中文摘要

本文旨在设计一个以教学为目的的机器人开发平台。该开发平台由三个子平台构成，即机械平台、硬件平台和软件平台。机械平台中包含了各种机械部件，由它们搭建机器人的躯干、手脚等外围的框架。硬件平台是围绕主控芯片 MC9S08GB60 展开设计的硬件电路板，其中包含了机器人各种控制机构的硬件电路。主控芯片中驻留了自主开发的用于配合 PC 方软件平台下载用户程序的监控程序以及控制机器人动作的功能性模块。机器人控制程序由用户根据 PC 方软件平台提供的开发工具自行“定制开发设计”，并“下载”到硬件平台的 MCU 中执行。软件平台提供了图形化设计界面和类 C 界面两种设计方式，用户可以根据自身特点选择相应的程序设计方式。该教育机器人基础开发平台主要针对中小学课外科技活动开发，也可供高校学生进行机器人设计、参加机器人设计竞赛之用。文章阐述了整个系统的设计背景、设计思想、软硬件实现方法，并对其中的技术要点进行深入的分析。

关键词：教育机器人基础开发平台，MC9S08GB60，图形化设计界面

作　　者：蒋建武
指导老师：王宜怀

ABSTRACT

This paper focuses on designing a platform for robot development with educational purpose. The platform is composed of three separate sub-platforms, namely: machinery platform, hardware platform and software platform. The machinery platform includes various machinery parts, which are used to organize robots ' peripheral frameworks, including bodies, arms, legs, etc. The hardware platform is a circuit board designed around the host MCU - MC9S08GB60 and contains hardware circuits for kinds of robot-controlling unit. MCU is the kernel of the hardware platform, with kinds of routines residing in it. These routines includes independently-developed monitor program which can be accessed by PC software to download user program to MCU and functional routines used to control the actions of robot. The user can customize their own robot-control program and download to MCU by using the developing tools presented by the software platform. The software platform provides two development styles: GUI –based style and C-similar style. The user could select the appropriate style according to his/her preference. The fundamental platform of robot development for education is primarily used in elhi extracurricular technological activity. At the same time, the platform can be used to design robot and attend various robot competition. This paper exposed the design background, design conception and implement approach of software and hardware. Beside that this paper also provides in-depth analysis on technological main point.

Keywords: The fundamental platform of robot development for education, MC9S08GB60, GUI

Written by Jiang Jian Wu

Supervised by Wang Yihuai

目 录

第一章 绪论

机器人学科是一门交叉性极强的综合性学科，它涉及到了机械制造、自动控制、传感器技术、计算机软件技术、计算机硬件技术等许多学科。随着机器人技术的不断发展，它在社会生产的方方面面都得到了广泛的应用[1]。机器人应用中遇到的各种问题对机器人技术提出了新的要求，反过来又促进了机器人技术的进一步发展[2]。如此反复，使得机器人技术在其诞生后短短的几十年中得到了迅猛的发展，特别是在计算机广泛普及以后，机器人也逐步的由工业生产，走向了人们的生活。如此广泛的应用使得提高全体国民对机器人的了解显得尤为重要，普及机器人教育势在必行。在这种背景之下，机器人教育的概念应运而生，推进机器人教育发展的教育机器人基础开发平台也就随之而产生。

1.1 基本概念及其研究背景

1.1.1 教育机器人的概念

虽然机器人教育随着机器人的应用已经发展了好多年，在此过程中也出现了各种各样的教育机器人，但是对于教育机器人并没有一个明确的定义。作者根据自身对教育机器人的研究和理解给出如下定义：教育机器人就是结合教育学和机器人学原理，以教学为最终目的而制造的机器人，它主要用于辅助讲解机器人的工作原理及机器人学相关学科的基本原理[3]。

教育机器人以教育为第一目的，因而它与工业上使用的机器人有很大区别[4]。与机器人的本身的技术价值相比，教育机器人的教育价值更为重要。因而在设计教育机器人时要以教育理念为指导，以方便教学为目的进行开发[5]。

1.1.2 研究的意义

我国延续了几千年传统的教学模式都是强调多动脑少动手，过分的强调向学生灌输知识，教会学生某种知识，然而这种教学方式最大的弊端是使得教学与实践脱节，难以教会学生如何自己去发现问题，思考问题，解决问题，可以说这是一种“授之以鱼”的方法。为了改变这种教学模式，我国推行了学生素质教育改革，这项改革中强调在传授学生知识的同时，教会学生解决问题处理问题的方法，真正的“授之以渔”。

机器人教育强调学生动手能力的培养，在实际操作的过程中注重动手与动脑相结合，促进学生的全面发展[6]。这一点恰好符合了素质教育的初衷，因而国家的相关部门已经把“简单机器人的制作”纳入到了中小学信息技术教学内容之中[7]，同时在普通高等院校招收保送生时把在全国性的机器人大赛中获得一二等奖的学生也纳入其中[8]。由此可见国家已经意识到了机器人教育的重要性。

实施机器人教学必须有一个操作的平台，因为对于中小学学生来说，要求他们一切从零开始设计机器人是不现实的，而且机器人教育的最终目的也不是要学生自己去设计机器人。因而教育机器人基础开发平台成为了普及机器人教育的一个非常重要地环节，设计一个方便而实用的教育机器人基础开发平台就显得尤为重要。

1.1.3 针对对象

从加强全民对机器人的认识角度来讲，机器人教育的对象应该是全体国民。但是从目前国内的发展状况来看，此项教育的最适宜的方式是首先在大、中、小学中开展，然后再向全民普及。因而在目前设计教育机器人基础开发平台时，主要分为两个层次，即分为中小学机器人普及教育和大学机器人研究教育。在中小学中开展时，主要是减少对机器人的神秘感，寓教于乐，培养学生对机器人的兴趣，在此过程中教会学生一些机器人相关学科内容的最基本原理；在大学中开展机器人教学时，则要培养学生利用机器人来解决问题的能力，以便其在以后的工作中能真正的把机器人应用到实际问题的解决中去[9]。因而对于两个不同层次学生，设计基础开发平台时所使用的设计思路也是不同的，前一种为教学型，后一种为研究型。本课题设计的机器人开发平台主要是面向中小学学生的教学型开发平台。

1.2 教育机器人的发展

1.2.1 产生背景

机器人的历史并不遥远，它是在二战以后才发展起来的一项新技术。1959 年美国人英格伯格和德沃尔制造出世界上第一台工业机器人，更准确的说它应该叫做机器手臂。但是机器人在美国发展并不如日本快，二战后日本由于劳动力缺乏，因而大力发展工业机器人来解决劳动力问题，因此日本也号称为“机器人王国”。随着机器人在工业上的广泛应用，如何加强工人对机器人了解从而提高他们对机器人的控制能力

就成为一个显著的问题，机器人教育也就随之而产生，专门用于教学的教育机器人也就出现了。

1.2.2 发展历史

国外教育机器人的开展较早，早在上世纪六七十年代日本、美国、英国等西方发达国家已经相继在本国大学开展了机器人的研究，到了七八十年代在他们国内的中小学中也进行了简单的机器人教学，在此过程中也推出了各自的教育机器人基础开发平台。我国的机器人研究在七八十年代就开展了，在我国的“七五”计划，“863”计划中均有相关的内容。而针对中小学的机器人教学起步较晚，直到上世纪九十年代的中后期才得到了初步的发展，直到目前发展仍然不是很完善。

1.2.3 现状

随着国家对机器人教育越来越重视，各地的重点中小学中均开展了机器人兴趣小组活动，有条件的地方甚至已经开始在学生中全面开展机器人教育。同时每年由国家相关部委组织的面对大中小学生的机器人比赛也进一步促进了机器人教育的开展，比如教育部主办的“全国中小学电脑制作活动”[10]，中国科协主办地“中国青少年电脑机器人竞赛”等。与此同时对于教育机器人基础开发平台的研究也得到了蓬勃的发展，国内已经有许多公司推出了自己的产品，例如上海广茂达公司能力风暴机器人，中鸣仿生机器人，北京交大阳光公司的 Sunny618 机器人[11]等。

1.2.4 前景

虽然目前市场上已经出现了各种各样的教育机器人基础开发平台，但是他们做的并不完善，在实际应用中仍然有很多问题，并不完全适合中小学生使用，比如机器人编程界面问题。因而教育机器人基础开发平台的开发 还有很多值得研究和改进的地方，做好这方面的工作对于国内机器人教育的发展有一定的促进作用。

1.3 教育机器人基础开发平台的设计思想

如图 1-1 所示为教育机器人基础开发平台结构图，整个平台命名为 SD-HCS08

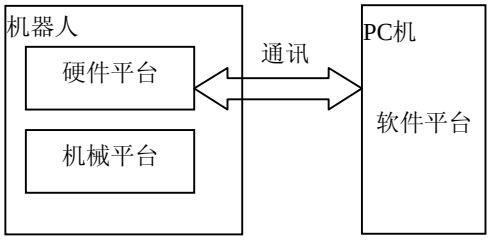


图1-1 教育机器人基础开发平台结构图

型教育机器人基础开发平台(简称 SD-HCS08-Robot 开发平台)。其中主要包括三个组成部分：软件平台、机械平台和硬件平台。软件平台设计为图形化设计界面，用户根据实际情况选择相应的控制部件设计机器人的执行程序，不需要直接编写程序代码，代码由平台软件直接生成，最终程序编译后固化到硬件平台中。机械平台包含用户搭建机器人所使用的机械部件，如行走电机，机械框架等。硬件平台由平台设计者开发，它是一块控制主板，控制机器人的各个机械部件的运动。在此三个平台上，用户就可以按照自己的需要，任意的组装机器人，设计执行程序，最终由机器人根据外部环境的变化完成各种任务。

1.4 毕业设计工作及论文结构

1.4.1 毕业设计工作

(1) 选题

作者根据对目前教育机器人发展的现状以及教育部相关文件精神的分析，认为教育机器人开发平台的研究很有实用价值。同时结合作者对嵌入式软、硬件设计的经验，分析了完成教育机器人开发平台设计的可行性，决定选用“教育机器人基础开发平台设计”为毕业设计课题。

