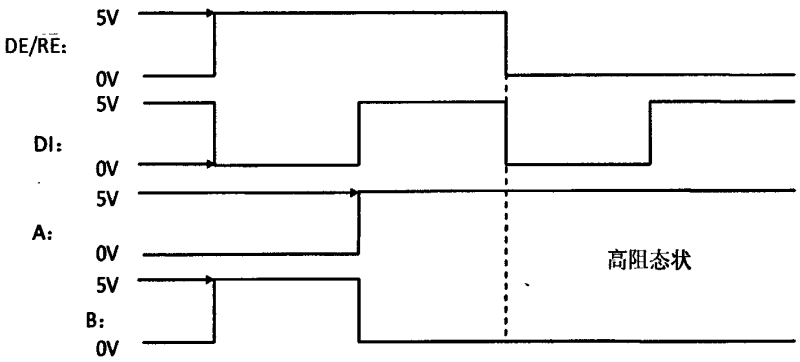


图 3-3 单片机发送时序图

3.2.2 人机界面模块

人机界面分为将用户指令通过输入接口输入到系统的输入部分及系统将当前运行内容及可控信息输出给用户查看的输出部分。

在系统中，显示状态信息内容宽度一般不超过 15 个汉字即可以表达单条的控制信息，且针对每个不同的显示页面，信息行数可能不确定，范围在 3～10 行之间不等。输入按钮针对不同页面，也会有数量上的区别，范围在 5～20 个不

等。

常用的输出显示部分大致可分为 LED 及 LCD 两种显示方式。LED 显示屏是以发光二极管组成的显示阵列。采用 LED 显示屏对文字信息进行显示，每显示一个汉字需要 8×8 或 16×16 个 LED 组成，若需要每行显示 15 个汉字信息，相对于小型单片机控制系统，则用于显示的 LED 阵列体积庞大。且一旦预先设定好 LED 阵列大小后，如若需要同时显示更多的文字，就需要重新修改、组建 LED 阵列，不符合系统设计中可在不改变硬件系统的连接、配置的情况下，动态组态系统功能的要求。若在系统初始时，设计大像素 LED 阵列，但实际使用范围不确定，若系统显示内容较少，硬件资源浪费。

LCD 即为液晶显示屏，相对于 LED 显示阵列，LCD 的显示像素小，在相同的单位面积里可以比 LED 显示更多的内容，可以大大减小显示输出时的显示屏的体积。且其还具有平板型结构、被动显示型、无电磁辐射、长寿命、耗能低等诸多优点。因此考虑选用 LCD 模块作为系统的人机界面显示实现模块。

传统的人机界面模块的输入部分由按钮担当，实现人机界面的参数设定和功能切换功能。这种硬件选择存在问题：系统可进行输入的按键功能的个数受按键物理个数的限制，若需要对增添人机界面中更多的输入按键，需要从硬件上添加外部按钮，并与系统主控单片机端口相连。这样，若所需输入按键不确定，需要按照最多需要量添加按钮，此时若发生系统输入功能更改，无论是实际需要按钮较少造成的浪费或需要添加按钮造成的扩展麻烦，对小型动态控制系统来说，无法实现设计系统动态组态的目的。

LCD 显示屏的触摸式按钮可以任意划分组合。触摸式液晶屏的输入部分是整块液晶屏幕范围，用户可以根据需要将屏幕分成不同的按钮功能块，还可以针对不同页面，定义不同的按键和按键功能。当需要增添新的按键时，无需更改硬件，只需添加按键的软件程序，扩展、修改方便，符合动态组态系统的思想，故选用可触摸式液晶显示屏作为系统的人机界面模块。当然，在按键定义时，还需要考虑按键的舒适性和方便性，不能在有限的触摸屏范围内定义过多的按钮，以防止产生按键输入的误操作。

系统的显示信息每次均不超过 15 个单位汉字，行数不确定，针对此种情况，选用 LCD 显示屏的宽度为 15 个显示单位，高度的选取为不超过 10 行为合理范围，对于显示内容超过 10 行的信息，可以采用翻页的方式进行输出显示，这样就能保证显示内容均能完全输出，且无需使用体积大、价格高的 LCD 显示屏。

金鹏电子的 OCMJ8X15D 显示模块符合设计的要求。OCMJ8X15D（240x128 点阵）中文液晶显示模块[29]是一个中英文文字与绘图模式的点矩阵液晶显示模块。以 16×16 点阵的方式，可以在其上显示 8×15 个汉字，即以每行显示 15 个汉字，显示 8 行方式显示内容。如存在显示内容超过行数，则采用翻页方式进行全部显示。同时，OCMJ8X15D 可以在文字模式中，可接收标准中文文字内码直接显示中文，而不需要进入绘图模式以绘图方式描绘中文，相对于其他绘图模式点矩阵液晶显示模块的可以节省许多单片机时间，提升液晶显示中文之处理效率。同时其还提供了触控屏幕功能，以及键盘扫瞄（Key Scan）功能，可以将触摸其上的点击进行 A/D 转换得到具体的触摸位置放到相关寄存器中，单片机通过读取相关寄存器可以到点击的具体位置，以供后续处理。

OCMJ8X15D 的引脚定义如附录 1。

系统单片机与 OCMJ8X15D 引脚连接如图 3-4 所示：

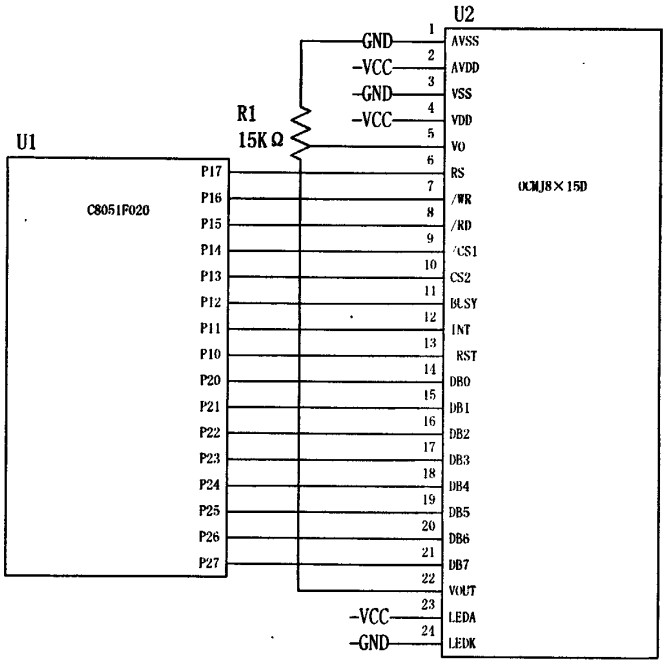


图 3-4 单片机与 OCMJ8X15D 连接图

图中 DB0~DB7 为数据传输口，与单片机 P2 口相连接。RS 为缓存器/DDRAM 选择口，高电平时表示 DB0~DB7 与 DDRAM 通信；低电平时表示 DB0~DB7 与缓存器通信。/WR、/RD 为读写控制，低电平时有效。当/CS1 为低和 CS2 为高时，模块处于致能，可接受指令，反之不可接收指令。作为和用户交互的人机界面，LCD 必须始终处于工作状态，因此芯片的/CS1 脚保持为低和 CS2 脚保持为高。对显示器的数据修改则通过单片机的 I/O 口，由系统给出相应的读/写时序，控制数据的读出和写入。

由上面分析，假设 RS、/WR、/RD 分别为缓存器/DDRAM 选择口、读控制口和写控制口取高电平的集合；$\overline{RS}$、$\overline{/WR}$、$\overline{/RD}$ 分别为缓存器/DDRAM 选择口、读控制口和写控制口低电平的集合。于是可以得到如下操作的表达式：

读缓存器操作：$\overline{RS} \cap /WR / \overline{/RD}$；写缓存器操作：$\overline{RS} \cap / \overline{/WR} / /RD$；

读 DDRAM 操作：$RS \cap /WR / \overline{/RD}$；写 DDRAM 操作：$RS \cap \overline{/WR} / /RD$。

由图 3-4 可知，OCMJ8×15D 仅与单片机的普通 I/O 口连接，不需要占用单片机的特殊资源，为系统节省了其他特殊功能端口。

主控模块在对液晶屏缓存器写入数据操作时，保持 CS1、RS、WR 引脚为低

电平，并置 RD 引脚为高电平。主控模块的 I/O 端口与液晶屏数据线相连接，首先将缓存地址通过数据线送出，然后在送出需要写入的数据信息。写入完毕，各引脚回复到高电平。如图 3-5 所示。

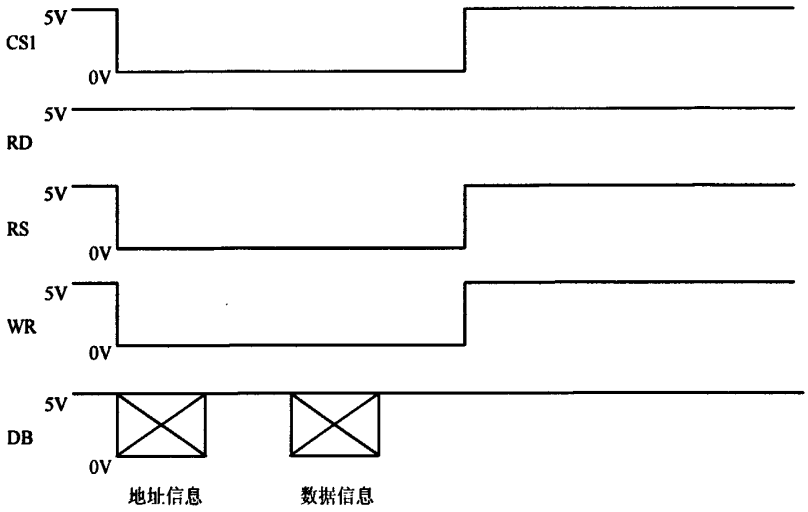


图 3-5 人机界面写缓存器时序图

其他缓存器的读及 DDRAM 的读写操作与缓存器写类似，只是控制引脚的状态不同（参照上面的分析），写入方式相同，故不在此赘述。

3.2.3 大容量外部存储模块

由于系统的功能需求是由数据库信息动态组态实现的，各种动态组态的信息量非常大，单片机自带的 RAM 容量无法满足系统的需求，因此需要大容量外部存储设备的存储数据库信息。当系统需要某个特定功能组态信息时，从系统处理的容量和速度两方面考虑，都不可能将大容量外部存储设备的所有数据全部读入内部 RAM。实现的方法是先对保存在外部存储设备上的数据库信息进行查找匹配，找到所需数据库信息后，将其调入到单片机 RAM 内存中，供主控程序解析，实现相应的组态功能。同样，当系统需要对数据库信息进行主动修改时，也是先在外部存储设备上查找到所需写入信息的结构位置，将需要修改的数据结构调入 RAM 内存中，待修改结束后再重新写入外部存储设备中保存。因此系统需要内部 RAM 和外部存储设备协同工作。

大容量可读写存储设备分为多种：EEPROM[30]，CF[31]，SD[32]等。

EEPROM （Electrically Erasable Programmable Read-Only Memory），电可擦可编程只读存储器——一种掉电后数据不丢失的存储芯片。EEPROM 可以在电脑上或专用设备上擦除已有信息，重新编程。一般用在即插即用（Plug & Play）接口卡中，用来存放硬件设置数据；在防止软件非法拷贝的"硬件锁"上面也能找到它。但 EEPROM 存在缺点，例如写入速度慢，容量小等。

CF 卡（Compact Flash）是 1994 年由 SanDisk 最先推出的。CF 卡具有 PCMCIA-ATA 功能，并与之兼容。CF 卡采用闪存（flash）技术，是一种稳定的存储解决方案，不需要电池来维持其中存储的数据。对所保存的数据来说，CF 卡比传统的磁盘驱动器安全性和保护性都更高；比传统的磁盘驱动器可靠性高 5 到 10 倍，而且 CF 卡的用电量仅为小型磁盘驱动器的 5%。但 CF 也同样存在缺点：

1、容量有限、体积较大。与其他种类的存储卡相比，CF 卡的体积略微偏大，容量较小。

2、性能限制。CF 卡的工作温度一般是 0-40 摄氏度。因此 0 度以下的环境中，CF 基本无法使用。

SD 卡（Secure Digital Memory Card）是一种基于半导体快闪记忆器的新一代记忆设备。SD 卡由日本松下、东芝及美国 SanDisk 公司于 1999 年 8 月共同开发研制。拥有记忆容量大、数据传输率高、移动灵活和数据安全等特点。通过 9 针的接口界面与专门的驱动器相连接，不需要额外的电源来保持其上记忆的信息。而且它是一体化固体介质，没有任何移动部分，所以不用担心机械运动的损坏。随着 SD 卡在现代的大量使用，其价格与 CF 已经十分接近。

综合性能、价格多方面考虑，决定采用 SD 卡作为大容量外部存储设备。

SD 卡有两种总线模式，即 SD 总线模式和 SPI 总线模式[33]。其中 SD 总线模式采用四条数据线并行传输数据，数据传输速率高，但是传输协议复杂，只有少数单片机才提供有此接口，而用软件方法模拟 SD 总线又比较繁琐，并会降低 SD 卡的数据传输速率；而 SPI 总线模式只有两条数据传输线，数据传输速率较低，但绝大多数中高档单片机都提供 SPI 总线，也易于用软件方法来模拟，此外，

SPI 总线模式的传输协议简单，易于实现。C8051F020 带有此总线接口，因此，本设计采用 SPI 总线模式。

SPI（Serial Peripheral Interface，串行外围设备接口总线）总线技术是 MOTOROLA 公司推出的一种同步串行总线接口，它是目前单片机应用系统中最常用的几种串行扩展接口之一。SPI 总线主要通过三根线进行数据传输：同步时钟线 SCK，主机输入/从机输出数据线 MISO、主机输出/从机输入数据线 MOSI，另外还有一条低电平有效的从机片选择线 CS。SPI 系统的片选信号以及同步时钟脉冲由主机提供。系统单片机与 SD 卡的连接如图 3-6：

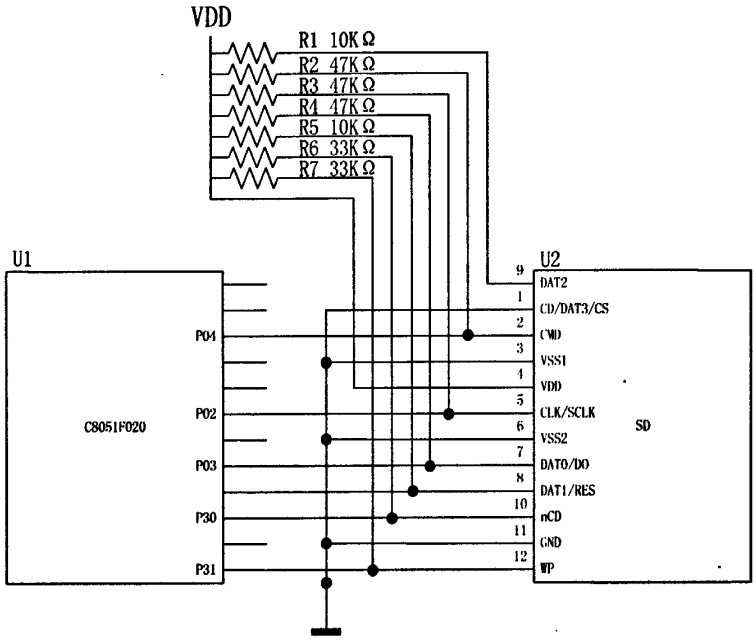


图 3-6 单片机与 SD 卡连接图

3.2.4 I/O 端口模块

考虑到系统 I/O 端口的扩展升级的灵活性，本文专门设计一个 I/O 端口控制模块。I/O 端口扩展板与主控模块之间采用双 CPU 结构。主控模块的 CPU 负责管理系统整体功能。扩展板的 CPU 管理 I/O 端口信息。主控板、扩展版设计图分别如附录 2 主控板电路原理图、附录 3 扩展板电路原理图所示。

扩展板的主要功能是对外部端口进行检测并将检测结果返回给主控板，或者对主控板的控制信息进行输出控制，功能比较单一，因此单片机功能不需要太多。

美国 ATMEL 公司推出的 AT89 系列单片机以 MCS-51 为内核，即为传统意义上的单片机。AT89 系列单片机可以用 C 语言进行程序开发，且价格便宜。因此 I/O 端口模块的单片机采用功能成熟的 AT89S52 芯片。

系统实现的功能主要体现在外部 I/O 端口对外部设备查询和控制上。系统在功能预定义时，只将扩展板 I/O 设备定义为基本的读写操作，具体实现何种功能则由用户确定以后，通过数据库配置信息和系统预定义的基本读写操作进行组合实现。

主控板和扩展板之间采用 RS485 通信协议传输命令。I/O 端口模块以 RS-485 串口通信方式，通过两条数据线与主控单片机模块相连。连接图如图 3-7：

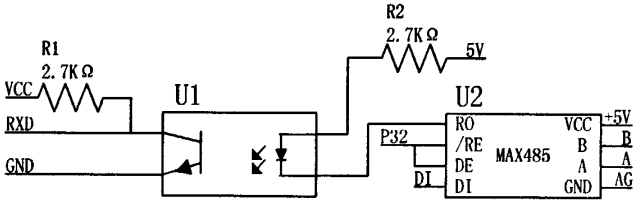


图 3-7 I/O 端口 485 连接图

主控板发来的信号利用数据线进行传输，通过 MAX485 芯片后，经过光电隔离到达扩展板单片机芯片，避免了外部传输的信号干扰对 I/O 端口单片机的影响。并且 I/O 扩展板所有的输入、输出端口都采用光电隔离技术，以提高系统的抗干扰能力。

为了使 I/O 模块能够检测、控制更多的输入输出端口，本文对单片机的部分 I/O 端口设计采用端口复用技术。下面以 P2.0 复用端口为例，讲解端口复用原理。如图 3-8 所示：

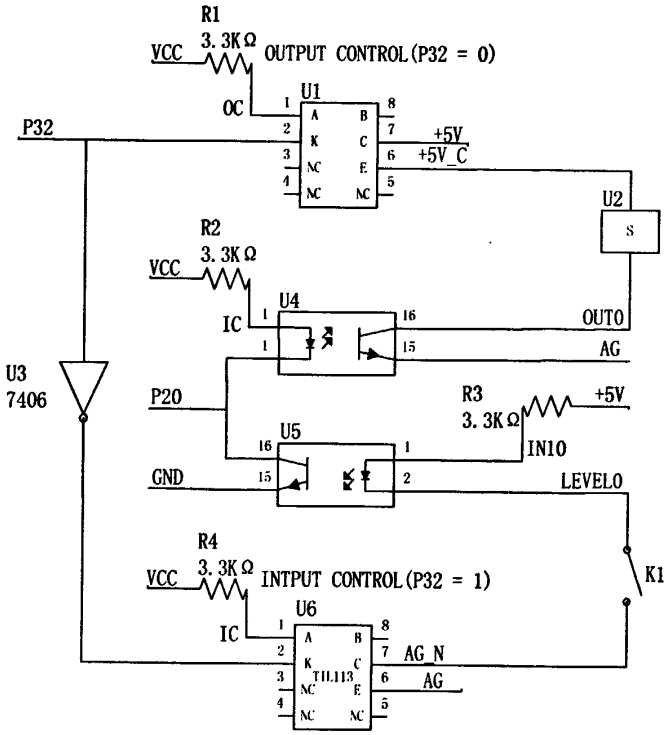


图 3-8 I/O 端口复用

利用端口 P3.2 作为输入输出复用控制端口，当 P3.2 置为低电平时，P2.0 作为输出端口功能，通过光耦的发射极驱动外部设备；当 P3.2 置为高电平时，P2.0 作为输入端口功能，处于输入查询状态，通过光耦的输入端读取外部输入端口的开关状态。

具体工作流程如下：当 P3.2 置低时，OC 端为高电平，U1 输入端光耦导通，此时输出端+5V_C 与+5V 接通，输出回路由 5V 电源供电。此时 OUT0 状态与 P2.0 同状态，故 P2.0 可以作为 OUT0 的输出控制信号，当 P2.0 为高时，控制设备 S 两端电压差为 5V；当 P2.0 为低时，控制设备 U2 两端无电压差。由于 P3.2 的电平经过反相器反相，在 U6 的第二脚上为高电平，U6 输入端的发光二极管截止，故输出端 AG_N 为高阻态。此时，无论 K1 是否闭合，LEVEL0 均为高阻态，U5 输入端不导通，故无法影响到 P2.0 状态，此时端口输入不影响输出状态。如果 P3.2 置高，分析方法一样。输入输出端口复用流程如图 3-9 所示：

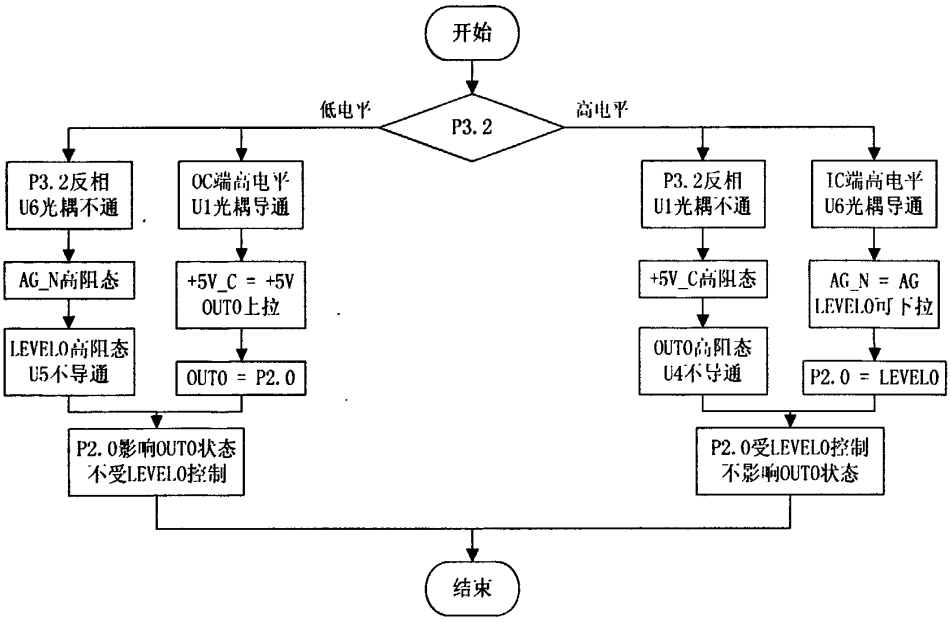


图 3-9 I/O 端口复用流程图

通常单片机的输入信号分为两类：一类是快速瞬时跳变信号，另一类是慢速电平保持信号。检测这两类不同信号，单片机的 I/O 端口引脚需要采取不同的连接方式。对于快速瞬时跳变信号，不宜采用 I/O 端口复用模式。若采用复用模式，可能会发生在端口处于输出时未能对快速信号进行有效获取，而在信号在端口转变为输入状态时可能已经消失，这样就会发生信号的丢失。对于慢速电平保持信号，则可以采用复用模式，采用复用模式时，可以等待输出端口将需要发送的信号发送完毕后，再去检测信号的状态。因此系统设计时，保留了一部分纯输入功能的输入端口。对两种不同的检测的信号，分别对待，采用不同的硬件端口及软件检测方法。

4 系统的软件设计

4.1 软件设计思想

单片机控制系统软件的主要功能是检测、处理按键事件的响应，根据用户需求查找所需数据库信息，检查系统输入端口状态，按组态信息进行各种功能及输出端口的控制。系统软件设计框图如图 4-1 所示：

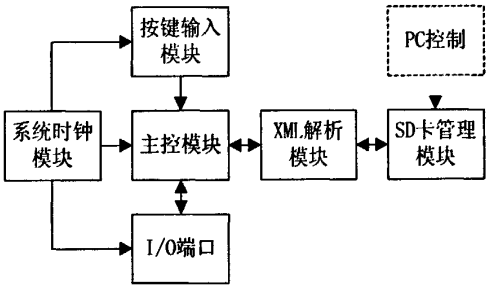


图 4-1 系统软件设计框图

4.2 系统软件各模块设计

4.2.1 单片机核心模块程序流程及实现

主控模块作为系统的核心部分，负责以下功能：系统运行流程的控制、系统配置数据库信息的修改、键盘输入操作的处理及显示信息的输出、串行通讯的建立、系统控制命令向外部端口的发送、外部端口应答命令的响应、接口扩展等。

系统主控模块运行流程如图 4-2：

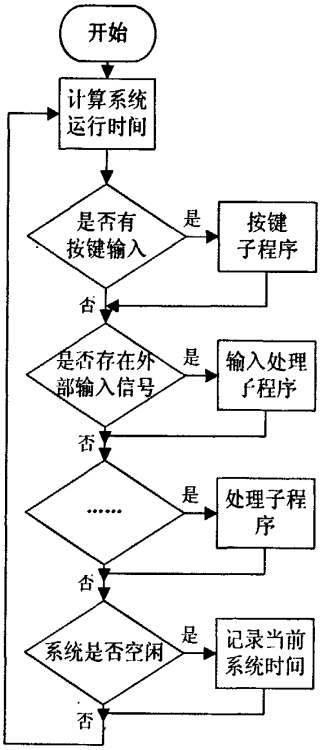


图 4-2 系统核心模块运行流程图

主控模块接收到按键模块的请求处理信息或 I/O 端口的检测状态信息（例如开机信号）后，进入相应的服务处理子程序。

首先系统主控模块计算当前系统工作时间，为需要进行时间控制的任务提供准确的时间信息。然后检测是否有按键输入，若检测到有按键输入，进入按键响应子程序，根据数据库信息组态完成相应的操作。如果没有输入按键或按键处理子程序运行结束，检测是否是有外部输入端口输入信号。若存在端口输入信号，进入端口输入处理子程序。否则将系统时间保存到数据库信息中，以备查询或调用。系统进入下一次循环。

子程序中获取到主控模块发送的请求命令后，通过 XML 解析模块在 SD 卡上查找所对应的数据库信息，将其翻译成所单片机能识别的命令和数据，根据需要送到显示页面显示或输出端口控制外部设备。

主控模块和外部端口之间使用 RS485 串口工作模式进行通信，以位可变波特率方式双向通信。

通信协议定义如表 4-1 所示：

表 4-1 通信协议定义表

	起始码	485 站号	命令	模式	设备编号		结束码	
代码	3A						0D	0A

定义解释如下：

（1）通信协议总长度为 8 字节，起始码 0x3A 开始，一条发送指令占各模块接收缓冲区 7 个字节。

（2）模块接收命令结束以结束码 0x0D、0x0A 为标志。当正确接收到结束码后，各模块对缓存内容长度进行判断。若起始码与结束码之间的内容长度超过 5 位，则表示接收错误，命令自动丢弃；若内容长度正确，则对按照各位标志对命令进行识别、判断。

（3）在控制系统中，以不同的 485 站号区分不同的系统模块。例如：以 0x50 表示主控模块，以 0x51 表示外部端口模块。

（4）由命令和模式两部分共同构成系统的输入、输出控制命令，具体配置如表 4-2 所示：

表 4-2 命令、模式定义表

命令	模式	功能
00	00	发送操作命令
00	01	发送读取命令
01	00	返回读取值
02	00	返回执行失败消息
02	01	返回执行成功消息

（5）设备编号是决定某一模块上具体设备的编号，针对不同设备，通过 485 站号、命令、模式和模块号可以唯一决定一台具体设备。

（注：因为某些系统设备为只读设备，某些设备为只写设备，故相同 485 编号、及设备编号的可能存在着两台完全不同的设备。）

通过系统的通信协议的定义，主控模块和外部模块之间通过有效的协议通信，能实现对外部模块各个设备进行有效的输入输出控制，以达到控制系统功能的目的。

4.2.2 LCD 显示及键盘扫描方法

人机界面：由液晶触摸屏实现，对系统进行信息的显示和参数的设定。显示信息包括：系统当前工作状态、内部参数、外部 I/O 端口状态等。键盘功能实现系统工作参数设定、显示页面切换等功能。

人机界面的初始化如图 4-3 所示：

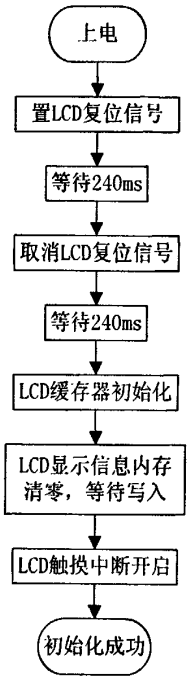


图 4-3 人机界面上电初始化过程

系统上电后，主机向 LCD 发送 240ms 的复位信号，以保证液晶屏能完成复位动作，并清空缓存器原有数值。复位完毕后，取消复位信号，进行 LCD 工作初始化。首先，先对缓存器进行初始化，对保存 LCD 的电源模式、显示工作模式、字体显示类型、大小等一系列参数进行设置。为 LCD 后续操作做准备。然后，将保存显示信息的 DDRAM 清零，为写入显示内容做准备，与此同时，开启液晶屏显示，等待信息的写入，并进行显示操作。最后，开启触摸屏中断，等待触摸屏的输入信号。

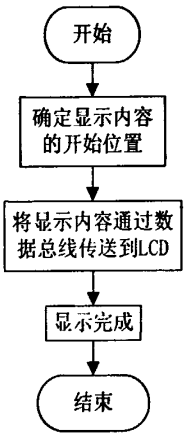


图 4-4 液晶显示屏内容显示过程

当存在字符串需要显示在 LCD 显示出来时，先确定其显示内容的第一个字符在 LCD 上的坐标位置，将坐标位置通过总线，传送到 LCD 上，进行定位。然后，将显示内容字符串通过数据总线传送给 LCD 模块，LCD 模块将其显示出来。过程如图 4-4 所示。