(2) 总体设计

对软硬件设计的具体内容进行分析，确定软、硬件平台的设计方案。

(3) 硬件选型和独立元件实验

根据确定的硬件方案选择适当的芯片元件和电子元件，并做相关的实验，最终确定可行的硬件方案。

(4) 电路板设计、焊接和测试

完成电路板原理图和电路图的设计，并完成最终的布板、焊接以及硬件测试。

(5) 软件设计

根据软件设计方案设计 PC 方和 MCU 方软件。

(6) 综合调试

软硬件联合调试，最终完成开发平台的设计。

(7) 论文

总结毕业设计的过程，完成最终的毕业论文。

1.4.2 论文结构

全文共分为七章，具体介绍如下：

第一章介绍了教育机器人基础开发平台的的设计背景，发展历史，以及SD-HCS08-Robot开发平台的设计思路，最后介绍了设计过程和最终的论文结构。

第二章描述了整个软硬件设计的基本方案。

第三章详细讲述了系统硬件设计过程，具体介绍了硬件选型的原则，所选主要元件的特性，最终电路板的设计过程和测试方法。

第四章描述了主控芯片MC9S08GB60写入器的设计，其中详细阐述了利用背景调试模块(BKGD)设计写入器的思路。

第五章描述了MCU方软件的设计，其中介绍了监控程序的功能，驻留子程序设计，以及用户程序的设计。

第六章讲述了PC方软件的具体实现，其中包括图形化文件的存储结构，图形化文件向嵌入式C语言的转化，以及嵌入式C语言的编译和下载。

第七章对全文进行总结，提出了开发平台设计的后继工作。

第二章　设计方案

基于对目前国内的一些教育机器人基础开发平台的 分析研究，吸取了他们的优点，弥补了其中存在的不足，我们研制了一款新的 SD-HCS08-Robot 教育机器人基础开发平台，以下将给出这款新的开发平台的设计方案。

2.1　目前教育机器人基础开发平台的特点分析

在设计之初对目前市场上现有的部分教育机器人基础开发平台做了一番调研，对它们的各种性能作了分析比较。这些平台的设计均以一个单片机作为主控芯片，所有的开发围绕这个主控芯片展开[12]，底层尽量发挥主控芯片各个模块的功能，上层软件通过串行口与底层硬件系统实现通信。上层软件通过使用嵌入式的 C 语言进行开发，编译后下载到底层主控芯片中实现控制[13]。这样就实现了教育机器人基础开发平台的基本功能，可以为学生提供基本的实验开发环境。但比较分析后发现仍有如下两大局限性：

(1) 编程语言仍局限于高级语言。

在多数的开发平台上使用的编程语言局限在高级语言上，一般为 C 语言。个别虽然推出了图形化的编程界面，但是使用时发现图形化的界面只是作为参考形成流程图，最后仍然要用高级语言来完成最终的编程。这种高级语言的编程风格对于高年级的中学生来说或许还可以接受，但如果放在小学中肯定是不适合的，他们的理解能力可能远没有达到使用高级语言来编程的程度。因而这就限制了这些平台在中小学中的推广。将面向中小学教育机器人开发平台编程界面设计为图形化界面已逐步成为了一种共识，目前已有部分产品能初步达到此要求。

(2) 硬件平台完全封装。

很多产品可能是出于对硬件平台保护的目的，将硬件平台进行完全的封装，用户使用时不用再搭建这个平台，只要在外面加一两个传感器就可以了，这样的设计可以说有悖于开发平台的设计 初衷。开发这个教育机器人平台的目的 就是要让学生多动手，发挥自己的奇思妙想来设计一个有独立风格的机器人，而如果把所有硬件内容都固化了就限制了学生的想象空间，这不利于学生学习，也不符合以教育为目的的设计

宗旨。

2.2 SD-HCS08-Robot 开发平台的基本思路

通过对现有的各个平台的分析，设计的思路仍然沿用围绕主控芯片展开设计的思想，将 SD-HCS08-Robot 开发平台分为三部分来完成，即机械平台，硬件平台，软件平台。

2.2.1 机械平台

机械平台用于搭建机器人的机械框架。它由一系列的机械元件构成，包括组成机器人身体的接插塑料件，作为机器人腿脚的轮子和电机，作为机器人器官的传感器[14]等。这部分用于组成机器人躯干部分，用户设计时根据不同的需要可以搭建成不同的形状，并无特殊要求。这部分内容由协作开发单位的机械工程师来设计，因此在本论文中不做详细介绍。

2.2.2 硬件平台

硬件平台是一块包含了各种控制芯片及其驱动电路的电路板，这些芯片包括主控芯片，串行通信芯片，电机控制芯片，电源转换芯片等。它是作为机器人的心脏和大脑而存在的，机械平台搭建好以后将它固定在其中适当的位置，最好是不易被外界接触到的地方，因为它和人的心脏和大脑一样不能随便的被碰撞，以免发生损坏[15]。硬件部分由平台设计者设计，软件部分由用户通过上层的软件平台设计而成，通过串行通信口下载到硬件平台的主控芯片中。硬件平台通过外连的传感器了解外部的各种情况，作出判断后对机器人的各个部件发出指挥命令，从而控制机器人的所有行动。

2.2.3 软件平台

软件平台主要由两部分组成：初级用户使用的图形化设计界面和高级用户使用的类 C 语言设计界面。将图形化界面设计出的程序转化为类 C 语言，然后通过编译器编译，最终下载到硬件平台的主控芯片中。如图 2-1 所示，为软件平台的结构图。

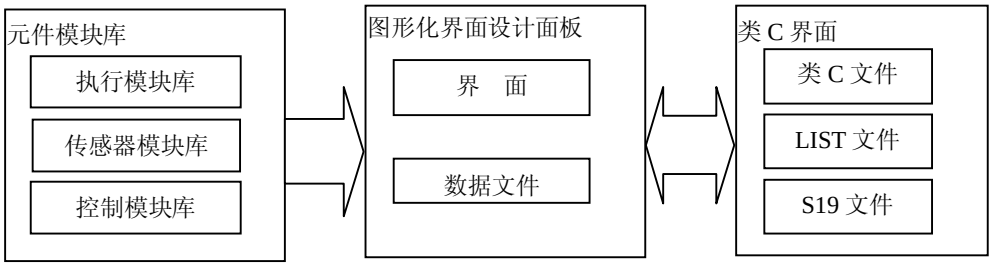


图 2-1 软件平台结构

(1) 类 C 语言设计界面

类 C 语言设计是利用类似于 C 语言的程序设计语法结构来开发机器人的运行程序。之所以称之为类 C 语言是因为在其设计的语法结构和 C 语言基本相同，因而它的编程方法和标准的 C 完全相同。不同之处在于，由于在主控芯片底层程序中固化了部分的通用子程序，在本软件平台中提供了该接口函数，因此在用户直接用类 C 设计机器人程序时可以直接调用该接口函数。

(2) 图形化设计界面

图形化设计界面包括两部分即元件模块库和设计面板。

元件模块库包括：执行模块库，传感器模块库，控制模块库和子程序控制模块库。

执行模块库中包含各种行动指令，包括前进、后退、转动等；传感器模块库中包含了接收外部环境信息的各种传感器部件，包括温度、灰度、红外等各种传感器；控制模块库中包含各种控制程序执行流程的部件，包括直到循环，条件循环，分支判断等；程序控制模块库中包含子程序控制部件，包括子程序开始、子程序调用。

设计面板用于存放从模块库中选中的控件对象，用户设计机器人程序时将这些控件对象组合成完整的流程图，然后转化为类 C 语言程序，最后编译下载到机器人主控芯片中。

(3) 编译功能和下载功能

既然使用了类 C 编程语言，其编译也必然要使用平台软件提供的编译器进行。因为此类 C 语言并不是真正的嵌入式 C 语言，而是设计时为了便于用户使用而设计的一种中间语言代码，它在编译之前必须经过本平台软件的处理后转化成为标准的嵌入式 C 语言，然后再调用编译器编译并下载。

2.3 SD-HCS08-Robot 开发平台的特点

2.3.1 图形化界面设计

图形化界面设计是本平台软件设计的一大特色。它使得用户在设计程序时真正的做到不需要了解任何的编程语言规范，不要写一句的程序代码，即可实现编程，这使得中小学生更容易接受和使用 SD-HCS08-Robot 开发平台，避免了中小学生要使用开发平台就必须学习高级编程语言的尴尬。

2.3.2 类 C 语言使用

类 C 语言使用是本平台软件的另一特色。标准 C 对于学过计算机编程的人来说比较熟悉，而嵌入式的 C 语言对于大部分用户来说是就很陌生了。由于嵌入式的 C 语言要求对硬件内部的堆栈、内存和 Flash 作各种设置，操作时也和常用的标准 C 有很大区别，因而许多用户只能望而却步。由于本软件中使用了类 C 语言，它使得用户在使用时仍然像使用标准 C 一样方便，将复杂的配置工作和对底层的操作都交给开发平台来完成。这样就使得使用标准 C 的用户可以直接用 C 来设计机器人程序。

2.3.3 硬件平台和机械平台分离设计

在前面我们提到在众多的开发平台上均是将硬件结构封装起来，不利于用户的动手操作。为此在设计这个 SD-HCS08-Robot 开发平台时采用了硬件平台和机械平台分离的设计方法，将硬件平台独立于机械平台而设计。这样使得用户可以任意的搭建机械平台，最终只要将硬件平台电路板固定在机械平台框架上即可。没有必要为防止硬件平台的损坏而要将其封装起来，而只要在搭建机械平台时稍作注意为其留出一个适当的位置。

第三章 硬件设计

3.1 选型的原则

在嵌入式产品设计中，硬件选型的好坏将直接影响着产品设计进度，同时也决定了产品的性能，还可能会影响到产品成形后的生产。因而硬件选型是嵌入式产品设计的一个重要环节。在硬件选型时应该综合考虑成本问题，开发的难易程度问题，元件购买途径问题，用户需求问题等等。以下将以主控芯片为例从技术角度来阐述一下对硬件的选型原则。由于在嵌入式的开发中，所有的设计都是围绕着主控芯片展开的，所以对它的选型问题就显得尤为重要，它应该遵循以下的原则：

(1) 合理的 RAM 和 Flash 大小

(2) 通用的 I/O 引脚数目

(3) 内部包含的功能模块

(4) 芯片的封装形式

(5) 写入器，编译器和集成开发环境

目前市场上的芯片种类很多，芯片价格也从几元到几十元不等，因而从这方面考虑对于开发者来说的可选择性很大。由于最终的用户只要实现功能对具体使用哪一款芯片并无太大要求，因此在设计过程中还是尽量选用自己比较熟悉的芯片开发，这样会节省开发的周期。RAM 和 Flash 的大小与 I/O 引脚的数目以及芯片的价钱基本上是成正比关系的，价钱越贵的芯片前两者也就越大。所以应该选择一个性价比相对比较合适的芯片才不至于浪费芯片资源，增加产品的成本。芯片的内部功能模块应结合用户的开发需求来定，尽量在所选择的芯片中能包含大部分的用户要求，这样才能物尽所用。芯片的封装形式是出于对实验和后期生产的考虑。通常的封装形式有双列直插型和贴片形式，双列直插封装实验起来比较方便，而贴片形式相对比较麻烦，但是其体积小，节约空间，对于空间比较紧张的产品则必须选用这种形式。牵涉到芯片就必须要考虑到写入器，编译器以及集成开发环境的问题。通常进行嵌入式开发使用两种语言即汇编和 C，各个厂家的芯片对这两种语言所定的标准并不相同，因而选用某种芯片时就必须要考虑到程序开发和编译写入的问题。通常一个通用的写入器要几千到上万元，一个带编译器的集成开发环境也要几万元，这对于一般的实验性小项目是不

适合的，只有进行自主开发，一次性开发完成，以后做类似的课题就可以进行移植，不必从头再来了。

在作者以前的研究和开发过程中一直使用的是FreeScale公司(原Motorola公司的半导体部分)的 MC68HC908GP32 芯片(以下简称 GP32)[16]，但在本系统需求分析后时，发现这款芯片不能适合用户要求，具体表现如下：

(1) 机器人平台中要使用到多路的 PWM 信号，而 GP32 芯片中只有两路 PWM 信号输出，不能够满足要求。

(2) 机器人平台中要求至少 35 个通用 I/O 引脚，GP32 一共只有 34 个。

(3) 机器人平台中的超声波传感器要求能产生 40KHz 的频率输入，GP32 也难以办到。

针对以上问题，设计中决定采用 FreeScale 公司 2004 年刚刚推出一款新的增强型芯片 MC9S08GB60(以下简称 GB60)。这款芯片中拥有 8 路的 PWM 输出，56 个通用的 I/O 口，最高达 20M 的内部总线频率，弥补了 GP32 所存在的不足，满足了用户的要求，在下一节中将会详细介绍这款芯片。

本平台设计中要选择的另一款芯片是电机驱动芯片。因为电机驱动电流较大，因而不能简单地接在芯片的引脚上，必须用专门的驱动芯片来驱动。经过调研决定采用 L298 芯片来实现驱动，它的驱动电压最高可达 46V，工作电流可达 2A，是一个理想的小电机驱动芯片。

3.2 硬件说明

3.2.1 主控芯片 MC9S08GB60

GB60 芯片主要有以下特征[17]：

(1) 60KB 的 Flash 存储器，具有密码保护和在线编程能力

(2) 4KB 片内 RAM

(3) 2 个异步串行通讯 SCI 口，1 个同步串行通信 SPI 口

(4) 56 个通用 I/O

(5) 8 通道的 TIMER/PWM

(6) 8 个键盘中断口

(7) 8 个模数转换口

(8) 背景调试模式

(9) 100Kbps 的 I^2C 总线

(10) COP,IRQ,TRI 功能模块

GB60 的功能结构框图参见附录 A.1,其主要的功能模块介绍如下：

(1) CPU HCS08 核：GB60 的处理器使用了增强型的 HCS08 核，它虽然还是 8 位的处理器，但是比原来的 HC08 核已有了大大的增强，最高总线频率可达 40M，即最小的指令执行时间可达 25ns;最长的操作指令 RTI 需要 11 个指令周期,也只要 275ns。同时它增加了更多的 16 位操作指令，使得对于 16 位寄存器 HX 操作更加灵活方便。

(2) 存储器：GB60 寻址空间为 64K，存储器的映像图见附录 2 所示，其中包括：61268 字节的 Flash、4096 字节的 RAM、127 字节的直接页寄存器和 44 字节的间接页寄存器。

其中 Flash 区分为两块一块大小为 1920 字节，通常我们在其中存放驻留在系统中的一些程序，比如监控等；另一块大小为 59348 字节，这一块用来存放用户程序。

(3) 串行通信模块：GB60 芯片中有两个串行通信模块，它们二者所用的寄存器各不相同，相互独立工作，互不影响。

(4) 定时器和脉宽调制模块:GB60 中有两个定时器和脉宽调制模块，它们和PTD 口复用，PTD0～PTD2 对应模块 1 中的三个通道，PTD3～PT7 对应模块 2 中的 5 个通道，两个通道独立工作互不影响。

(5) 模数转换模块：GB60 中提供了 8 路的模数转换通道，它与 PTB 口复用。8 路通道互相独立工作。

(6) 背景调试模块：这一模块是新推出的增强型芯片中新增加的功能，它提供的背景调试方式数据通信速度快,调试方便,使得系统中实现写入和调试功能更加简单。这一部分内容在写入器一章中会作详细的介绍。

3.2.2 电机驱动芯片 L298

L298 是一双全桥电机驱动芯片，可驱动两组电机。具有以下的电气特性[18]：

(1) 电源驱动电压 Vs 可达 5V～46V，逻辑支持电压 Vss 为 4.5V～7V；

(2) 输入高电压 Vih 为 2.3～Vss，输入低电压为 0V～1.5V；

(3) 峰值驱动电流可达 3A，正常工作电流为 2A，总驱动电流可达 4A；

(4) 相应速度快，提供纳秒级的响应速度；

(5) 提供过温保护，工作温度范围可达-25℃～130℃，正常工作温度为 13℃～35℃。温度过高或温度过低时，芯片均会停止工作，防止其损坏。

拥有了以上的特性完全能够满足机器人硬件平台的电机驱动要求。

3.3 硬件电路设计

3.3.1 硬件功能概述

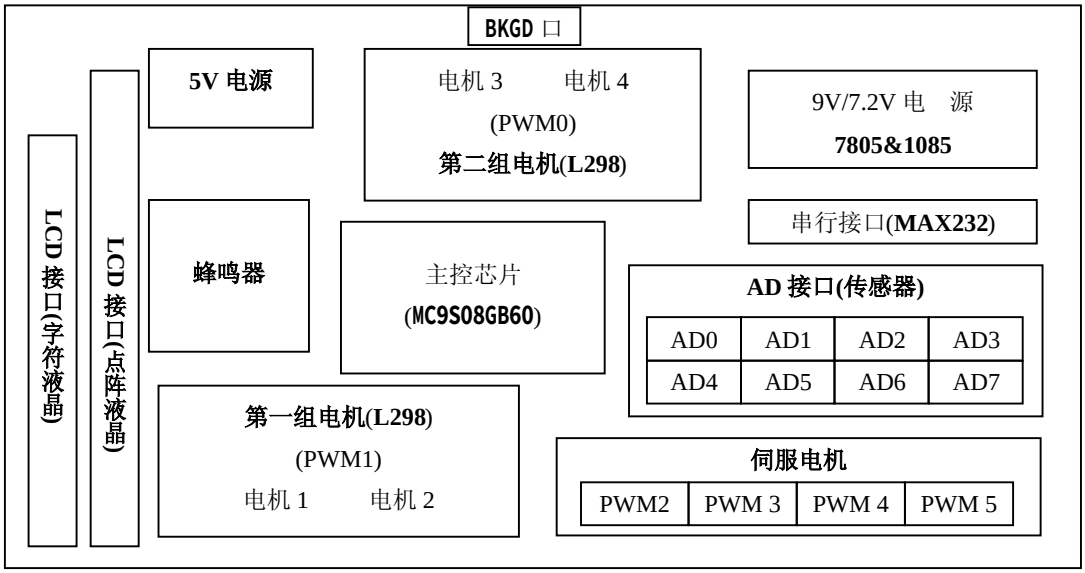


图 3-1 SD-HCS08 教育机器人基础开发平台的硬件电路板结构框图

如图 3-1 所示为 SD-HCS08-ROBOT 教育机器人基础开发平台的硬件电路板结构框图。由图中可以看出此硬件平台应该具有以下功能：

(1) 四路直流电机控制接口；

(2) 四路伺服电机控制接口；

(3) 8 路的 AD 信号接口(接传感器)；

(4) 两路液晶模块接口(汉字和英文)；

(5) 串行下载接口；

(6) 9V，7.2V，5V，3.3V 电源接口；

(7) BKGD 接口。

3.3.2 电路原理图

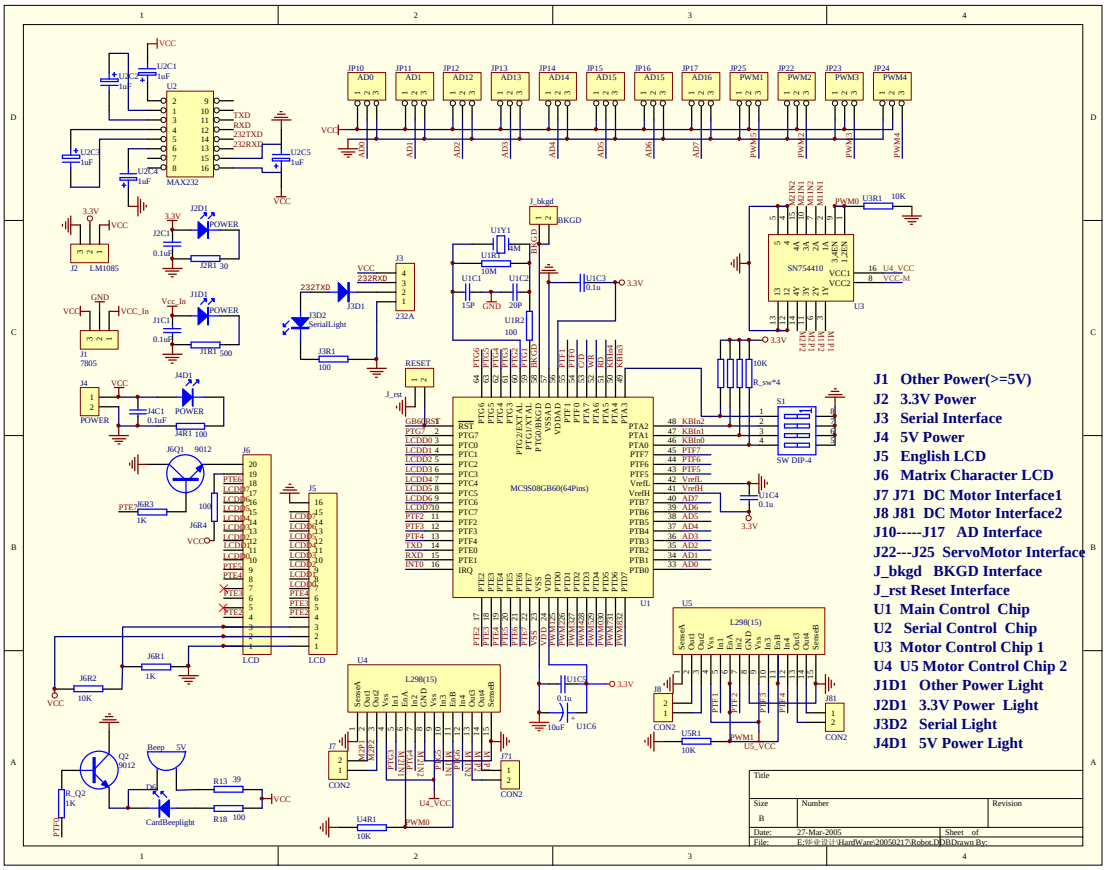


图 3-2 SD-HCS08 教育机器人基础开发平台的硬件电路板电路原理图

3.3.3 硬件连线

如图 3-3 为 SD-HCS08-ROBOT 教育机器人基础开发平台硬件电路板连线框图。由图中可以看出，硬件连线可分为六大块：主控芯片外围电路以及控制连线，电源控制电路，电机驱动电路，串行通信电路，传感器接口电路和液晶驱动电路。