对于按键输入模块，先将液晶触摸屏页面按显示文字信息的行及半角像素点大小对显示范围进行单元格划分，使屏幕上每一个像素点的位置都有一单元格与之相对应，称为按键单元格映射。

按键处理流程如图 4-5 所示：

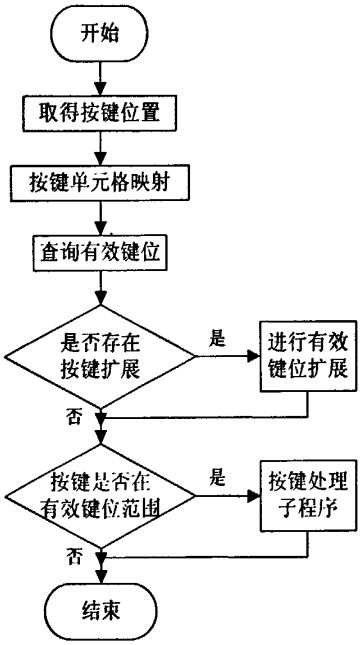


图 4-5 按键处理流程

系统检测到有按键事件触发时，通过触摸屏硬件模块按键位置的 A/D 转换结果，获得该按键在液晶屏的相对坐标，经按键单元格映射，得到按键的单元格位置，再通过查找数据库信息中对应页面的按键单元格定义的功能进行匹配，查找是否存在有该按键单元格的功能定义，若该单元格有定义的功能，则执行数据库信息中对应的事件处理命令。

为了防止误操作，在软件实现时，添加了可识别多次连击的功能。当用户点击按键时，可以通过设置单个按键点击的时间长度区分用户输入的次数。并通过参数设置，可以对按键定义是否扩展其有效范围。

在按键输入模块中有两个功能函数：分别是设定按键扫描的 KeyScan 函数和判断按键位置扫描的 InKeyScale 函数。

通过 KeyScan 函数中参数的设定，用户可以对键盘输入进行众多的特性设置，例如：通过 KeyScan 函数中的标志位 KeyOn 变量的置位与否，可以设定是否对按键进行响应；通过 singleKey 变量可以设定是否允许按键的连发操作；通过设置键盘扫描按键的次数，可以设定识别按键连击的时间。

在位置扫描函数 InKeyScale 中，将按键定义的位置与触摸屏检测的实际点击位置带入函数进行判断，实际点击位置是否在按键定义范围内。判断的同时还要考虑按键位置扩展的行扩展 wKey 及列扩展 hKey 变量对按键定义范围的修正。如若 wKey 或 hKey 变量分别被置为 1，则表示按键有效的左右邻域或上下邻域范围增加了 1 位。若 wKey 及 hKey 变量均被置为 1，则表示按键的有效范围增加了一个八邻域。将触摸屏检测出来的实际点击位置放入修正后的按键定义范围进行判断，检测其是否是为有效点击。这样可以通过调整按键大小，修改按键的实际功能范围，避免按键过小所带来的误操作，同时也可以按实际需要进行修改按键，避免按键定义的重叠。识别过程如图 4-6 所示：

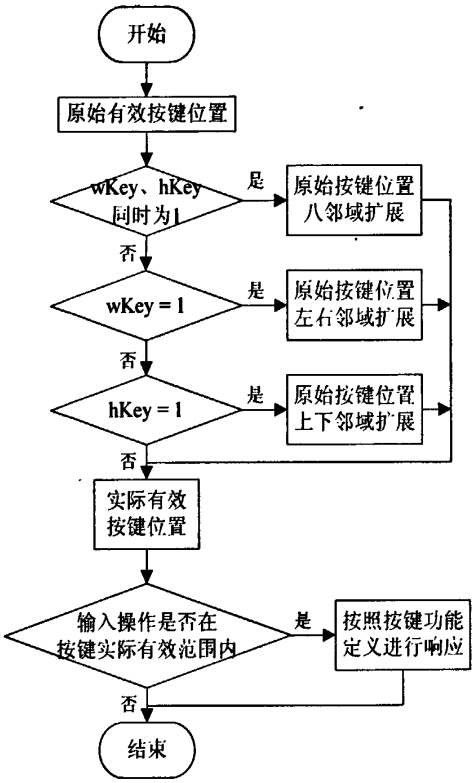


图 4-6 按键识别过程

4.2.3 大容量外部存储模块读写和文件管理

读大容量外部存储 SD 卡的软件支持分为两个部分：

1、单片机按照 SPI 模式对 SD 卡实现读写操作。

2、单片机对 SD 卡的读写操作的内容按照文件格式实现。

下面分两部分对 SD 卡的两部分实现分别做出介绍[33]：

1、SPI 模式的实现

SPI 总线模式的数据是以字节为单位进行传输的，每字节为 8 位，每个命令或者数据块都是字节对齐的（8 个时钟的整数倍）。主机与 SD 卡的各种通信都由主机控制，主机在对 SD 卡进行任何操作前都必须先要拉低 SD 卡的片选信号 CS（card select），然后由主机向 SD 卡发送命令，SD 卡对主机发送的任何命令都要进行响应，不同的命令会有不同的响应格式（1 字节或 2 字节响应）。SD 卡除了对命令响应外，在执行写操作时，还要对主机发送的每个数据块进行响应（向主机发送一个特殊的数据响应标志）。

SPI 通信使能，将主控单片机设置为主模式。

SPI 总线模式下的所有命令都是由 6 个字节构成，且发送时高位在前，其命令格式如表 4-3 所列。

表 4-3SPI 命令格式

Byte 1			Byte 2-5	Byte 6	
7	6	5~0	31~0	7	0
0	1	Command	Command Argument	CRC	1

其中，7 位 CRC（Cyclic Redundancy Check——循环冗余校验）校验位可以全部写入 0，因为默认情况下，SPI 总线模式无需 CRC 校验。

SD 卡的 SPI 模式的实现主要包括两部分内容：SD 卡的上电初始化过程和对 SD 卡的读写操作。

1）SD 卡的上电初始化过程

上电过程流程如图 4-7 所示：

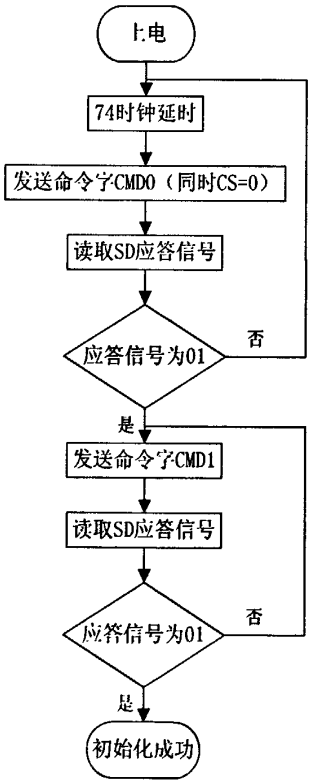


图 4-7 SD 上电初始化过程

这里给出上电初始化的时序图，下面结合时序图说明上电初始化程序流程。

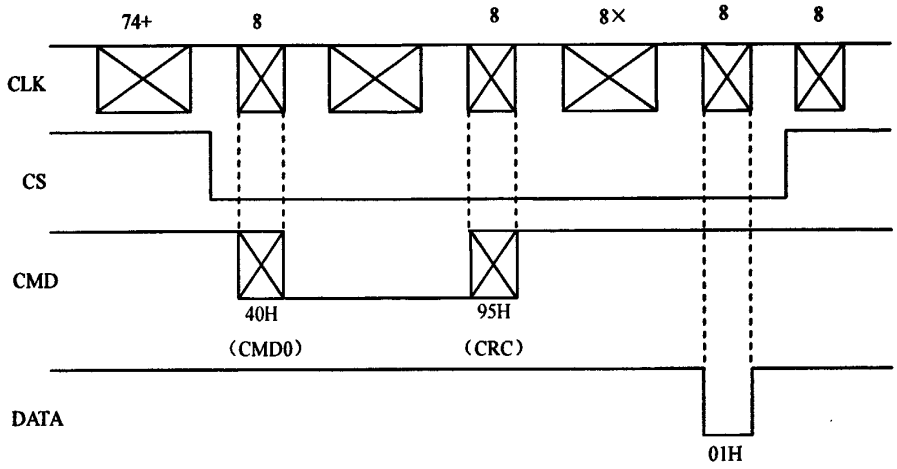


图 4-8 SD 卡复位时序图

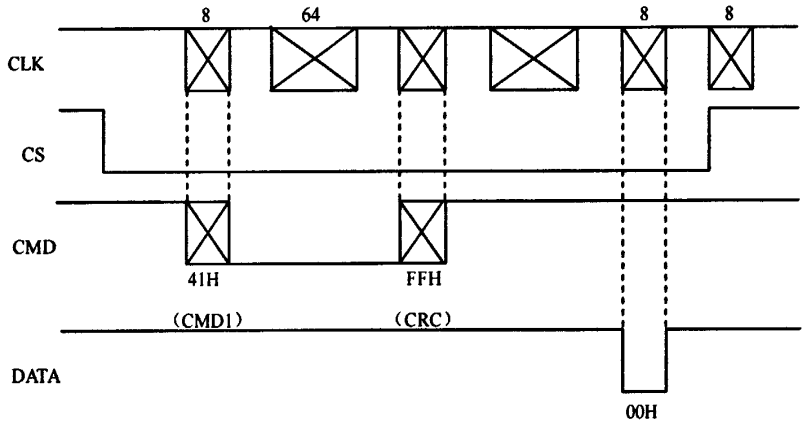


图 4-9 SPI 模式初始化时序图

SD 卡上电后，需要约 64 个 CLK 周期才能到达 SD 卡的正常工作电压，再需要 10 个 CLK 与 SD 卡同步。故主机必须先向 SD 卡发送 74 个时钟周期，以完成 SD 卡上电过程。SD 卡上电后会自动进入 SD 总线模式，在 SD 总线模式下向 SD 卡发送复位命令（CMD0），若此时片选信号 CS 处于低电平态，则 SD 卡进入 SPI 总线模式，否则 SD 卡工作在 SD 总线模式。SD 卡进入 SPI 工作模式会发出应答信号，若主机读到的应答信号为 01H，即表明 SD 卡已进入 SPI 模式，如图 4-8 所示。此时主机即可不断地向 SD 卡发送命令字（CMD1）并读取 SD 卡的应答信号，直到应答信号为 00H，以表明 SD 卡已完成初始化过程，准备好接

受下一命令。如图 4-9 所示。此后，系统便可读取 SD 卡的各寄存器，并进行读写等操作。应当注意的是：主机在向 SD 卡发送命令字 CMD0 时，SD 卡是处于 SD 总线模式的，此时要求每一个命令都要有合法的 CRC 校验位，所以，此时的命令字 CMD0 必须有正确的 CRC 校验位（其校验位为 95H）。而在发送命令字 CMD1 时，SD 卡已处于 SPI 模式，而默认的 SPI 模式无需 CRC 校验，此时的 CRC 校验位可直接写入 0。

2）SD 卡的读写操作

SD 卡的写操作流程如图 4-10 所示：

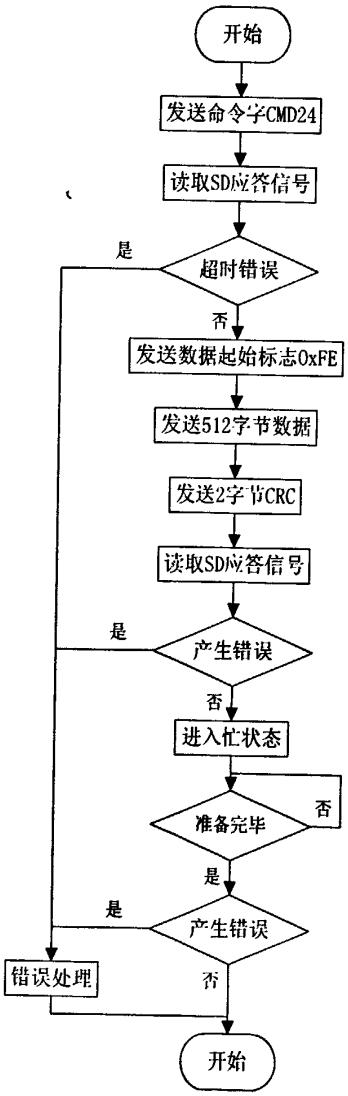


图 4-10 SD 卡写操作过程

SPI 模式支持单块（命令字为 CMD24）和多块（命令字为 CMD25）写操作。多块写操作指的是从制定的位置开始写，直到 SD 卡接收到一个停止命令（命令字为 CMD12）才停止写操作。单块的写操作数据块的长度只能是 512 字节，但由于每次写入内容信息量不大，内容大小均不超过 512 字节，故本设计采用的是单块写操作，其流程图如图 4-10 所示。操作时序如图 4-11 所示，首先向 SD 卡发送写数据块的命令字 CMD24，在接收到 SD 卡的响应信号（00H）后，再发送数据起始标志（FEH），然后发送 512 字节的数据，并后跟两字节的 CRC 校验。当 SD 卡的回应信号为 E5H 时，即表明 SD 卡可正确接收数据，之后 SD 卡的输出口变为低电平，表明正在写 SD 卡，当输出口变为高电平时表明写操作完成。SD 卡的读操作与写操作类似，故此不在赘述。

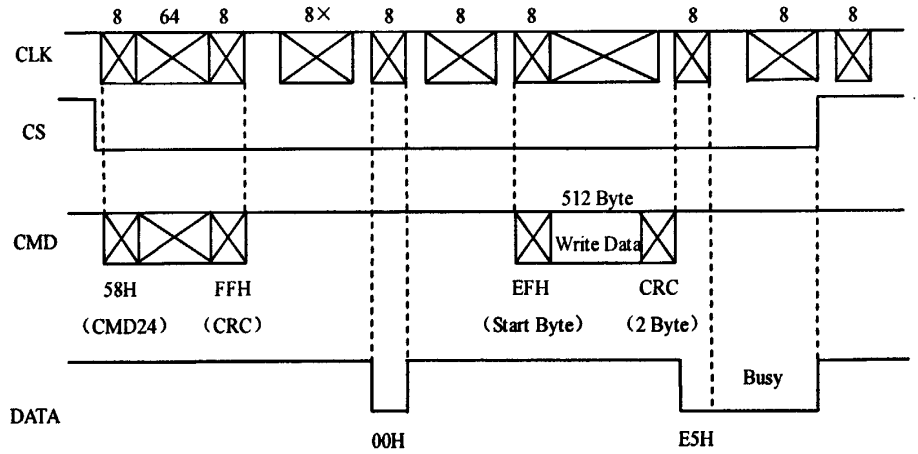


图 4-11 SD 卡写操作时序图

2、SD 卡文件格式的建立

SD 卡管理模块：SD 卡上可以采取两种模式进行信息的写入和读取：按位存放及采用文件系统的方式。

采用按位方式进行信息的存放时，将信息按位依次存放在 SD 卡上，读取信息时也是按位依次读取信息。这样操作的特点是信息存放、查找、修改效率极低。

采用文件系统方式进行信息的存放时，无需考虑信息具体存在的物理位置，只需将信息写入文件系统中，有文件系统来安排确定具体存放的物理地址，用户在读取、写入、查找时无需考虑具体的地址问题，大幅度减小了信息操作的复杂