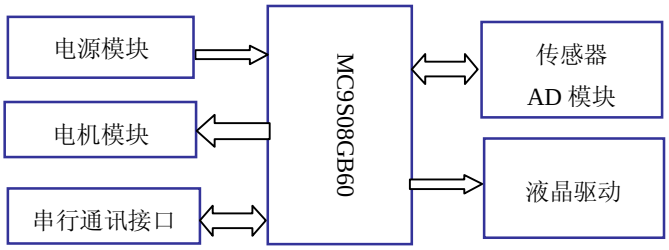


图 3-3 机器人平台硬件连线框图

(1) 主控芯片外围电路以及控制连线[17]。

在写入器一章中已经详细介绍了 GB60 芯片的外围电路，在此不再重复，以下将重点介绍 GB60 与外界接口的连线。

① PTA:PTA 口为程序选择开关接口，分别表示程序 1，程序 2，程序 3，程序 4。优先级从高到低分别为 1，2，3，4，表示系统执行时运行的程序，当四个开关均为打开时执行默认程序。

② PTB:PTB0～PTB7 接 AD 转换的 8 根数据线 AD0～AD7。

③ PTC:PTC0～PTC7 接 LCD 的 8 根数据线 DB0～DB7。

④ PTD:PTD0～PTD7 接 PWM 的 8 根数据线 PWM0～PWM7。

PTD0【25 脚】：(O)接 PWM1，第二组直流电机.

PTD1【26 脚】：(O)接 PWM2，伺服电机 1。

PTD2【27 脚】：(O)接 PWM3，伺服电机 2。

PTD3【28 脚】：(O)接 PWM4，伺服电机 3。

PTD4【29 脚】：(O)接 PWM5，伺服电机 4。

PTD5【30 脚】：(O)接 PWM0，第一组直流电机。

⑤ PTE:PTE0，PTE1 串行口发送和接收脚；PTE2～PTE7 分别接 LCD 控制信号脚。

PTE0【14 脚】：(O)串行口发送脚 TXD。

PTE1【13 脚】：(I)串行口接收脚 RXD。

PTE2【16 脚】：(I)接 LCD 的 A0 引脚，数据指令选择脚。

PTE3【19 脚】：(I)接 LCD 的/RD(E1)引脚,屏幕前半区域选择脚。

PTE4【24 脚】：(I)接 LCD 的 R/W 引脚，读写控制脚。

PTE5【22 脚】：(I)接 LCD 的 E2 引脚，屏幕后半区域选择脚。

PTE6【26 脚】：(I)接 LCD 的/RES 引脚，液晶块复位引脚。

PTE7【25 脚】：(O)接 LCD 的背光驱动引脚，低电平亮，高电平暗。

⑥ PTF:PTF0 蜂鸣器驱动脚，PTF1～PTF4 第一组电机驱动引脚。

PTF0【54 脚】：(O)接蜂鸣器及其指示灯的驱动脚，低电平响，高电平不响。

PTF1【55 脚】：(O)第一组电机驱动脚 1。

PTF2【11 脚】：(O)第一组电机驱动脚 2。

PTF3【12 脚】：(O)第一组电机驱动脚 3。

PTF4【13 脚】：**(O)**第一组电机驱动脚 **4**。

⑦ **PTG:PTG0** 为 **BKGD** 引脚，**PTG1**，**PTG2** 晶振，**PTG3～PTG6** 第二组电机驱动引脚。

PTG0【58 脚】：**(I)BKGD**。

PTG1【59 脚】：**(I)XTAL** 晶振 **1**。

PTG2【60 脚】：**(I)EXTAL** 晶振 **2**。

PTG3【61 脚】：**(O)**第二组电机驱动脚 **1**。

PTG4【62 脚】：**(O)**第二组电机驱动脚 **2**。

PTG5【63 脚】：**(O)**第二组电机驱动脚 **3**。

PTG6【64 脚】：**(O)**第二组电机驱动脚 **4**。

(8) 其他

Vss 【3 脚】：电源地。

Vdd 【7 脚】：**3.3V**。

RST 【28 脚】：复位脚。过 **10K** 电阻接 **VCC**，过 **0.1uF** 的电容 **C2** 接地；复位按钮与一 **51** 欧姆电阻串连并将它们并联接到电容 **C2** 两端。

(2) 电源控制电路

由于系统工作时有三路电源，即主控芯片工作电源 **3.3V**，电机及其驱动芯片工作电源 **5V**，外接电池电压通常为 **6V～9V,**因此在设计时选用了两个电源转换芯片 **L7805C** 和 **LT1085**。具体介绍如下：

① **L7805C 5V** 电源转换芯片

【1 脚】VCC_in 【2 脚】GND 【3 脚】VCC

VCC_in 输入大于 **5V** 时，输出引脚 **VCC** 为 **5V;**当 **VCC_in** 输入小于 **5V** 时，输出引脚 **VCC=0**。

② **LT1805 3.3V** 电源转换芯片

【1 脚】VCC 【2 脚】VDD 【3 脚】GND

VCC 输入大于 **3.3V** 时，输出引脚 **VCC** 为 **3.3V;**当 **VCC_in** 输入小于 **3.3V** 时，输出引脚 **VDD=0**。

(3) 电机驱动电路

① 直流电机控制

【1 脚】SensorA 【2 脚】Out1 【3 脚】Out2

【4 脚】Vss 工作电源 【5 脚】In1

【6 脚】EnableA 电机驱动引脚 1,2 使能 【7 脚】In2

【8 脚】Gnd 【9 脚】Vss 工作电源 【10 脚】In3

【11 脚】EnableB 电机驱动引脚 3,4 使能位 【12 脚】In4

【13 脚】Out3 【14 脚】Out4 【15 脚】SensorB

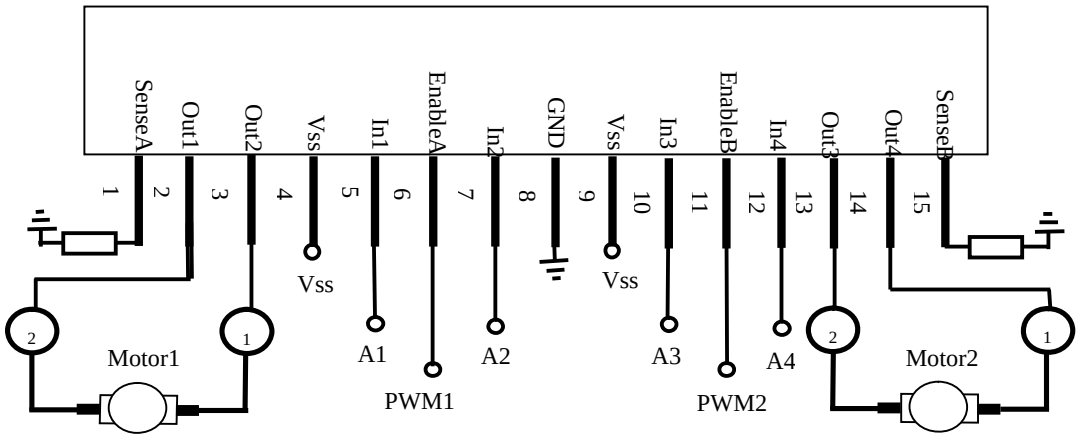


图 3-4 L298 驱动电机示意图

如图 3-4 所示为 L298 驱动电机的连接图[18]。在图中 Vss，GND 接电源和地。Sense 脚过感应电阻接地，用于控制驱动电流的大小,通常此感应电阻接 0.5 欧姆。In1，In2，In3，In4 接芯片的控制引脚，Out1，Out2，Out3，Out4 接电机控制引脚。InX 与 OutX 成为一对，控制线 InX 高，OutX 高；控制线 InX 低，OutX 低。InX 驱动电流较小，OutX 驱动电流较大。当然 InX 能控制 OutX 的前提是 EnableX 引脚上加了高电平，因而通过控制 EnableX 的高低电平的占空比可以控制电机转动的速度。如此通过其中的一对控制引脚比如 A1，A2 就控制一个电机的转动：如 PWM1=1，A1=1，A2=0，电机正转；PWM1=1，A1=0，A2=1，电机反转；PWM1=1，A1=0，A2=0，电机停转；PWM1=1，A1=1，A2=1，电机停转。通过两个电机的协同工作就可以完成机器人的行进动作了。

② 伺服电机控制

【1 脚】Vss(GND) 【2 脚】Vcc

【3 脚】Control

如图 3-5 为伺服电机控制示意图,用它来控制机器人的手臂的转动.标准的伺服电机有三条控制线，分别

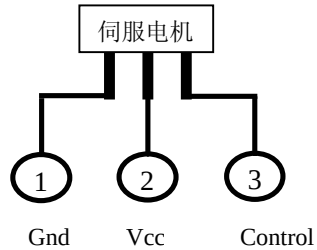


图 3-5 伺服电机控制示意图

为：**Vcc**、**Gnd** 及 **Control**。**Vcc** 与 **Gnd** 用于提供内部的直流电机及控制线路所需的能源，电压通常介于 **4V**—**6V** 之间，该电源应尽可能与处理系统的电源隔离(因为伺服电机会产生噪音)。输入一个周期性的正向脉冲信号，这个周期性脉冲信号的高电平时