性，大大提高了信息存取、读写的便利性，因此决定在 SD 卡上采用文件系统管理内容信息。

文件系统是指文件在磁盘中命名、组织和存储的方式，不同的操作系统所支持的文件系统也不同。文件系统简称 FAT（File Allocation Table 文件分配表），

常见的 PC 文件系统有 FAT12、FAT16、FAT32、NTFS 等[34]。

（1）FAT12 支持的分区容量小，只支持最多 32MB 的分区。

（2）FAT16 可以支持 2GB 大小空间，但其存在以下问题：

其一，FAT16 系统不能管理大容量的硬盘，2.15GB 是 FAT16 系统所能管理单个硬盘分区容量的最大限度。

其二，因为簇大小问题，FAT16 系统造成硬盘空间的大量浪费。

（3）FAT32 分区容量最大可达 16TB，其可大大提高硬盘利用率，使硬盘的可用空间较 FAT16 增加 20%～30%。而且 FAT32 可以以比 FAT16 更快的速度进行读写。

（4）NTFS 是 Windows 专门的文件系统，它比 FAT32 要更复杂，功能也更强大。它所支持的硬盘空间可高达 16EB（1EB = 1024TB）。但是，NTFS 算法过于复杂，其在硬件兼容性上远不及 FAT32。

综合上述考虑，单片机系统的 SD 卡采用与 FAT32 相兼容的文件系统。在 SD 卡上构建 FAT32 兼容文件系统，可实现与计算机共享信息。

与 FAT32 兼容的嵌入式文件系统有 UCOSII 公司的 UC/FS[35]和开源的 FatFs[36]等。其中 UC/FS 为商业软件，成本较高，故选择开源的 FatFs 文件系统[37]在单片机上实现文件系统。

FatFs 源代码包括用于硬件层的 diskio.c 和 diskio.h 文件以及 FatFs 的文件系统层和文件系统的 API 层的 ff.c 和 ff.h 文件。

Diskio.c 中包含有底层相关的硬件读写操作，例如初始化磁盘的 disk_initialize 函数、读扇区的 disk_read 函数以及写扇区操作的 disk_write 函数。将前述的 SD 卡初始化及基本读写操作替换 diskio.c 文件中的相关函数，即满足了在 SD 卡上构建文件系统的底层操作。

ff.c 文件中提供的大量 API 函数，例如打开文件的 f_open 函数、关闭文件的 f_close 函数、读取文件内容的 f_read 函数以及写入文件内容的 f_write 函数等。通过这些操作，可以极为方便的对 SD 卡中保存的内容按照文件格式进行进行操作和修改。

至此，在 SD 卡上构建了与 Windows 操作系统相互兼容的文件系统，故保存数据库信息的文档可以在计算机中查看及修改，能简单方便地实现系统功能修改及升级，体现了系统动态组态文件修改的便利性。

4.2.4 I/O 端口模块

系统的功能包括对外部端口的检测和控制。在主控板上，仅保留集成在 SoC 芯片上的 A/D、D/A 等特殊功能模块，其他的数字量读取和控制部分均在扩展板上实现，这样如果以后系统需要扩展 I/O 数字端口读取和控制量，可以方便的进行扩展，系统升级方便。下面对扩展板进行介绍。

扩展板是由一个管理 I/O 端口信息的 CPU 作为核心控制芯片来控制 I/O 端口模块响应主控模块命令输入或者输出请求，并对外部 I/O 模块进行查询或输出响应的集成模块。同时，扩展板将外部 I/O 模块查询的结果通过系统自定义协议返回给主控 CPU。主控板和扩展板之间采用 RS485 通信协议传输命令。

端口的引脚信号采用扫描的方式进行检测，当检测到引脚信号发生有效变化时，置位反映引脚状态的标志位，无论引脚以后状态变化，在有主控模块查询命令查询该标志位之前，标志位不发生任何改变。当有查询命令需要查询端口状态时，直接访问读取端口信号标志位信息，并将信号标志位清零，以备下一次引脚有效信号的保存。

I/O 端口工作流程如图 4-12 所示：

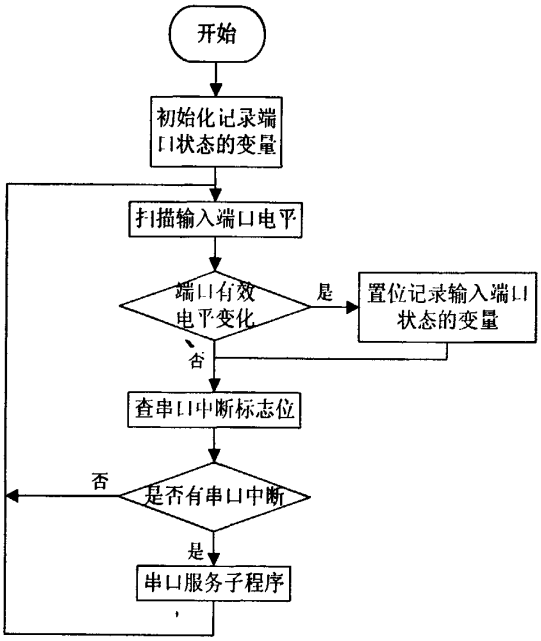


图 4-12 I/O 端口工作流程

I/O 端口模块运行时，首先清零初始化保存端口信息的标志位，并扫描识别当前各端口状态。若检测到端口的状态为有效信号时，置位对应端口信号标志位，以备查询。查询是否有串口中断标志位置位，如有置位，则进入串口服务子程序；如若没有，则系统继续循环检测端口状态。

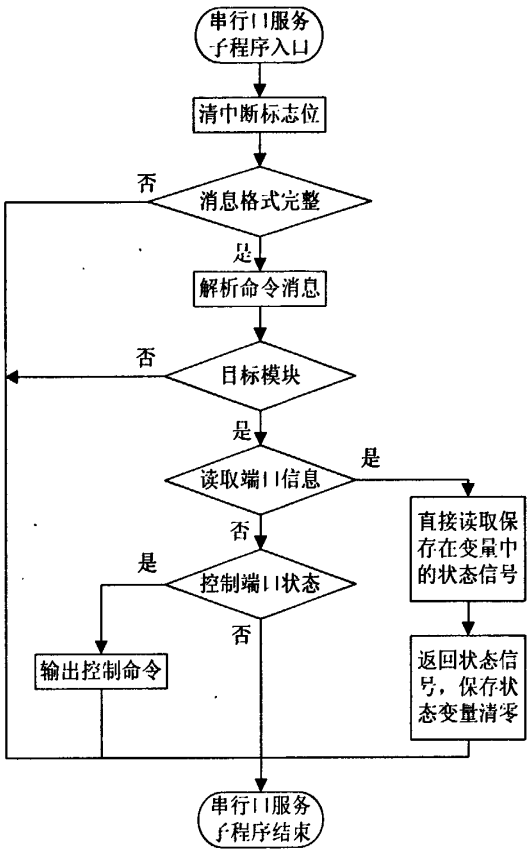


图 4-13 串行口服务子程序

当存在串口中断时，扩展板进入串口服务子程序。进入子程序后，首先清除中断标志位，进行传输命令完整性判断，检查其是否符合先前定义的命令格式。若不符合，则丢弃消息，结束服务子程序；如符合命令格式，则进行下一步操作。

如果消息格式完整，对命令消息进行识别解析。识别解析后，若命令的目标模块不是本端口模块，对命令进行丢弃，模块不做任何处理；若命令的目标模块为本模块，对后面的命令及模式进行识别判断。根据表 4-2，查找出命令及模式号对应的系统命令，确定是读入还是写出操作。最后，再根据设备编号查找到具体的设备端口，进行读入或写出操作。若是读取操作，则将读取到的信息返回到主控板。服务子程序结束。过程如图 4-13 所示。

5 系统配置的数据库文件的设计与实现

5.1 系统数据库文件的设计思想

基于数据库的多 MCU 控制系统，数据库文件决定着系统的具体功能。系统软硬件基本模块，根据数据库配置，进行动态组合，针对不同需求可以快速组建单片机控制系统。

作为系统配置文件 INI 文件和 XML 文件均可以表示系统的配置文档，但两种文件各具特点。INI 文件格式简单，没有标准的固定格式，通常文件信息较小，方便存储。而 XML 文件有标准的定义格式，高度结构化、层次化的数据组织形式，用户可以根据标准格式自行定义 XML 文件的具体结构，通过自定义文件结构用户可以存储容量大、稳定的的 XML 文档。通过标准化的结构化的 XML 文档信息，可以方便用户查看、手动修改配置文件信息，也可以方便解析程序读取、解析信息。

由于 XML 文档的以上特点，故数据库文件采用 XML 文档形式保存。系统数据库配置信息是按照不同的功能模块进行区分，都按照 XML 的规范格式，预先在 PC 机上编译完成，经系统软件下载到大容量外部存储模块上保存。

单片机在系统运行时，检索、读取 XML 文档上的配置信息，并将重要的配置信息写回 XML，以备调用。

单片机系统与 XML 文档系统没有直接的接口方法实现读写操作，系统对数据库文件中所包含的信息进行查找和修改，必须经中间解析过程。XML 解析模块的引入可以解决单片机和 XML 文档系统的接口问题。XML 解析模块接收到主控模块查询令后，提取 XML 文档中所需信息，对读取的信息根据 XML 的各种解析命令进行解析，将解析的结果放入系统内存，供主控模块调用。同时，解析模块还能实现 XML 中节点的元素、属性值按主控模块要求进行修改，将修改后的数据库文件保存到 SD 卡上功能。

5.2 XML文档语法简介

XML 文档的基本语法由 W3C 创建的一个文档所定义的一组规则所规定。在深入了解 XML 文档结构之前，先需要了解以下几条基本语法规则[38]：

（1）XML 区分大小写，元素的起始标注必须正好匹配结束标注。

（2）所有起始标注必须有结束标注，即所有 XML 文档都必须包含一个根元素，这个根元素是包含了文档所有内容的最外层元素。

（3）元素必须正确的嵌套。文档中元素不是孤立的（除非只有一个元素——根元素），所有元素都必须包含在以根元素开始的元素层次中。

（4）XML 用五种预定义实体“<”、“>”、“&”、“'”和“"”来分别代替“<”、“>”、“&”、“’”和“””字符。

（5）XML 说明优先，即如果使用一个 XML 说明，它必须最先出现。

5.2.1 文档说明

XML 节点包括一下几种类型：

（1）注释类型，形如 <?……> 。

（2）完整节点，形如 <……/> 。

（3）结束节点，形如 </……> 。

（4）父节点，形如 <……> 。

以上四种节点类型为 XML 文档所会遇到主要类型，XML 解析器的设计也是针对以上几种不同节点类型做主要考虑。

5.3 XML解析模块

5.3.1 解析原理

XML 解析器是 XML 应用的基础。XML 本身只是以纯文本对数据进行编码的一种格式，要想利用 XML，或者说利用 XML 文档中所编码的数据，必须先将数据从纯文本中解析出来，因此，要求必须有一个能够识别 XML 文档信息的文本文件阅读器（即 XML 解析器），用来解析 XML 文档并提取其中的内容。

显然，XML 解析器在 XML 应用程序中有着重要的地位。随着 XML 越来越广泛地被采用，高效解析 XML 文档也变得越来越重要，尤其是对于那些要处理大量数据的应用程序，这种技术尤为重要。选择合适的 XML 解析技术对应用系统的性能有着较大的影响。不正确的解析会导致过度的内存消耗和过长的处理时间，从而大大降低系统的整体性能[39]。

XML 解析器（处理器）用来读取 XML 文档，利用它可以访问文档的结构和内容。假定 XML 处理器代表应用程序执行了这个工作。根据 XML 处理器如何读取 XML 文档中的数据库信息以及它需要向应用程序提供的信息，规范描述了 XML 处理器所必需的行为。读取一个 XML 数据的过程称为解析。

另一个方面，可以这样定义解析：解析是一个过程，即把信息流按照它的组成（通常称为标记）进行分解。在 XML 环境下，解析就是扫描 XML 文档（可以是物理文件，也可以是数据流）的过程，它把 XML 文档分解为不同的元素（标记）以及相应元素的属性。由于元素的嵌套暗示了某种分层结构，所以解析 XML 可以揭示信息的结构。如果 XML 文档不遵循在 XML1.0 规范中描述的格式规范规则，那么这种文档有可能不会被完全解析。一个被成功解析的 XML 文档要么是格式规范的，要么是有效的。

XML 解析器根据是否验证合法性，可分为验证性和非验证性解析器；而根据解析方式的不同，又可分为基于树的解析器（DOM）和基于事件的解析器（SAX）。

5.3.2 常用 XML 解析方式

1、基于树的文档对象模型（DOM）

DOM 是一种基于树型的解析技术，将 XML 文档一次性解析，生成一个位于内存中的对象树用以描述该文档。它可以实现对整个 XML 文档的全面、动态访问。

DOM 以及广义的基于树的处理具有几个优点：

（1）由于树在内存中是持久的，因此可以修改它以便应用程序能对数据和结构做出更改。

（2）它还可以在任何时候在树中上下导航，而不是像 SAX 那样是一次性的处理。DOM 使用起来也要简单得多[40]。利用 DOM，开发人员可以动态地创建 XML，遍历文档、增加/删除/修改文档内容，具有较好的导航能力。

DOM 解析存在如下一些问题：

（1）DOM 构建整个文档驻留内存的树。在内存中加载整个文档和构建完整树结构成本很高，尤其当文档非常大的时候。典型的 DOM 树的容量比文档容量要大一个数量级，所以它要消耗大量内存。

（2）整个 XML 文档必须一次解析完成，不可能只做部分解析。如果只需关注 XML 文档的一小部分，那么创建那些永远不被使用的对象是极其浪费的；DOM 解析器必须在应用程序代码取得控制权之前读取整个文档。对于非常大的文档，这会引起显著的延迟。

2、基于事件的简单应用编程接口（SAX）

SAX 是一个用于处理 XML 的事件驱动的“推”模型。它不是 W3C 标准，但它是一个得到了广泛认可的使用 Java 开发的 API，大多数 SAX 解析器在实现的时候都遵循标准。SAX 解析器不像 DOM 那样建立一个整个文档的树型表示，它采用回调机制，在读取文档时激活一系列的事件，这些事件被推给事件处理器

SAX 的优点就在于[41]：

（1）它能解析任何大小的文件。因为没有必要把整个文档加载到内存中，所以 SAX 对内存的消耗量比 DOM 小得多，并不会随着文件变大而增加。

（2）SAX 无需像 DOM 那样为所有节点创建对象，开发人员可以根据需要创建自己的 XML 对象模型。

（3）SAX 本身很简单，而且速度快。

SAX 的缺点如下：

（1）不能随机访问文档。

（2）当文档包含许多内部交叉引用时，SAX 的实现是很困难的。

图表（表 5-1）对两种方式的优缺点进行了归纳。

表 5-1 两种解析方式的比较

优缺点 / 解析方式	优点	缺点
DOM	易用性强，遍历简单，支持 XPath	解析速度太慢，内存占用过高（原文件的 5x~10x），对于大文件来说几乎不可能使用
SAX	解析速度快，内存占用不与 XML 大小相联系（可以做到 XML 涨内存不涨）	易用性差，因为没有结构信息，无法遍历，不支持 XPath，可维护性差

5.3.3 基于单片机的 XML 解析器的设计

针对上文所述的两种主流 XML 解析器，结合单片机平台，得出以下结论：

（1）文档对象模型（DOM）的解析方式虽然能够快速、方便对整个 XML 文档的数据和结构做出定位，并进行读取和修改。但无论操作内容多少，必须对整个 XML 文档进行一次解析，不可能只做部分解析。整个 XML 文档驻留内存，对于单片机小内存平台无法实现。

（2）基于事件的简单应用编程接口（SAX）的解析方式，它不必像 DOM 方式那样，将整个 XML 文档读入内存，可以只访问 XML 文档中用户感兴趣的部分文档，从而显著提高内存利用效率和运行速度。SAX 本身很简单，而且速度快。但现有 SAX 解析函数是基于 JAVA 接口的解析函数，需要操作系统及开发环境支持，对于内存空间小、处理速度相对较慢的单片机环境，移植不现实，故无法在单片机平台上直接使用。

针对以上两种主流 XML 解析器的设计，考虑采用 SAX 的设计思想，对 XML 文档进行基于文档流的分析，发现特殊的符号的时候，它会产生相关的事件，应用程序对相关事件进行处理分析，获取所需要的内容数据。

解析器将 XML 文档当作流读入。当解析器遇到某一节点时，就会产生一个事件，然后将该事件发送给处理类。

在 XML 文档中主要内容形式为节点，节点可以分为元素节点、属性节点及处理指令节点等。节点也可以内嵌有其他节点。内嵌节点为子节点，包含内嵌节

点的为父节点。故 XML 解析器可以采用节点操作的方式对 XML 文档进行解析。

基于 SAX 解析的思路，在设计 XML 解析器时，首先为 XML 文档设定一个文档指针，指向当前处理节点。因为 XML 文档是按照流的方式进行处理，不是整个文档被保存在内存中，该指针只能做单向处理移动，不可以进行前向移动。

在实际查找所需节点时，按两种方法进行查找：

一、对 XML 文档按属性或属性进行分析，读取元素或属性，比较元素或属性与所需查找值是否一致，一致则返回该节点，不一致则读取下一同级节点，再进行比较，以此循环，直至查找到指定元素或同级节点查找完毕为止，如图 5-1 所示：

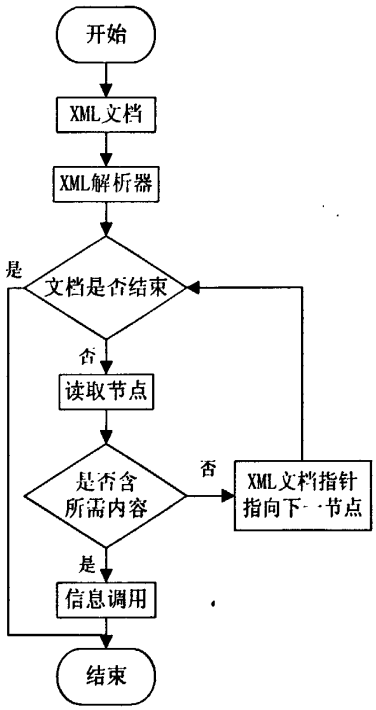


图 5-1 同级节点检索流程

二、根据节点之间父子关系，直接进行节点查找。若需查找子节点，顺序遍历同级节点，直至子节点为止。同理可查找到父节点。如图 5-2 所示：

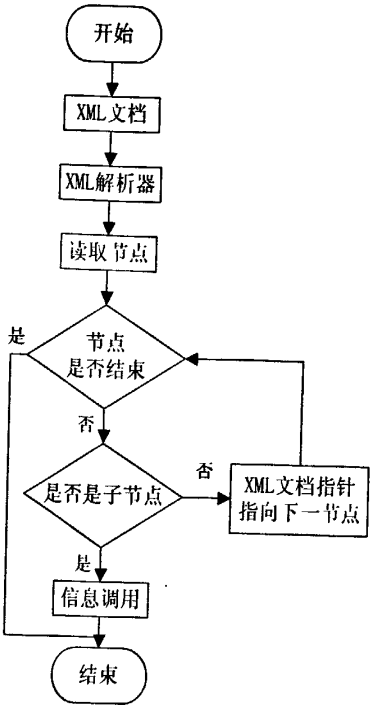


图 5-2 子节点检索流程

根据以上思想结合实际使用，从规范外部接口考虑，XML 解析器设计七个操作函数：用于遍历节点的 FindElem 函数，其目的为查找下一个为非注释节点的同级节点，或查找带有参量的节点，直至查找到该节点为止；用于检索子节点的 IntoElem 函数，其目的为查找当前节点的子节点；用于退出到父节点的 LeaveElem 函数，其目的为遍历同级节点，直至退出到父节点；用于读取当前节点元素属性值和设置当前节点元素属性值的 GetAttrib 和 SetAttrib 函数；因为节点操作的单向性，故还需要保存 SavePos 和恢复 RestorePos XML 文档指针的两个函数，方便定位 XML 节点位置，提高系统运行效率。

解析函数实现方法如图 5-3 所示：

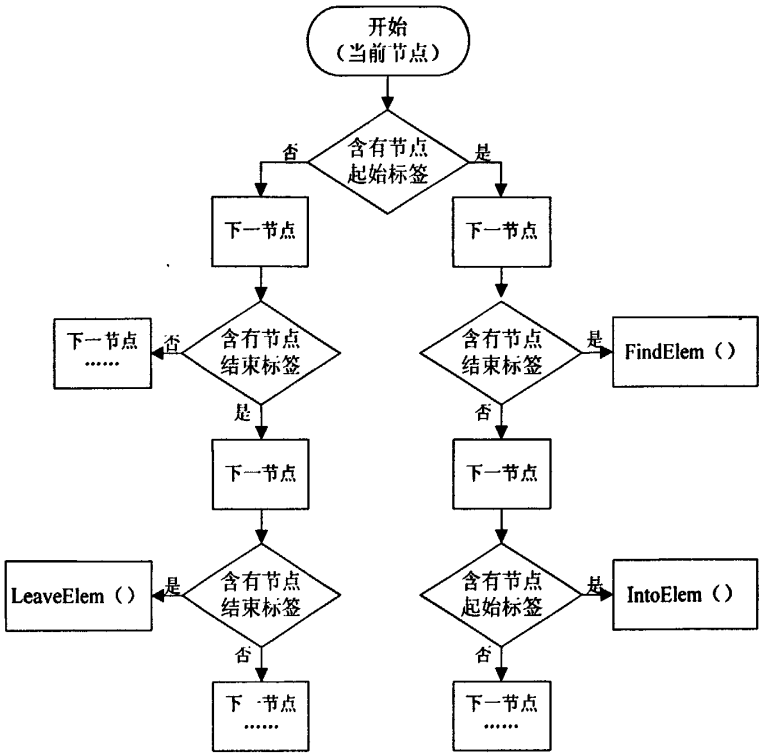


图 5-3 XML 解析函数实现方法

（1）FindElem 函数。用于在 XML 文档中查找到当前节点的下一同级节点或含有特定参数的目标节点。

函数首先检索节点起始标签（例如“<”），节点起始标签和节点结束标签（例如“/>”或“</”）之间即为节点的元素及属性部分。若 FindElem 函数不存在形参变量，则将 XML 文档指针指向当前节点。若 FindElem 函数存在形参变量，则将节点中元素及属性与形参变量进行比较，内容匹配成功，XML 文档指针指向当前节点。内容匹配不成功，需先查找到节点结束标签，然后再寻找下一个节点。查找到的新节点起始标签为前一节点的同级节点，继续对此节点进行检索。以此循环，直至查找到与形参匹配节点或同级节点遍历结束。

在函数内部设置两个标签统计变量，记录遍历节点的起始标签和结束标签的个数。具体查找过程如下图 5-4 所示：

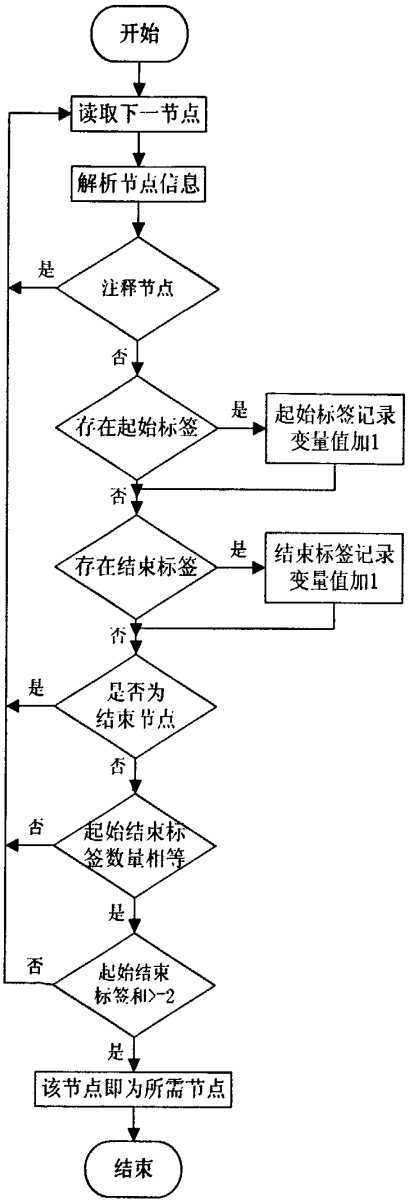


图 5-4 FindElem 函数实现过程

当前节点下一个节点存在四种情况：

1、为注释节点。

2、当前节点的子节点。

3、为同级节点。

4、为当前节点的结束节点。

针对上述四种节点，采用如图 5-4 处理方式进行寻找处理：

1、下一节点为注释节点的情况：

当下一节点为注释节点，形为<?……>时。此时，程序检测到此节点情况，放弃对该节点判断操作，直接读取下一节点。

2、对于下一节点为当前节点的子节点或同级节点的情况。

当下一节点为为当前节点的子节点或同级节点时，此时判断起始标签和结束标签个数。若起始标签个数可技术标签个数相等且其和大于等于 2，则此节点即为所要查找的同级节点。若起始标签个数可技术标签个数不相等或者其和小于 2。继续查找访问下一级节点，直到返回符合要求的节点。

3、对于下一节点为结束节点的情况。

下一节点为结束节点，表示当前节点不存在下一同级节点，返回值为空值，节点指针继续指向当前节点。

采用以上几种方式，可以针对各种不同类型的节点，最终可以正确返回下一同级节点，或下一同级节点不存在，返回空值。

（2）IntoElem 函数。用于在 XML 文档中查找到并指向当前节点的子节点。

函数首先检索到节点起始标签，若在再次查找到节点起始标签之前未检测到节点结束标签，则再次查找到的节点起始标签后节点即为先前节点的子节点。若不存在此种情况，则返回 NULL。

查找方式类似于 FindElem 函数，所不同的是查找节点的判断结束条件为起始标签个数与结束标签个数之差为 1 且其和大于等于 1，则此节点即为所要查找的子节点。若起始标签个数与结束标签个数之差为 1 或者其和小于 1。继续查找访问下一级节点，直到返回符合要求的节点。

（3）LeaveElem 函数。用于在 XML 文档中查找到并指向当前节点的父节点。

函数顺序遍历节点，记录节点起始标签数及节点结束标签数。当满足节点起始标签数比节点结束标签数少一个时，满足条件的节点即为原节点父节点。若不存在此种情况，则返回 NULL。

查找方式类似于 IntoElem 函数，所不同的是查找节点的判断结束条件为结束标签个数与起始标签个数之差为 1 且其和大于等于 1，则此节点即为所要查找的子节点。若结束标签个数与起始标签个数之差为 1 或者其和小于 1。继续查找

访问下一级节点，直到返回符合要求的节点。

（4）GetAttrib 和 SetAttrib 函数。分别用于读取当前节点元素值或属性值和设置当前节点元素值或属性值。

对于 GetAttrib 和 SetAttrib 两个函数进行不同处理：

一、对于 GetAttrib 函数，可以直接读取、返回当前节点的元素及属性值。

二、对于 SetAttrib 函数，因为 SAX 解析方式只是对文档流进行操作，如直接在原有位置修改、重置节点元素值或属性值会发生修改后的内容信息无法完全填充原有位置导致错误信息产生或是覆盖其后节点信息，造成 XML 文档发生结构错误。针对以上问题，采取新建临时中间 XML 文档的方式，即将修改节点前的文档内容写入到临时文档中，再将修改的节点内容写入文档，并将其后节点原样写入中间文档。最后删除原文档，将临时文档命名为原文档，节点修改完成。这样就解决了 SAX 解析无法进行写操作的问题，在实际使用中发现，此方式对于小 XML 文档，修改时间在可以接受的范围之内。不影响系统整体运行。

具体过程如图 5-5 所示：

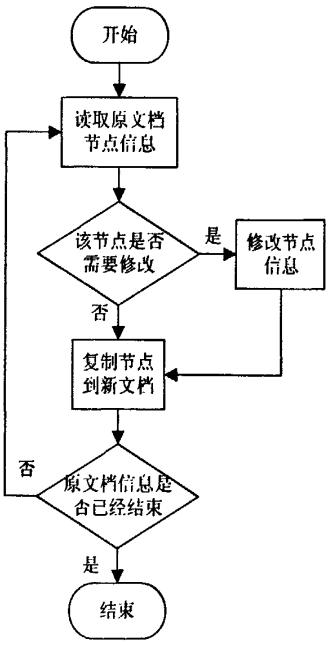


图 5-5 SetAttrib 函数

（5）SavePos 函数。用于保存当前节点在 XML 文档中位置，以供恢复备用。RestorePos 函数则用于 XML 文档指针恢复到形参所定义的节点位置。

指向当前节点位置的指针保存能方便的进行节点的定位，例如在 SetAttrib 函数中能快速保存、恢复当前节点的位置，能大大提高系统的运行效率及操作灵活性。

6 洗衣机控制系统的实现

这一章给出用动态组态系统实现的多路洗衣机洗液分配控制系统。针对多路洗衣机洗液分配控制系统的设备特性讨论优化设计思想。

洗衣机分配器的数据库信息是以页面为单位，保存在 XML 文档中。这样处理数据库信息，可以解决所有的数据信息存放在一个 XML 文档中所带来的信息量过大、检索效率低下等问题。