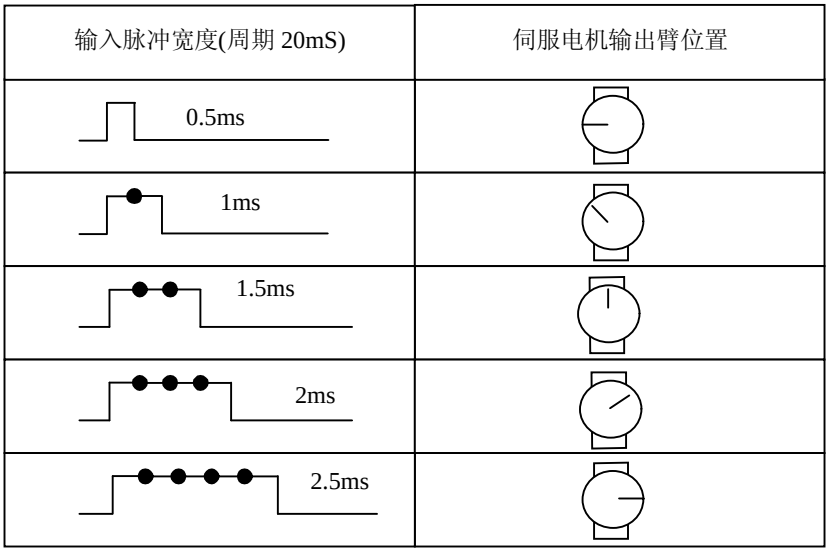


图 3-6 输入脉冲与伺服电机输出臂位置对应图

间通常在 **1ms** 到 **2ms** 之间，而低电平时间应在 **5ms** 到 **20ms** 之间，并不很严格[19]，图 **3-6** 给出了一个典型的 **20ms** 周期性脉冲的正脉冲宽度与微型伺服电机的输出臂位置的关系。

(4) 串行通信电路

【11 脚】主控芯片串行发送脚 【12 脚】主控芯片串行接收脚

【13 脚】232TXD MAX232 发送端，过 100 欧姆接串行指示灯 D3。当 MCU 通过串行口向外发送数据时，D3 闪亮。

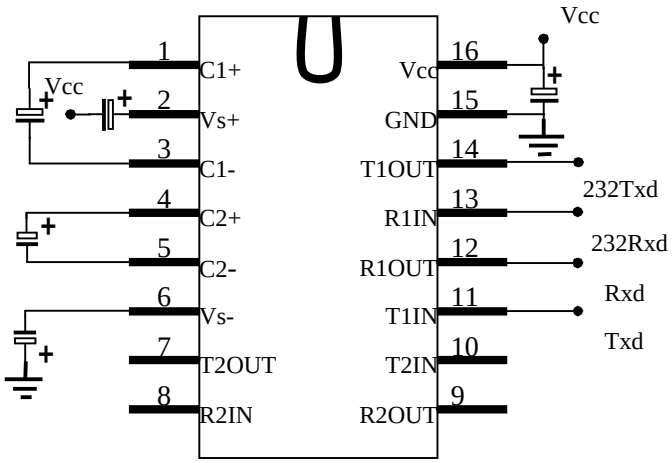


图 3-7 MAX232 引脚示意图

【14 脚】232RXD MAX232 接收端。

【15 脚】GND 接地。 【16 脚】VCC 电源。

主控芯片 **GB60** 通过 **MAX232** 芯片实现与 **PC** 的通信，由此来完成程序的下载。**MAX232** 的主要功能是实现信号电平的转化，在发送端需要用驱动电路将 **TTL** 电平转换成 **RS-232C** 电平，在接收端需要用接收电路将 **RS-232C** 电平转换为 **TTL** 电平[20]。

(5) 传感器接口电路

传感器接口是用来采集各种传感器的信号，**GB60** 提供 **8** 路 **10** 位的 **AD** 采样，具体位置见图 **3-8** 的结构框图。**AD** 的接口可连接灰度传感器，红外传感器，超声波传感器,钢铁传感器，噪音传感器等各种类型的传感器，图 **3-8**，给出了传感器与 **AD** 口的接线方式。**Vcc** 和 **Gnd** 可为外接的传感器提供电源，从 **Signal** 引脚可获得 **AD** 的采样信号。

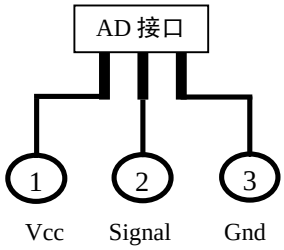


图 3-8 AD 接口连线示意图

(6) 液晶驱动电路

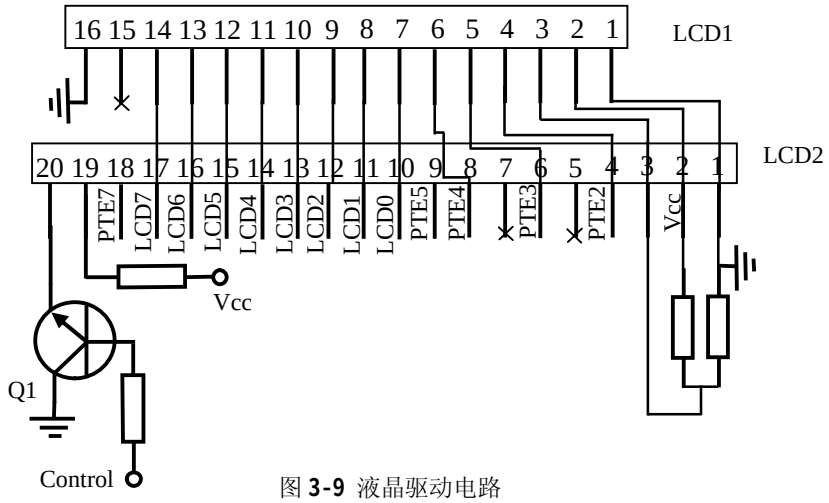


图 **3-9** 液晶驱动电路

① **LCD1** 英文液晶驱动引脚

【1 脚】GND 【2 脚】VCC 【3 脚】VEE 【4 脚】RS(PTE2)

【5 脚】R/W(PTE3) 【6 脚】E(PTE4) 【7 脚】～【14 脚】DC0～DC7

【15 脚】空 【16 脚】GND

② **LCD2** 点阵液晶驱动引脚

【1 脚】Vss(GND) 【2 脚】VCC 【3 脚】Vo 【4 脚】A0(PTE2)

【5 脚】NC 【6 脚】R/W E2(PTE3) 【7 脚】NC

【8 脚】/RD(E1) (PTE4) 【9 脚】R/W(PTE5)

【10 脚】~【17 脚】DB0～DB7(PTC0~PTC7) 【18 脚】/RES(PTE6)

【19 脚】LEDA(驱动) (VCC) 【20 脚】LEDK(PTE7)

如图 3-9 所示为液晶驱动模块的连接图，图中给出了两套液晶连接方式，即英文字符液晶[21]和点阵字符液晶[22]。它们二者复用了部分引脚，因而二者不可同时使用。点阵液晶可以用来显示汉字或者是任意的点阵图形。

3.3.4 硬件测试

嵌入式系统包含软硬件两部分内容，而软件的设计必须建立在稳定的硬件基础上。当硬件设计完成后，进行布板加工，电路板做好以后就要进行焊接了。当电路板第一次焊接时，并不知道电路板是否有问题，一般采用边焊接边测试的方式进行，这样可以避免由于所有的元器件都焊好后才发现有问题，这时很难定位错误的位置。当然对于可以独立测试的元件也可以把相关的元件焊好后做独立的测试。

在本平台电路板焊接时要遵循以下测试步骤：

(1) 电源测试

拿到电路板后首先测试电源和地是否短路，如果没有短路焊接电源转换芯片 L7805C 和 LT1085，焊接完毕后观察相应的指示灯是否正常工作。然后用万用表测试电压是否正常，一切正常，继续焊接。

(2) 主控芯片写入监控测试

主控芯片的监控程序写入是通过写入器写入的，它只要简单的外围支持电路就可以完成写入。当写入完成后应该从串行口不断的向外发出握手信号，电路板上的串行指示灯不断的闪烁。利用串口调试工具可以收到握手信号 73。

(3) 主控芯片在线写入测试

当串行模块全部焊接好以后就可以通过串行口和上层的集成开发环境相连接，实现程序的在线的写入。

至此说明芯片已经正常工作。其他元件可以焊接了。

(4) 独立的电机驱动测试

电机驱动芯片由于其工作并不一定要主控芯片来控制，直接用电平信号驱动也能够实现，所以这一部分内容可以单独的测试。接线方式如图 3-4 所示，在 A1，A2，

PWM1 上打上不同的电平信号，测试是否能够达到预期的驱动效果。

3.4 硬件设计过程中的体会

经过近三年的硬件设计训练结合毕业设计过程中的体会对硬件设计过程有了如下的一些认识：

(1) 需求分析是设计过程的一个重要环节

需求分析就是用户提出要求，设计者根据要求给出设计方案的过程。这个过程对于硬件设计尤为重要，因为硬件设计时要进行原理性的分析，而如果对用户的要求理解有偏差，得到的设计原理必然是有出入的，当最后落实到电路板上时可能已经和用户的需求大相径庭了。为此在设计初期要和用户有充分的沟通，切实地了解其对设计的要求。

(2) 硬件设计的过程是一个不断实验的过程

嵌入式开发本身就是一个不断实验的过程，硬件设计中反映的更加明显。从设计之初的需求分析，到设计中期的程序设计，再到后期的调试维护，其实都是在实验中进行的。为此在硬件设计时就要做到多动手，多做实验，并对实验做好记录，对实验数据认真分析总结，得出正确的实验结论。

(3) 布板设计

布板设计是硬件设计的一个必然过程，它决定了最终产品的硬件外形。对布板设计有以下几点心得：

① 布板设计前要对原理要分析清楚

原理正确是最终布板正确的保证。原理正确与否很大程度上决定于前期实验是否做的正确，对实验结果分析是否正确。因而在分析系统原理时必须有足够的实验做保证。基本原理确定正确后，还要对各个部件的使用进行筛选，使其尽量的合理。