系统通过主控页面、密码输入页面、流程选择页面、流程编辑页面四个页面的 XML 配置文件，实现其功能的配置。

页面由两大块组成，一块包含页面的名称及页面显示范围的页面定义，另一块是由页面中调用的参数、页面的显示内容和页面中按键的定义所组成的页面属性。

6.1 全局变量的保存

系统中存在一些多个页面都需要调用的变量，将这些变量集中起来放在一个单独的 XML 文件中，这样既能保持变量更新和调用时数据的正确性，也为数据的查找带来了方便。

Global.xml 就是保存这些全局变量信息的独立的 XML 文档。如图 6-1 所示：

```
<?xml version="1.0" encoding="gb2312" ?>
- <MDB>
  + <Attr>
  + <GVAR>
  </MDB>
```

图 6-1 Global.XML 文档根节点

在 Global.XML 文档中，以 MDB 作为根节点。根节点中保存着两个子节点，一个是记录当前 XML 属性的 Attr 节点和一个记录全局独立变量的 GVAR 节点。如图 6-2 所示：

```
<?xml version="1.0" encoding="gb2312" ?>
- <MDB>
  - <Attr>
      <StrArr Name="Globals" Version="1" Revision="0" Date="2009-08-11" />
    </Attr>
  - <GVAR>
      <Bit Name="IOStat" Val="0" />
      <Bit Name="MachStat1" Val="0" />
      <Byte Name="MachOpt1" Val="0" />
      <Bit Name="MachStat2" Val="0" />
      <Byte Name="MachOpt2" Val="1" />
      <Bit Name="MachStat3" Val="0" />
      <Byte Name="MachOpt3" Val="2" />
      <Bit Name="MachStat4" Val="0" />
      <Byte Name="MachOpt4" Val="3" />
    - <StrArr Name="Opt">
        <Str Val="洗桌布" Uri="Spread.xml" />
        <Str Val="洗床单" Uri="Sheet.xml" />
        <Str Val="洗衬衫" Uri="Shirts.xml" />
        <Str Val="洗毛巾" Uri="Towel.xml" />
      </StrArr>
    - <StrArr Name="Process">
        <Str Val="洗衣粉" />
        <Str Val="漂白剂" />
        <Str Val="柔顺剂" />
        <Str Val="消毒液" />
        <Str Val="浆料" />
      </StrArr>
    </GVAR>
  </MDB>
```

图 6-2 Global.XML 文档详细

在 Attr 节点中保存当前 XML 文档的文件名、版本等信息。IO 在 GVAR 节点中保存着所有可供其他 XML 文档调用的标志位及变量信息。

IOStat 用于记录系统 I/O 端口的使用状态。若外部端口在进行检测输入或控制输出时，系统端口为独占状态，其他 I/O 端口此时不能访问系统端口。

MachStat（1、2、3、4）用于保存各机器设备的运行状态。高电位表示当前设备正在运转，反之，则表示设备正处于停机状态。

MachOpt（1、2、3、4）用于保存各机器设备的运行的流程编号。系统不同设备的功能是动态组态的，系统所拥有的功能由 MachOpt 值决定。

StrArr Name 为 Opt 的数组里保存的是 MachOpt 里对应的功能状态，例如 MachOpt1 中保存的值是“0”，则其对应的功能即为洗桌布，在实际运行时，根据两次映射，设备 1 检索洗桌布流程对应的 Tablecloth.XML 文档，参照

Tablecloth.XML 文档，实现具体功能。

StrArr Name 为 Pro 的数组里保存的是流程中的可能出现的步骤名。流程中使用的步骤以数字来代替。当需要具体显示或者步骤实际运行时，通过流程中步骤的数字，在该数组中对应查找到具体的流程名。

6.2 主控页面的实现

首先，在洗衣机初始化时，显示主页面，如图 6-3 所示：

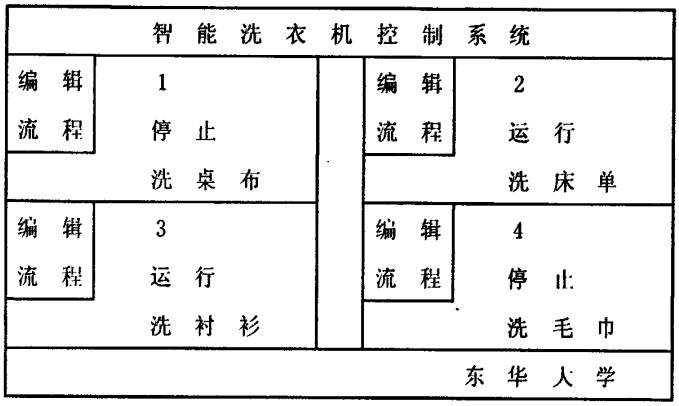


图 6-3 洗衣机初始化页面

页面节点中所用到的变量以<Var>节点进行定义：例如标题的名称、机器号等。这些的内容放在<Var>中，方便数据的管理，并可以很方便的被整个 XML 的其他节点进行调用，提高了系统维护的效率。

页面显示的内容以<Disp>节点及其子节点定义。其中，<Disp>子节点有这样几种定义类型：

表示固定显示内容的文本框定义为<Label>类型：例如“智能洗衣机控制系统”；

表示按键的显示框定义为<Btn>类型。<Btn>的显示信息由包含其显示内容的动态变量决定，例如“编辑”。除此之外，在<Btn>类型的子节点中，将按键功能响应定义为<Func>节点，通过该子节点中的函数名和变量，对按键的操作进行响应。

还有一种表示动态显示内容的文本框定义为<Text>类型：这些文本框的显示内容由变量决定。

通过以上三种不同显示类型的定义，可以对实际使用中各种类型的显示文本进行划分，并以此进行定义即可以满足实际的功能。一旦有新的功能需要添加或修改，只需按照以上格式进行定义，即可实现目的，此时不需对程序源代码做出更改，体现了系统功能动组态的特点。

具体实现过程如下：当单片机获取需要显示初始页面信息时，调用初始页面显示函数。

其运行时显示流程如下介绍。图 6-4 为初始页面 Disp 节点信息：

```
<Label Name="Title" Val="智能洗衣机控制系统" Pos="6,0" />
- <Btn Name="" Val="编辑\a流程" Rect="0,1,3,2">
    <Func Name="Jump" Para="ThePass" />
  </Btn>
  <Label Name="Machine" Val="1" Pos="6,1" />
- <Btn Name="" Src="Stat[gbStat1]" Pos="6,2">
    <Func Name="Compare" Para="Stat[gbStat1], 0" />
    <Func Name="Assign" Para="Stat[gbStat1], 1" />
  </Btn>
  <Edit Name="" Src="Opt[gcOpt1]" Pos="6,3" />
```

图 6-4 洗衣机初始化 Disp 节点信息

在显示过程中，对于不同类型的显示信息采取相同的处理方式，即按照配置文档将显示内容按照指定的显示位置显示出来。

首先，主控模块通过 XML 解析程序 FindElem、IntoElem 函数组合，访问存于 SD 模块之上 TheHome.xml 配置文档。查询到需要显示的 Label Name 为“Title”的节点，通过 GetAttrib 函数的读取 Val 中需要显示的内容信息“智能洗衣机控制系统”放入临时变量，在通过 GetAttrib 函数的读取 Pos 信息“6，0”。“6，0”表示其需要放在第 0 行、第 6 列。申请一个大小为 240（OCMJ8×15D 可以显示 8×15 个汉字信息，故其可显示半角个数为 8×15×2＝240 个）位的显示缓存，将读取的内容信息放入从显示缓存中的第 6 位（每行 30 个半角字符，故在显示缓存中存放起始位置为 0×30+6 = 6）开始缓存中。以此反复循环，直到显示内容输出完毕或需要显示的信息超过显示页面的行数。如果是显示信息超过页面可显示的行数，则放弃显示后续内容，但添加翻页标志，为后续翻页操作做准备。具体解析过程如图 6-5 所示：

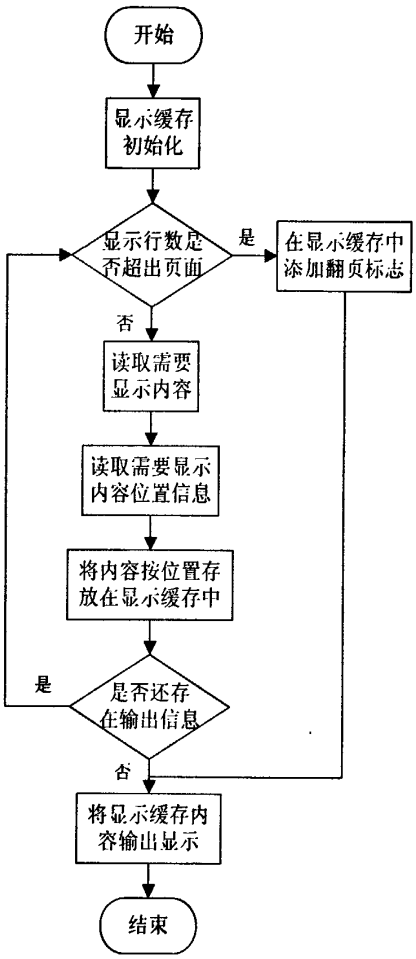


图 6-5 页面显示流程图

如若遇到形如 <Btn Name="" Src="Stat[MachStat1]" Pos="6,2"> 节点，其 Src 属性内容需要从 Global.xml 中读取 MachStat1 信息进行显示，此过程中首先对当前处理的 XML 文档位置运用 SavePos 函数保存当前节点位置。当从 Global.xml 文档读取到 MachStat1 信息之后，通过 RestorePos 函数，恢复处理的节点位置，继续向下处理。

当用户点击显示屏幕，主控模块通过访问液晶寄存器获取到上次点击的具体位置，将位置信息进行单元格映射，对应到 240（8×15×2）个单元格中的某一个，保存该信息，以备后用。XML 解析器读取到该页面所有 Btn 节点位置信息及按钮长度（即显示内容长度），并综合考虑是否对按键进行扩展，最终确定该页面所有的有效按键范围，如果查找到对应按键，执行该按键的子节点功能函数 Func 的操作。具体过程如图 6-6 所示：

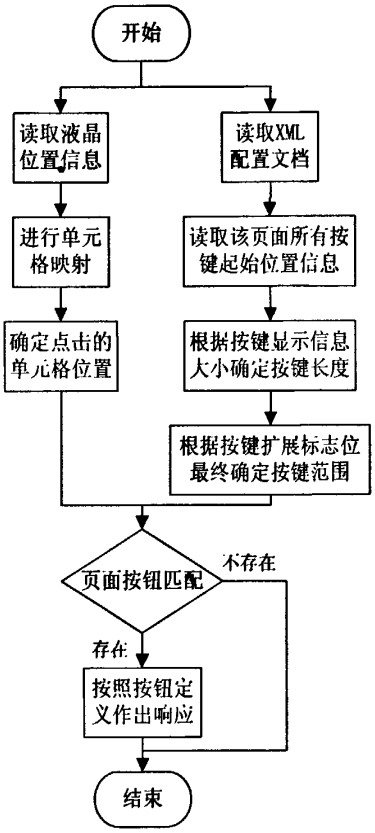


图 6-6 按键处理流程图

如若遇到形如 <Func Name="Assign" Para="Stat[MachStat1], 1" /> 功能子节点时，需要对 MachStat1 变量进行赋值。如上所述，在 Global.xml 文档中查找到 MachStat1 节点，通过 SetAtrrib 函数将 MachStat1 的 Val 值修改为“1”。

同理，其他页面均可以按照此种定义进行系统配置，以实现其各自的目的功能。

当用户点击编辑流程画面，系统跳转到如图 6-7 所示密码输入界面，界面显示及操作内容与主控页面相似，故不在此赘述。

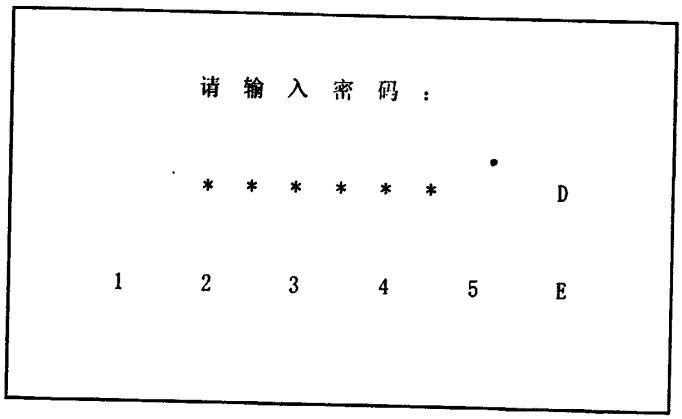


图 6-7 密码输入页面

需要注意的是，系统的管理员拥有系统密码，当管理员输入正确密码时，可以进入流程编辑页面；对于非管理员，输入错误密码或不输入密码只能进入流程查看页面。管理员密码可以动态更新图 6-8 中名为 strPSWSet 的 Val 值。

```
- <MDB Name="The XML for the password picture." Version="1" Revision="0" Date="2009-08-11">
  - <LVAR>
      <Str Name="strPSWSet" Val="1234" />
      <Str Name="strStar" Val="" />
      <Str Name="strPSWIn" Val="" />
    </LVAR>
```

图 6-8 密码输入界面局部变量

当密码输入正确后进入流程编辑页面如图 6-9 所示：

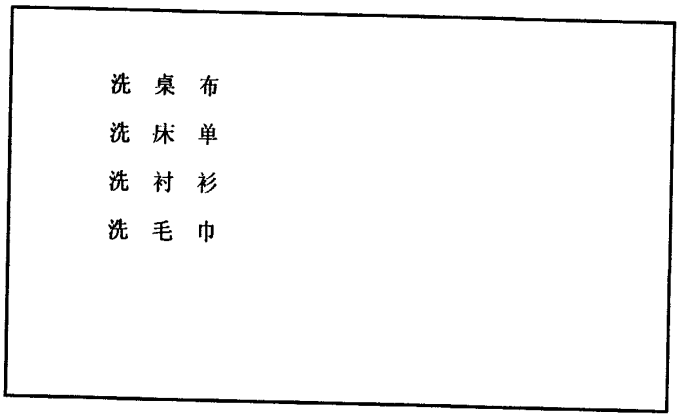


图 6-9 流程显示页面

流程中保存着每一种洗衣方式的最后更新。点击进入可以进行流程修改页面。

6.3 流程编辑页面的实现

以洗桌布流程编辑画面举例说明，如图 6-10：

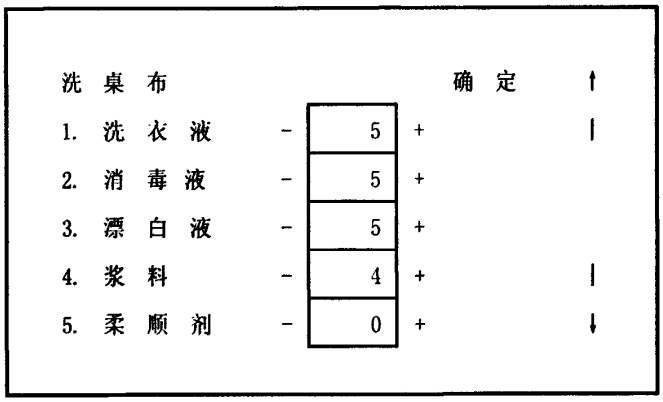


图 6-10 洗桌布流程页面

洗桌布流程的 XML 文档的页面定义中，定义画面名为 Tablecloth。为了显示画面的全部信息，需要 Global.xml 及 Tablecloth.xml 共同实现完成。

图 6-11 为洗衣流程对应的 Tablecloth.XML 文档根节点内容。

```
<?xml version="1.0" encoding="gb2312" ?>
- <MDB>
  + <Attr>
  + <LVAR>
  + <Disp>
  </MDB>
```

图 6-11 Tablecloth.XML 文档根节点

在 Tablecloth.XML 文档中，以 MDB 作为根节点。根节点中保存着三个子节点，一个是记录当前 XML 属性的 Attr 节点、一个记录此 XML 文档内使用的局部变量 LVAR 及表示页面显示属性的 Disp 节点。

Attr 节点属性同 Global.xml 中的 Attr 节点定义，用于保存当前 XML 文档的文件名、版本等信息。

保存局部变量的 LVAR 具体内容如图 6-12 所示：

```
- <LVAR>
    <Byte Name="Drift" Val="0" />
    <Byte Name="ProDispNum" Val="5" />
    <Byte Name="ProTotalNum" Val="20" />
  - <StrArr Name="Tablecloth">
      <Str Val="0" TimeSet="5" />
      <Str Val="1" TimeSet="5" />
      <Str Val="2" TimeSet="5" />
      <Str Val="3" TimeSet="0" />
      <Str Val="4" TimeSet="0" />
      <Str Val="5" TimeSet="0" />
      <Str Val="6" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="8" TimeSet="0" />
      <Str Val="4" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="9" TimeSet="0" />
      <Str Val="5" TimeSet="0" />
      <Str Val="4" TimeSet="0" />
      <Str Val="3" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="8" TimeSet="0" />
      <Str Val="2" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="1" TimeSet="0" />
    </StrArr>
  </LVAR>
```

图 6-12 Tablecloth.XML 文档 LVAR 子节点

在洗桌布流程文件最前端，保存着页面显示的三个关键参数和洗桌布流程中每个步骤的加料时间的数组。

Drift 表示当前显示流程的偏移量。当前画面中，偏移量为 0，故从第一个洗衣步骤开始显示。

ProDispNum 表示当前画面的流程数。

ProTotalNum 表示的是流程步骤中总的个数。Val 值是可以是由程序动态修改，也可以有人工手动修改。

在名为 Tablecloth 的字符串，保存着洗桌布流程中的步骤及步骤时间。Val 值保存着表示步骤名称的数字值，通过 Global.xml 中名为 Pro 数组的对应，能得到具体的流程名称。TimeSet 值保存着每一个步骤所需的运行时间。这两项值均可人为手工设定或通过人机界面进行修改。

控制系统内容的显示信息的部分如图 6-13 所示：

```
- <Disp>
    <Rect Pos="0,0,29,7" />
    <Label Name="" Val="洗桌布" Pos="2,1" />
  - <Column Name="ThePro1">
      <Text Src="Drift+1" Pos="2,2" />
      <Label Val="." Pos="3,2" />
    - <Btn Src="Process[Tablecloth[0+Drift].Val]" Pos="4,2">
        <Func Name="Add" Para="Tablecloth[0+Drift].Val,1" />
        <Func Name="Complement" Para="Tablecloth[0+Drift].Val,10" />
        <Func Name="RefreshDisp" />
      </Btn>
      <Text Src="Tablecloth[0+Drift].TimeSet" Pos="17,2" />
    - <Btn Name="Add" Val="+" Pos="12,2">
        <Func Name="Add" Para="Tablecloth[0+Drift].TimeSet,1" />
        <Func Name="RefreshDisp" />
      </Btn>
    - <Btn Name="Minus" Val="-" Pos="19,2">
        <Func Name="Minus" Para="Tablecloth[0+Drift].TimeSet,1" />
        <Func Name="RefreshDisp" />
      </Btn>
    </Column>
```

图 6-13 Tablecloth.XML 文档 Disp 子节点部分信息

以洗衣粉的显示设定为例，其他步骤类似。

Rect Pos 的范围表示系统显示的范围，当前页面的显示范围是“0，0，29，7”，表示为整屏显示。

Label 表示“洗桌布”流程名的显示，显示位置为第 1 行第 2 列的位置上。

Text 表示序号“1”的显示，以偏移量加上当前行数计算，显示在第 2 行第 2 列的位置上。“.”显示在第 2 行第 3 列。

洗衣粉的名字为 StrArr 名为 Process 的数列中第 0 个数组的 Val 值（0 + Drift = 0），其值为 0。则在 Global.xml 中 StrArr 名为 Process 的数列第 0 项对应“洗衣粉”。其显示位置在第 2 行第 4 列。该位置是一个按钮，点击时，将在 Tablecloth[0 + Drift].Val 的数组的里数加 1 对 10 取余并保存。立即刷新显示为 Global.xml 中 StrArr 名为 Process 的数列第 Tablecloth[0 + Drift].Val 项。

洗衣粉的添加时间显示为 StrArr 名为 Process 的数列中第 0 个数组的 TimeSet 值，其值为 5。显示位置为显示在第 2 行第 17 列。

“+”显示在第 2 行第 12 列，点击该按钮时 Tablecloth[0 + Drift].TimeSet 值加 1 保存，并立即刷新显示。

“-”显示在第 2 行第 17 列，点击该按钮时 Tablecloth[0 + Drift].TimeSet 值减 1 保存，并立即刷新显示。

通过点击按键可以修改洗桌布每一步骤的所需加料的内容及时间，达到更新洗衣流程配置的目的。

6.4 洗衣工作流程的实现

编辑页面完成后，退出到主控页面。点击停止按钮，开始运行洗衣流程，此时停止按钮也转为运行状态。洗衣运行时，根据相应的 XML 配置文档分配各自的控制端口，现以洗桌布为例，解释具体步骤。

根据 Tablecloth.xml 的 LVAR 配置文件，如图 6-14 所示：

```
- <LVAR>
    <Byte Name="Drift" Val="0" />
    <Byte Name="ProDispNum" Val="5" />
    <Byte Name="ProTotalNum" Val="20" />
  - <StrArr Name="Tablecloth">
      <Str Val="0" TimeSet="5" />
      <Str Val="1" TimeSet="5" />
      <Str Val="2" TimeSet="5" />
      <Str Val="3" TimeSet="0" />
      <Str Val="4" TimeSet="0" />
      <Str Val="5" TimeSet="0" />
      <Str Val="6" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="8" TimeSet="0" />
      <Str Val="4" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="9" TimeSet="0" />
      <Str Val="5" TimeSet="0" />
      <Str Val="4" TimeSet="0" />
      <Str Val="3" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="8" TimeSet="0" />
      <Str Val="2" TimeSet="0" />
      <Str Val="7" TimeSet="0" />
      <Str Val="1" TimeSet="0" />
    </StrArr>
  </LVAR>
```

图 6-14 Tablecloth.XML 文档的 LVAR 节点配置信息

如图 6-14 所示，系统开始自动运行时，首先访问洗桌布对应洗衣机是否开机，液料是否准备完毕。单片机控制系统逐个读取 StrArr Name 为 tablecloth 的

数组中的信息。先对位于 0 号步骤的端口进行 5 秒的输出控制，输出结束后，洗衣机洗衣，等待下一步骤的请求信号，再对 1 号步骤的端口进行操作，以此循环，直至步骤时间为 0 结束。这里的 0 好步骤是对应实际上的 0 号加料设备，可以对此 Val 进行修改以动态分配不同的加料端口。

故通过以上显示和按键处理方式和自动运行模式即可实现系统的控制功能。

6.5 针对系统设备特性进行优化的设计

洗衣机系统可以控制多台洗衣机的洗衣流程，加料方法采用总线复用模式，如图 6-15 所示：

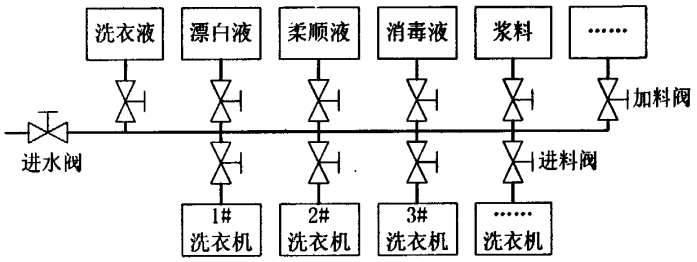


图 6-15 多设备系统结构图

系统要求在洗涤过程中，每次只能给一台洗衣机添加辅料。加料的过程是先打开进水阀，然后打开一个加料阀和一个进料阀，一个加料过程结束后关闭所有阀门，再进行下一个加料过程。每一个过程的洗涤时间由洗衣量和衣服的干净程度决定，一般为 3～5 分钟。如果发生多台洗衣机同时请求加料，若控制模块不能正确优化工作时序，则会在进水总线上同时输出不同的辅料，发生辅料添加混乱的情况。

针对上述问题，在 XML 文件里分别增加加料阀和进料阀 I/O 端口独占标志位。当控制模块要对某个 I/O 端口操作时，首先查询端口独占标志位状态。当表示为未占用状态，系统可以对当前的 I/O 端口操作，并置位端口独占标志位为占用状态。操作结束，重新置位端口独占标志位为非占用状态，交出占用权。当独占标志位状态表示端口被占用时，则需要等待占用的 I/O 端口操作结束，才可以进行新的 I/O 端口操作。在端口占用时，其他 I/O 端口操作请求信号按照一定的顺序保存起来。当占用的端口交出控制权时，系统按照命令保存顺序，优先处理

最先请求信号。这样保证每次只有一个 I/O 端口操作一台设备，且保证每条控制命令都能做出正确响应。

独占标志位的使用可能会在对端口进行输出控制时产生时间堵塞的问题，解决的方法是增大洗衣机每个加料步骤中等待添加辅料的时间（设为单独添加辅料时间的 N 倍，N 为机器台数），保证辅料能在等待时间内完成添加。

控制系统为低成本系统，扩展程度有限制。如果一味扩展设备，系统检测输入量大幅增多，系统辅料添加的等待时间就会过大，且占用总管时间过长，系统工作效率会大幅下降，系统扩展意义不大。如有需要大幅扩展设备的情况下，可以另行配置一套工作控制设备，以解决该问题。

7 结束语

7.1 结论

7.1.1 研究内容总结

本文设计了为用户提供更新、修改灵活方便的单片机控制系统。系统常用的函数作为系统的应用程序保存在单片机的固定程序中，系统的页面信息、输入输出端口信息、系统的功能、流程等数据库信息以 XML 配置文件保存。XML 配置文件可以由用户根据组态要求，由 PC 机直接访问和修改，下载到系统的 SD 卡，并通过 XML 解析模块进行解析和调用系统的应用程序，系统功能改变无需修改单片机程序，实现了系统动态组态的功能。一个洗衣机应用实例给出了动态组态系统的实现方法，表明本文提出的设计思想有实际应用价值。

本文以传统单片机控制系统和现有大型动态组态控制系统为参考对象提出一种基于单片机系统的动态组态控制系统结构，并提出相应的硬件结构和软件结构，重点介绍了主控模块、人机界面模块、SD 卡模块、I/O 端口模块及 XML 解析程序的实现。

首先对目前传统单片机控制系统和现有大型动态组态控制系统相对于小型动态组态控制系统的不适应性，提出了一种新型的小型动态组态控制系统。

其次通过对系统硬件构架特点的分析，分别阐述实现所需要的各硬件部分的组件。并根据个硬件部分功能，提出相应的软件模块实际功能。

进而介绍了系统软件的核心部分：XML 配置文件的解析程序。针对 MCU 内存空间小和处理速度低的特点，提出了一种新型的基于 SAX 思想的解析和改写程序设计思路和实现过程。