② 布线前调整好元件的位置

电路板设计过程中，在走线前一定要将所有的元件调整到适当的位置，这样可以使得在走线时比较通畅，相反元件放置位置不当可能使走线很困难，甚至难以走通。当使用机器布线时，在机器走线结束后要作适当的调整，将不规范的走线调整过来。

③ 布板时多留测试用点

布板时多留测试点可以使得对电路板测试时更加方便，同时也能够在软件设计遇

到问题时，及时地判断出是硬件电路的问题还是软件信号的问题。基本的测试点包括电源与地，以及各个功能模块的驱动引脚等。

④ 布板时要留足余量

这一点也主要是为了在最终使用时有多种的选择余地，以及为后期的扩展做好准备，使得设计的板子具有可重用性，可扩展性。但是当最后成为产品时要去掉那些不必要的内容，使得成型板相对简单，这样可以保证产品的稳定性。

⑤ 连接接口选择

最后说的各种连接接口的选择。连接的接口是本系统和外界相联系的通道，通常使用软线相连。各种接口一般也有相对的标准，比如串口用 **DB9**，网络口用 **RJ45** 等。但是这种接口方式并不是不可变化的，例如在我们通常使用串行口时只用了三根线，即地线，发送线和接收线，此时我们就不一定要使用 **DB9** 接口因为它比较大，很占空间，可以换成 **USB** 接口，其中可以接四根线，所以还可以将电源接进来，这就是一种变通的方式。

第四章 写入器

写入器的设计是每款芯片使用前的准备工作，它主要是完成向空白芯片中写入已经编译好的机器代码。每一款芯片的内部机构不同，引脚封装不同，因而也就不可能完全做一个对于每一款芯片都适用的写入器。这就要求在使用芯片时，特别是刚刚推出的新的芯片，必须首先解决写入器的设计问题。在本平台中使用的是 Freescale 公司最近推出的 MC9S08GB60 芯片，以下将简要介绍一下此款芯片写入器的设计思路。

4.1 硬件设计

MC9S08GB60 芯片新增了背景调试控制模块(BDC Background Debug Controll)。该模块提供了方便的写入和调试功能。在此平台中利用了 BKGD 引脚与 MC9S08GB60 芯片进行单线的通信，实现写入和读取操作，硬件接线如图 4-1 所示。

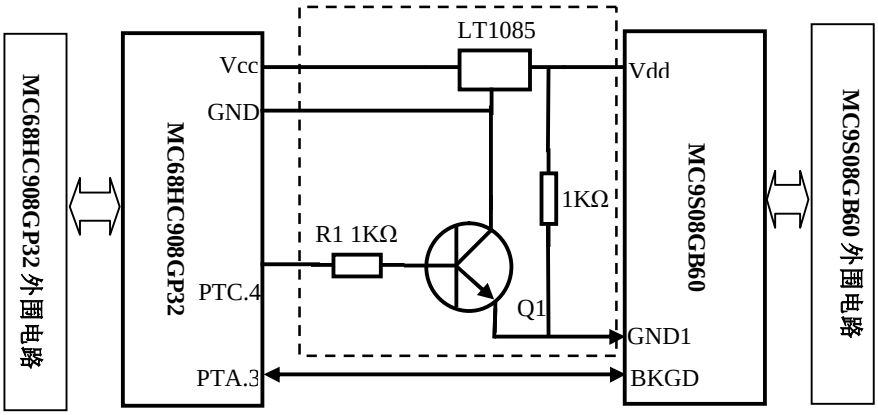


图 4-1 写入器硬件接线

4.1.1 MC68HC908GP32 外围电路

MC68HC908GP32 外围工作电路包括其晶体震荡电路,锁相环电路，电源模块电路等，由于 MC68HC908GP32 并非本论文介绍的重点，在此不再赘述，详细内容请见参考文献 16。

4.1.2 MC9S08GB60 外围电路

MC9S08GB60 外围电路包括晶振电路和电源滤波电路两块。晶振电路如图 4-2 所示。

EXTAL【60 脚】外接晶振引脚。

XTAL 【59 脚】内部晶振引脚。

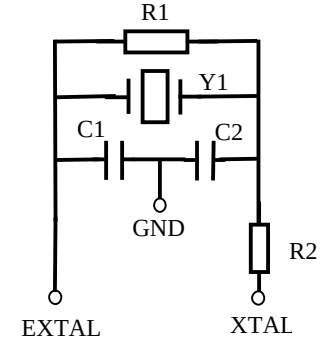
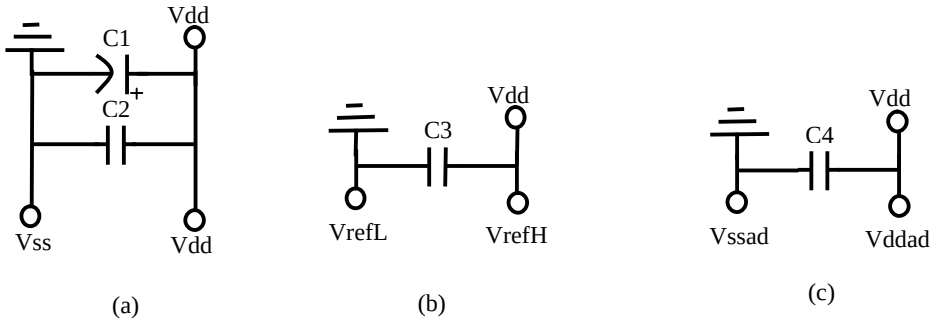


图 4-2 MC9S08GB60 晶振电路

外接 Y1 为 4M，反馈电阻 R1 为 10MΩ，串行电阻 R2 为 100Ω，C1 为 15P，C2 为 20P。在写入器中选用使用外部晶振来提供芯片的工作时钟。由此电路可产生 4M 的时钟频率。

滤波电路如图 4-3 所示：



(a)　(b)　(c)

图 4-3　MC9S08GB60 电源滤波电路

Vss 【23 脚】电源地

Vdd 【24 脚】外接电源

VrefL【42 脚】AD 转换模块参考地

VrefH【41 脚】AD 转换模块参考电源

Vssad【57 脚】AD 转换模块电源地

Vddad【56 脚】AD 转换模块电源

以上为芯片上用到的电源引脚，通常 Vdd，VrefH，Vddad 外接 3.3V 电源，Vss，VrefL，Vssad 同时接地。在所有的电源和地上接 0.1μF 的电容滤波，并在系统地和电源 Vss，Vdd 上接一滤高频信号的 10μF 电解电容。这样系统就会更加稳定了。

4.1.3　电源控制电路

图 4-1 中 GP32 为主控的写入芯片，GB60 是待写入的芯片。GB60 的电源由 GP32 的 PTC.4 控制供电，虚线框所示区域为电源控制部分。GP32 的工作电压为 5V，GB60 工作电压为 3.3V，系统外接 5V 稳压电源直接供给 GP32，通过 LT1085 转换后给 GB60 供电源 3.3V。PTC.4 控制 GB60 的地线 GND1，通常拉高到 Vdd；当 PTC.4 打高电平时三极管 Q1 导通工作，GND1 拉低到 GND，此时 GB60 可以正常工作，

4.1.4　信号传输电路

BKGD【58 脚】背景调试引脚

背景调试方式利用一根通讯线 BKGD 与主控 MCU 进行串行通信。在此写入其中将其与主控芯片 MC68HC908GP32 的 PTA.3 相连接。当芯片上电后，它们二者之间就可以进行数据通信了。

4.2 软件设计

软件设计的基本思想是利用芯片提供的固化在芯片内部的 BKGD 调试命令，先将擦除和写入 Flash 的程序以及监控程序的机器代码写入到 MC9S08GB60 的内存区，然后运行在内存中的擦除和写入 Flash 的程序把内存区的监控程序代码固化到 Flash 中。这样下次芯片复位后就会首先运行监控程序，与上层的软件配合就可以完成在线的调试写入了。以下将介绍实现过程中的几个关键技术。

4.2.1 BKGD 通信方式

BKGD 通信方式要求所有通信操作在一根信号线上完成，并且对于时序要求很高。双方通信过程中一位数据传输的时间大约为 13 个时钟周期。本系统中系统时钟

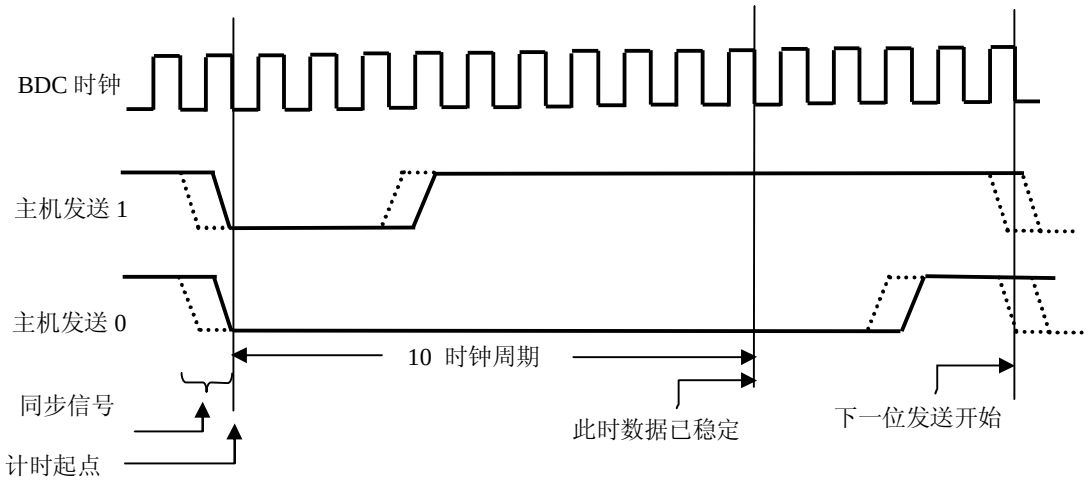


图 4-4 BDC 模块主机向目标芯片发送信号时序图

频率为 4MHZ，那么利用 BKGD 信号线传输波特率为 $1/(8*13)/(4*10^{6})$=40000bps。而通常使用的串口通信的波特率仅为 9600bps 左右。所以使用 BKGD 方式实现写入比通常的串口通信要快的多。以下具体介绍 BKGD 通信方式中主控芯片和目标芯片间的通信时序。