最后以实际的洗衣机控制系统验证实现了该小型动态组态控制系统的有效性。

7.1.2 研究创新点

1．研究的课题综合分析了现有单片机控制系统和大型组态控制系统，针对

小型控制系统，提出了一种动态组态的控制系统，提升了小型单片机控制系统应用的灵活性。

2．针对单片机处理速度慢、内存空间小的缺陷，在单片机系统上提出了一种新型的 XML 解析器，该解析器能够很好的在单片机系统平台上使用，具有广泛性。

3．提出的多 MCU 控制系统的优化方法，对于其他控制系统也具有一定的参考价值。

7.2 展望

本文在小型动态组态单片机控制系统的研究中，并取得了一定的研究成果，但鉴于系统是初步构想，在以下几个方面有待进一步改进和完善的余地：

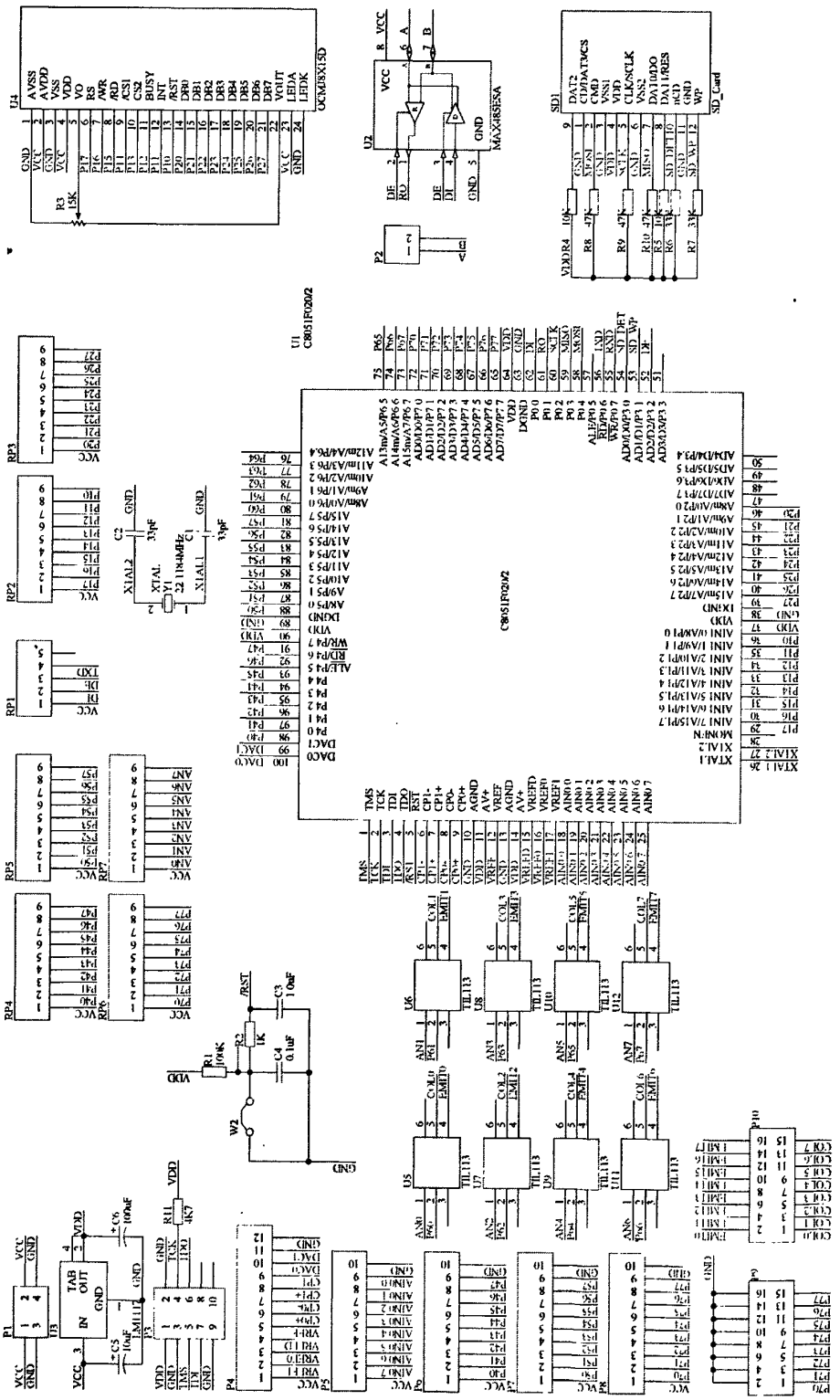
1、优化 XML 解析器算法，使之能够提高运行效率，解决系统的瓶颈问题。

2、对系统模块功能进行丰富，例如添加以太网络支持模块，则可以采用 PC 和单片机双上位机支持，并可控制基于 TCP/IP 协议的设备。使动态组态的单片机控制系统能够有更多更丰富的组态模块和功能。

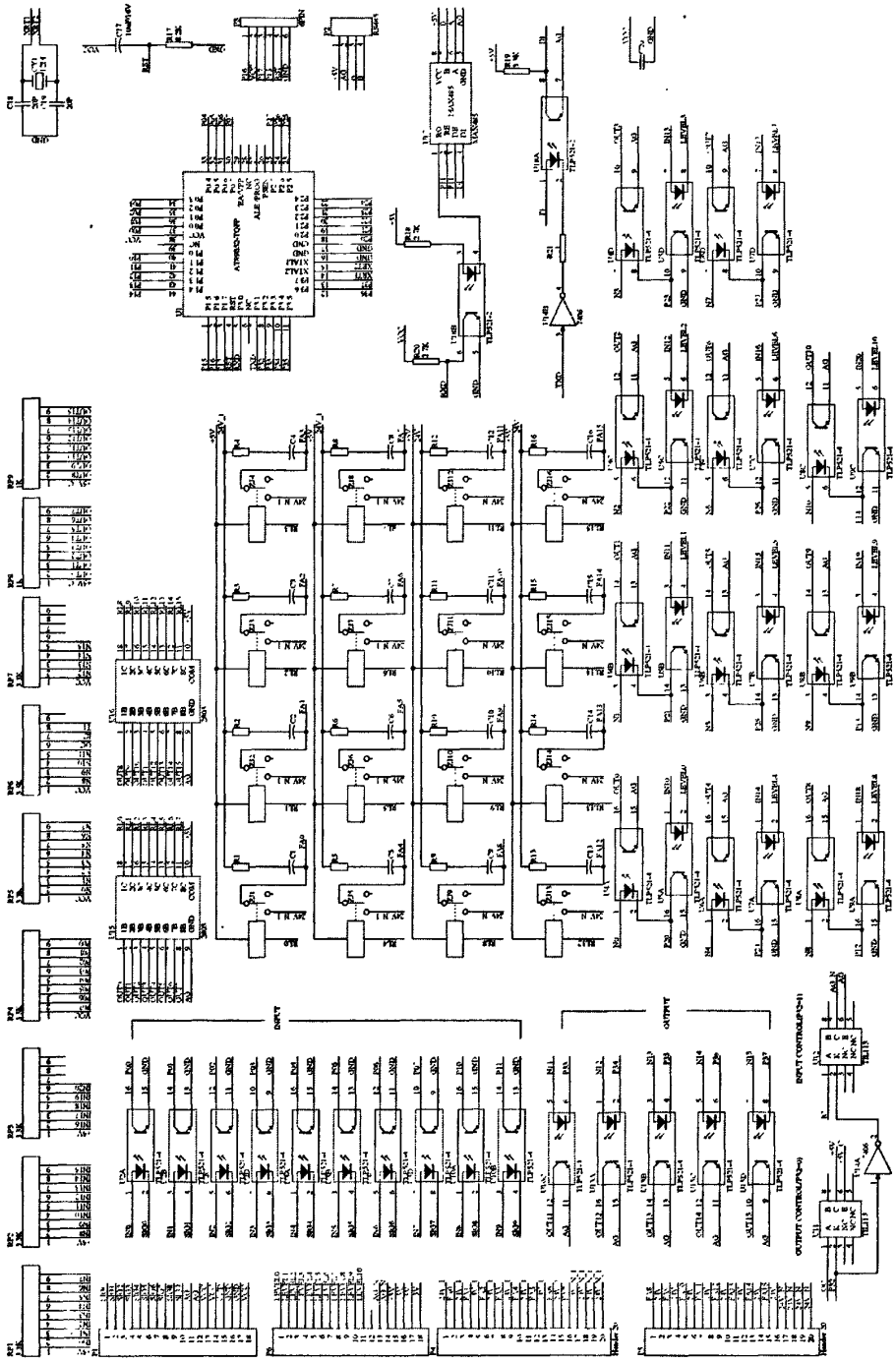
附录 1 OCMJ8X15D 引脚说明

引脚	名称	方向	说明
1	AVSS	--	模拟电源地（0V）
2	AVDD	--	模拟电源正（+5V）
3	VSS	--	数字电源地（0V）
4	VDD	--	数字电源正（+5V）
5	VO	--	LCD 驱动电压输入端
6	RS	I	H：存取 DDRAM；L：存取缓存器。
7	R/W(/WR)	I	6800 系列：读/写脚(R/W)，H：读，L：写。 8080 系列：写入脚(/WR)，低有效。
8	EN(/RD)	I	6800 系列：使能脚(EN)，高有效。 8080 系列：读入脚(/RD)，低有效。
9	/CS1	I	当/CS1 为低和 CS2 为高时，模块处于致能， 可接受指令，反之不可接收指令。
10	CS2	I	当/CS1 为低和 CS2 为高时，模块处于致能， 可接受指令，反之不可接收指令。
11	BUSY	O	用以回应模块内部的执行使用状况，可设成高或低电平触发。
12	INT	O	用以回应模块内部的中断状况，可设成高或低电平触发。
13	/RST	I	复位信号，低有效。
14	DB0	I/O	数据 0
15	DB1	I/O	数据 1
16	DB2	I/O	数据 2
17	DB3	I/O	数据 3
18	DB4	I/O	数据 4
19	DB5	I/O	数据 5
20	DB6	I/O	数据 6
21	DB7	I/O	数据 7
22	VEE	--	LCD 驱动电源
23	LEDA	--	背光源正（+5V）
24	LEDK	--	背光源负（0V）

附录 2 主控板电路原理图



附录 3 扩展板电路原理图



参考文献

[1] 李利. AT89C2051单片机在双组分纺纱自控系统中的应用[J]. 上海毛麻科技，2005(04)：18-20.

[2] 王沁敏，鲁植雄. 基于单片机的拖拉机电液悬挂控制系统的设计[J]. 机电一体化，2008(04):82-85.

[3] 张杰. MCS-51单片机在化工生产中的实践[J]. 宁夏石油化工，2003(04)：21-22.

[4] 张建辉. 基于单片机的家电远程控制系统设计[J]. 苏州科技学院学报(工程技术版)，2006(02)：79-82.

[5] 胡新. 基于单片机的嵌入式系统研究与开发[J]. 科技资讯，2008(09)：123.

[6] 杨虎，唐兴基，金祖升，等. 基于AT89S52单片机的测速雷达伺服控制系统设计[J]. 电子测量技术，2009(11)：89-91，128

[7] 杨恒玲，胡燕瑜. 基于单片机的水温模糊控制系统[J]. 科技创新导报，2009(32)：43-44

[8] 汤亮，席泽敏，魏钟记，等. 一种基于单片机的雷达发射机功放电源控制系统设计[J]. 舰船电子工程，2009(11)：103-105

[9] Groover M. P. Automation，production systems，and computer-integrated manufacturing [M]. Prentice Hall，2007.

[10] 马国华. 监控组态软件及其应用[M]. 清华大学出版社，2001.

[11] 郝迎吉，马德平. 一种基于单片机的组态王温度监控系统[J]. 西安科技大学学报，2005(02)：201-203，223.

[12] 但斌斌，马乾，时宝祥，等. 基于In Touch的监控系统的设计与应用[J]. 可编程控制器与工厂自动化，2009(02)：73-74，49.

[13] 梁伟栋，郭浩. MCGS组态软件设计及其应用[J]. 广东自动化与信息工程，2005(01)：33-35.

[14] 刘教瑜，张兰. 组态王在监控系统中的研究与实现[J]. 工业控制计算机，2008,21(12)：1-3.

[15] 祝常红. 单片机与MCGS组态软件在全自动中央供暖系统中的应用[J]. 电

子技术应用，2006(09)：85-87.

[16] 潘琰，朱善安．小型集散控制系统组态监控软件及实现[J]．机电工程，2002(04)：40-42

[17] 侯亮，徐燕申，唐任仲，等．面向广义模块化设计的产品族规划方法研究[J]．中国机械工程，2003(07)：52-65.

[18] 陶志东．基于C8051F系列单片机字符显示器的开发[D]．硕士学位论文，华中科技大学，2006：16

[19] 郭兵．SoC 技术原理与应用[M]．清华大学出版社，2006.

[20] 叶丽娜．C8051F与80C51系列单片机的不同初始化[J]．国外电子元器件，2004(03)：9-12

[21] 孙立香，赵不贿，刘星桥．C8051F020与80C51单片机的异同点[J]．国外电子元器件，2007(05)：31-35

[22] Semiconductor National Corporation．LM1117/LM1117I 800mA Low-Dropout Linear Regulator[Z]．2006.

[23] 潘琢金．C8051F020/1/2/3 混合信号 ISP FLASH 微控制器数据手册[M]．2005.

[24] 吴琦，廖启征，魏世民．基于RS232接口的电机状态上位机监控界面设计[J]．机电产品开发与创新，2009(06)：129-130，133

[25] 孙渊，王仕成，闵海波，等．RS422高速串行通信在AT91RM9200上的实现[J]．弹箭与制导学报，2008(01)：329-332

[26] 耿立中，王鹏，马骋，等．RS485高速数据传输协议的设计与实现[J]．清华大学学报(自然科学版)网络.预览，2008(08)：1311-1314

[27] National Instruments Corporation．RS-232，RS-422和 RS-485 串口通讯接口的快速比较[EB/OL]．http：//digital.ni.com/public.nsf/allkb/C26D1B994F540A938625713B0022D9DC.

[28] Products Maxim Integrated．低功耗、限摆率、RS-485/RS-422收发器[Z]．2003.

[29] 金鹏电子有限公司．D系列中文液晶显示模块使用说明书[Z].

[30] 程兆贤，戴宇杰，张小兴，等．RFID中EEPROM时序及控制电路设计[J]．微纳电子技术，2008(11)：677-680

[31] 刘春景，沈武群．CF卡在数控机床上的应用[J]．林业机械与木工设备，2008(11)：54-55

[32] 赵子恺，李青，吴秀山．基于SD卡的冷链温度记录仪[J]．仪表技术与传感器，2008(10)：25-27

[33] Specifications SDMC Part 1[J]．Physical Layer Specification Version, 2000,1.

[34] 王保成. Windows操作系统中的文件系统[J]．农业网络信息，2007(07)：135-137

[35] 郝伟，李敬兆．基于uC/FS的大容量微存储FAT32格式的实现与应用[J]．电脑知识与技术，2006(32)：113，185

[36] 洪岳炜，王百鸣，谢超英．一种易于移植和使用的文件系统FatFs Module[J]．单片机与嵌入式系统应用，2008(05)：29-31

[37] ChaN．FarFs[EB/OL]．http://elm-chan.org/fsw/ff/00index_e.html

[38] 邵敏，李力鸿，郑震坤．XML 编程实践[M]．北京：清华大学出版社，2002.

[39] 刘芳，肖铁军．XML应用的基石：XML解析技术[J]．计算机工程与设计，2005(10)：2823-2824，2839

[40] 李喆，严春莹，马琳．XML 高级编程[M]．2001.

[41] 朱前飞，高芒．XML解析技术研究[J]．电脑开发与应用，2004(11)：26-28

攻读硕士学位期间发表的学术论文

1. 连翔宇，唐莉萍，赵祖云．动态组态单片机控制系统的研究[J]．东华大学学报，2010,36(5)．（已录用）

致 谢

两年半的研究生活很快就要结束了，通过这几年的学习研究，自己在理论和实践方面都有了一定程度的提高。在此感谢我的导师唐莉萍副教授在多年的学习研究中给予我悉心的指导，她严肃认真的科学态度、严谨求实的工作作风以及孜孜不倦的育人方式让我受益匪浅。

在我攻读硕士学位期间，还得到了实验室多位老师的精心指导和无私帮助，在这里一并向他们表达我最诚挚的谢意和敬意！感谢实验室的同学们：王宇星、张营建、陆宇霆、杨鹏。在两年多的合作和学习交流中，我们积累了深厚的感情，各位同学在学习上的帮助以及在团队合作时从他们学习到的精神让我感受良多。感谢师妹赵祖云在课题研究和论文撰写过程中给予我的关心和帮助。

同时要感谢我的家人及我女朋友张玮和她的家人对我的支持，感谢他们一直默默的关心、照顾我，他们在学习上对我的督促和鼓励是我研究学习的动力，使我终于得以顺利地完成学业，我对他们的感谢难以言表。

最后由衷地感谢在百忙之中评阅本论文和参加答辩的各位专家、教授。