(1) 发送一位数据的时序关系

如图 4-4，当要向目标芯片发送数据时应该遵守以下的时序关系

① 将主机通信脚设置为输出状态，BKGD 脚上置为高电平。

② 主机发送 3~4 个时钟周期低电平作为握手对目标芯片的提示信号。

③ 主机发送 10 个周期的高或低电平信号作为正式的数据信息。

④ 将 BKGD 引脚置为高电平。

⑤ 继续下一位发送。

在以上的时序过程中从计时开始到第 10 个周期时，数据信号已经稳定此时目标芯片开始采样。

(2) 接收一位数据的时序关系

接收数据仍然由主机主动发出握手的低电平启动传输，具体时序如图 4-5 和 4-6 所示。

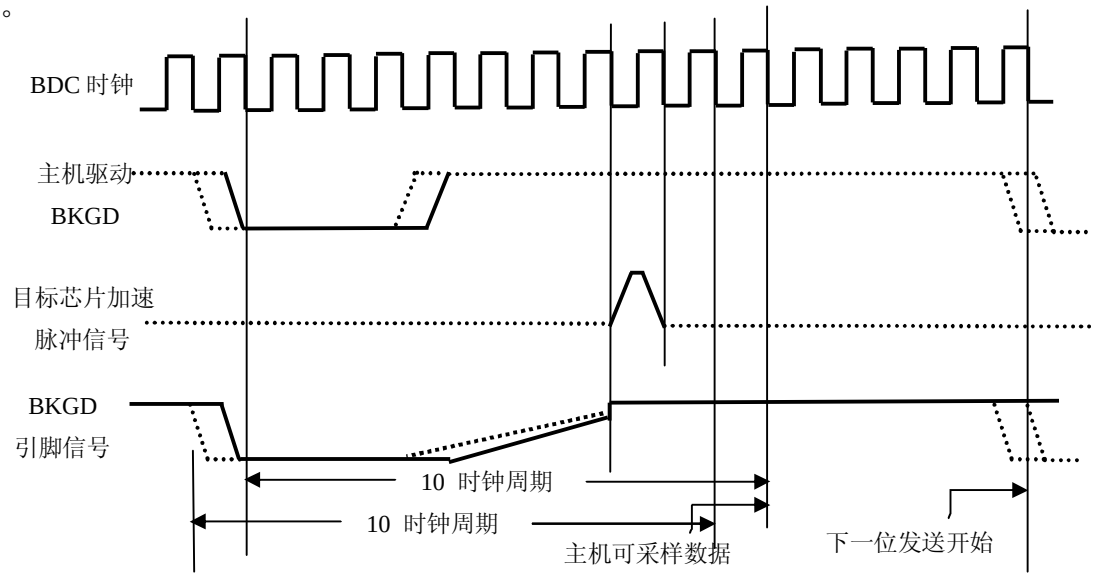


图 4-5 BDC 模块目标芯片向主机发送信号 1 时序图

图 4-5 给出了 BDC 模块目标芯片向主机发送信号 1 时序图，其时序结构如下。

① 将主机通信脚设置为输出状态，BKGD 脚上置为高电平。

② 主机发送 3~4 个时钟周期低电平作为握手对目标芯片的提示信号。

③ 主机撤销低电平信号，将主机通信脚设置为输入状态等待接收目标芯片的输出信息。

④ 计时开始 10 个周期左右时目标芯片发送的数据已经稳定，主机可以采样。

⑤ 13 个周期以后将主机通信脚设置为输出状态，BKGD 脚上置为高电平。

⑥ 启动下一位接收。

图 4-6 给出了 BDC 模块目标芯片向主机发送信号 0 时序图，其时序结构和接收一个字节的时序相同，唯一区别只是接收到的数据是低电平。

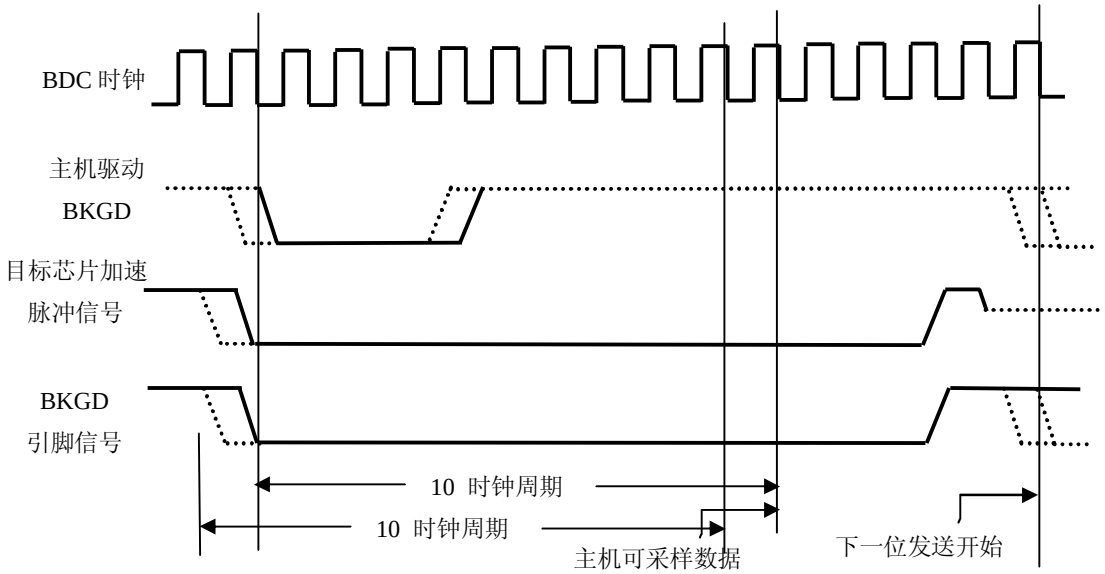


图 4-6 BDC 模块目标芯片向主机发送信号 0 时序图

在进行系统编程时要注意两边时钟信号的协调。主控芯片和目标芯片的工作时钟不一致时，为了保证目标芯片能够达到预定的周期数，在计算主控芯片方周期数时要做相应的调整。在本写入器设计时，主控芯片工作频率为 4.9MHZ，而目标芯片的工作频率为 4MHZ，因此主控芯片的 5 个周期与目标芯片的 4 个周期时间基本相同，进行周期换算时即可按 5:4 的比例计算。

4.2.2 BKGD 调试命令

BDC 模块共提供了 30 条调试指令，在设计写入器时只要使用其中的 5 条指令分别是 BACKGROUND，READ_BYTE，WRITE_BYTE，GO，WRITE_PC。

(1) BACKGROUND 是进入背景调试模式指令。代码格式为 90/d，90 为指令代码，d 表示要延时 16 个指令周期。

(2) READ_BYTE 是用于从目标芯片存储区中读出一个字节数据指令。代码格式为 E0/AAAA/d/RD：E0 为指令代码，AAAA 为 2 字节的待读出存储区的地址， RD 为读出的 8 位数据。

(3) WRITE_BYTE 是用于向目标芯片存储区中写入一个字节数据指令。代码格式为 C0/AAAA/ WD /d：C0 为指令代码，AAAA 为 2 字节的待写入存储区的地址，WD 为待写入的 8 位数据。

(4) GO 是要求芯片从 PC 寄存器中的地址开始执行。指令代码为 08/d，08 为指令代码。

(5) WRITE_PC 是写程序寄存器指令。指令代码为 4B/WD16/d，4B 为指令代码，WD16 为 2 字节待写入程序寄存器的地址信息。

通过以上指令的组合即可完成对 MC9S08GB60 的存储区中写入数据，并且可以启动程序运行。这样根据本节开始所说的方法，先把准备运行的程序和代码写到内存区，然后启动程序运行把监控代码固化到 Flash 中即可完成监控程序的烧入工作。

4.2.3 MC9S08GB60 的擦除写入 Flash 技术

(1) Flash 操作包含了八个寄存器[18]：

① 时钟分频寄存器(Flash Clock Divider Register FCDIV)　此寄存器用于设置 Flash 擦除和写入的分频因子。此选项在每次复位后被置为 0 且只允许修改一次，修改操作必须在程序尚未进行任何擦除和写入操作之前，此寄存器进行了修改或者进行了任意的一次擦除和写入后，此寄存器只读。

② 选项寄存器(Flash Option Register FOPT and NVOPT)，此寄存器用于控制是否启用安全机制。当安全机制启动后，要求用户必须输入正确的密码信息后才能启动 Flash 的擦除读写操作。当系统复位后，NVOPT 的值被复制到 FOPT 中，FOPT 内容不可以任意的修改。由于不能直接修改 FOPT 寄存器，因而当要修改 FOPT 的值时，只要在 NVOPT 先做修改然后重新复位。

③ 配置寄存器(Flash Configuration Register FCNFG)，此寄存器用于配置系统是否允许对密码访问位的写入。当设置为允许写入时，对密码区的写入被视为 Flash 区写入的开始，而当设置为不允许写入时，对密码区的写入则视为启动了一次密码比较。

④ 保护寄存器(Flash Protection Register FPROT and NVPROT)，此寄存器用于在进行 Flash 的擦除时对于其他未擦除区域进行保护。当系统复位后，NVPROT 的值被复制到 FPROT 中，FPROT 内容不可以任意的修改。由于没有直接修改 FPROT 的指令，当要修改 FPROT 的值时，只要在 NVPROT 中先做修改然后重新复位。

⑤ 状态寄存器(Flash Status Register FSTAT)，此寄存器给出了 Flash 擦除和写入的过程中的各种状态，包括 Flash 指令寄存器是否为空，Flash 指令是否执行完成，块保护机制是否启动，Flash 指令执行过程中是否有错等。

⑥ 命令寄存器(Flash Command Register)，此寄存器用于设定 Flash 操作指令。目前共提供了 5 种操作指令分别为：判空检查(Blank check)，按字节写入(Byte program)，快速 按 字 节 写 入 (Byte program-)，按页擦除，整体擦除。

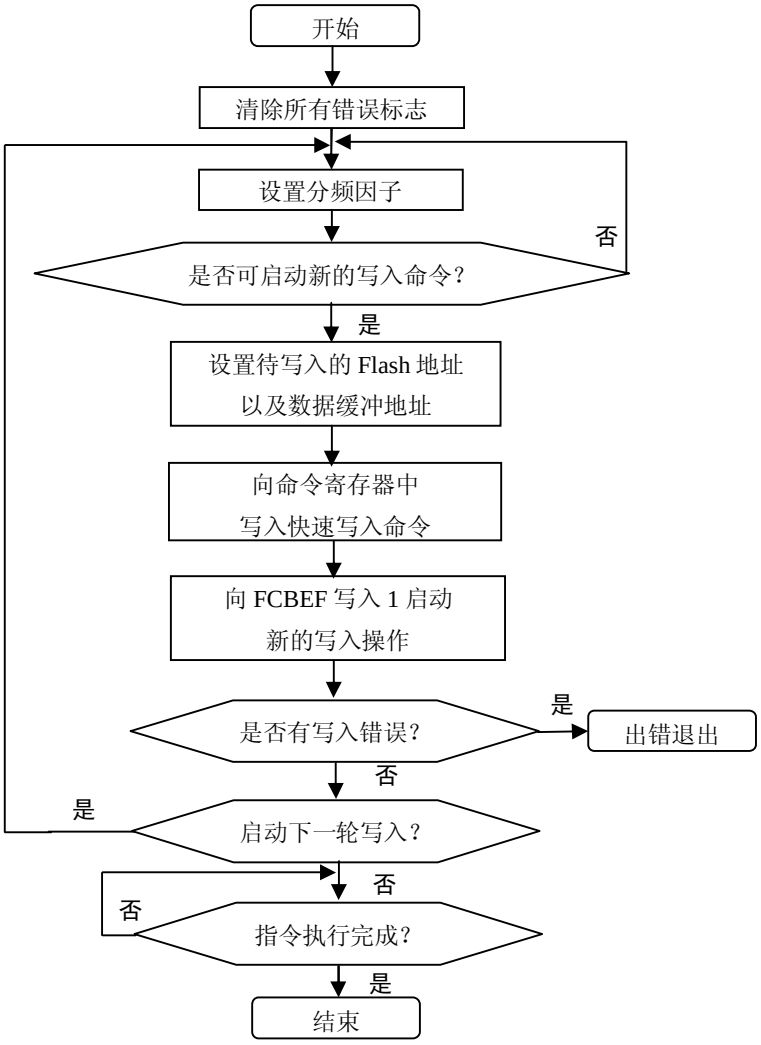


图 4-7 快速写入流程图

(2) 整体擦除与 按页擦除。整体擦除是指将芯片中所有固化在 Flash 中的数据一次性的完全擦除。在这种情况下不需要考虑原来芯片中包含了什么程序，擦除完毕后的芯片就恢复为一块 空白芯片，此时可以重新写入数据。按页擦除则是根据擦除前给定的页号信息(此信息通过向待擦除区域中任意地址中写入任意的值来完成),将其所在的一页内容擦除。MC9S08GB60 芯片中包含了 60K 的 Flash 空间，每页大小 512 字节，因而共有 120 页。使用页擦除时要注意对未操作 Flash 区域的保护，因为在 Flash 擦除和写入的过程中在 Flash 区域中加上了高电压，容易引起未操作区域的数据损坏。

(3) 快速写入

快速写入的流程如图 4-7 所示，由于在写入的过程，Flash 区的高电压一直不撤销，所以写入的时间将会大大缩短，所以称为快速写入。在此操作之前，必须注意对未操作区的保护。在写入器的设计过程中，我们将监控程序代码以及中断向量区的内容写入 Flash 区时就是使用此命令来完成。

4.2.4 MC9S08GB60 快速写入 Flash 子程序

MC9S08GB60 快速写入 Flash 子程序是为了实现将已经写入到 RAM 区的监控程序的内容和中断向量区内容快速写入到 Flash 中。执行流程如图 4-8 所示。

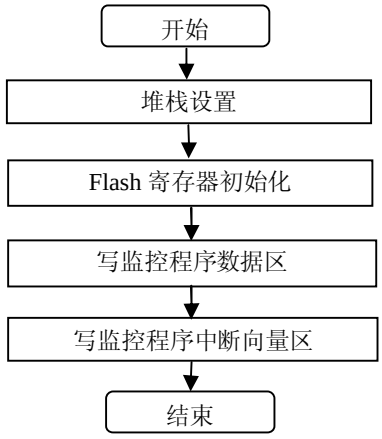


图 4-8　RAM 区内容写入 Flash 区程序流程

(1) 堆栈设置时把堆栈空间设置到 RAM 区的最末端，GB60 中为$107F。

```
LDHX  #RamLast+1
TXS          ;          堆栈指针初始化#107F
```

(2) Flash 寄存器初始化时把先把所有的 Flash 标志位清除掉，配置其时钟周期为 200KHz。

```
LDX   #%00110000  ;(mFPVIOL+mFACCERR) ;mask
STX   FSTAT       ;FSTAT,clear any error flags
STX   #%00010011  ;FLASH     时钟配置寄存器,
                  ;f/(19+1)=4M/(19+1)=200k#initFCDIV
STX   $1820       ;FCDIV,f(FCLK) = 200 kHz [4MHz xtal]        ，置为 200kHz
```

(3) 写监控程序数据区是将 RAM 区$0140~$1000 保存的监控程序数据写入到 Flash 区$1200 开始的区域内。

```
;----------------------------------
LDHX  #$0080
JSR    BusrtProg
```

(4) 写监控程序中断向量区是将 RAM 区$0110~$0134 保存的监控程序数据写入到 Flash 区$FFFE 开始的区域内。

```
;----------------------------------
;写入 FLASH 中的程序(监控)
LDHX  #$0086
JSR    BusrtProg
```

以上调用的程序中调用的子程序 BusrtProg 是实现将内存区的数据块写入 Flash 区,入口地址为 HX 为首地址的连续 6 个字节的地址空间，前两个字节为待读出的 RAM 区收地址，紧接着 2 个字节是待写入 Flash 区首地址，最后 2 个字节为待写入的数据长度。

4.2.5 主程序设计

主程序运行在主控芯片 MC68HC908GP32 上，控制对 MC9S08GB60 芯片的写入，具体工作流程如下。

(1) 系统初始化并启动 MC9S08GB60 的电源。

初始化设置中包括系统 Stop 模式的禁止与否及恢复时间，看门狗是否启动及其定时器的溢出时间，低电压禁止，串行通讯的波特率，PLL 锁相环等的设置。这些并非本设计的重点在此略过。启动 MC9S08GB60 的电源使其能够正常工作。

(2) Flash 区的整体擦除。

由于监控的写入不仅要针对空白芯片也要对已经灌入程序的芯片也能操作，所以在向其中写入数据之前不论是否为空白芯片均将原有的内容擦除掉，这样以后就可以方便的写入新的东西了。整体擦除时要将所有的区域都放开，不再设置保护。具体执行时分为三步走：

第一步 对 Flash 操作寄存器进行初始化。在此设置两项内容即取消整体保护和设置 Flash 工作的工作频率。在本写入器的设计中，将 Flash 分频因子设置为 19，由于系统总线频率为 4M，所以 Flash 工作频率为 4M/(19+1)=200K。

第二步 发送整体擦除指令，将 Flash 区域内容擦除。

第三步 进行 Flash 区判空检查，检查通过后将 RAM 区和 Flash 区放开，允许对其写入。

(3) 向 MC9S08GB60 写入相应得数据和程序。

将监控程序代码，Flash 写入程序代码，复位向量位置以及 Flash 写入程序的参数写入到 MC9S08GB60 的内存相应区域。写入内存区的数据形式如下：

$0080~$008B Flash 写入程序的参数

$0080 监控程序在内存块的首地址

$0082 监控程序在 Flash 中的首地址

$0084 监控程序的数据大小

$0086 中断向量在内存块的首地址

$0088 中断向量在 Flash 中的首地址

$008A 中断向量的数据大小

$0090~$00FF　Flash　写入程序代码

$0110~$0134　　复位向量地址

$0140~$1000　　监控程序代码

(4) 发送程序执行指令，运行 MC9S08GB60 内存区$0090~$00FF 的 Flash 写入程序代码，将监控程序写入到 Flash 中。

(5) 关闭 MC9S08GB60 电源，写入结束。

至此写入工作完成，MC9S08GB60 芯片再次上电复位，将从监控程序处开始运行，由于其中包含了可以重新写入用户程序的代码，因而可以实现在线的写入。

第五章 MCU 方软件设计

5.1 功能概述

MCU 方程序是指最终下载到硬件平台主控芯片中的所有程序。在主控芯片的软件设计中并不是所有的程序都是一次性设计完成，然后编译写入的。设计过程中将 MCU 方的软件进行适当的划分，将某些内容在底层适当的封装，使得最终用户在编程时忽略掉某些内容的实现细节，使用起来更加方便，这样做同时减轻上层软件的编程压力，提高上层软件的编程效率。

在设计时将 MCU 方软件划分成了三块内容，即监控程序，驻留子程序和用户程序。

监控程序是为了实现用户程序的在线写入而设计的一段短小精干的小程序，它是用专门的写入设备(写入器)写入到主控芯片中。

驻留子程序是软件中一些经常要用到的功能性子程序，比如延时程序，AD 采样程序等。它们的功能相对独立，最终用户也只需要知道其执行的结果，并不需要了解其内部的执行机制。设计时将这部分程序单独划出来进行编程调试，并在用户程序写入之前将其先写入到芯片中，并且保证用户程序写入时这部分内容不被擦除掉，使其一直驻留在芯片中，所以称之为驻留子程序。当然这是个狭义的概念，从广义上讲，监控程序也是驻留在芯片中的，也可以称之为驻留子程序。

用户程序则是用户在上层利用图形化编程界面设计或者通过调用本系统提供的底层接口函数库用嵌入式 C 编写而成。这段程序是最终用户编程思想的体现，它是芯片中真正用于控制机器人行动的程序。

监控程序和驻留子程序都是为了最终的用户程序而服务的，前二者控制机器人的各种器官，最终的用户程序才是机器人的大脑，用于统筹调度各种器官功能，来完成机器人的各种活动。以下将具体介绍以上三部分程序的设计。

5.2 监控程序

监控程序是为了 实现系统平台的在线写入而驻留在芯片中的用于控制用户程序写入的程序。由于它的存在使得用户可以方便向机器人硬件平台主控芯片中下载编译

好的用户程序代码。这样就不用像传统的芯片下载程序方式那样，每次向芯片中下载程序时都要把芯片从硬件板上拔下来，下载完成后再插回去。这点对于本系统尤为重要，因为平台中选用贴片元件，难以进行芯片的插拔操作。下面将具体介绍监控程序的设计思想。

监控程序就是实现对系统运状态的监控，主要功能是实现在系统复位以后通过判断是否有新的用户程序要求写入，从而决定芯片将运行那一部分程序。以下将分三部分来介绍监控程序的设计，即主流程控制，用户程序数据的写入，中断向量内容的写入[23]。

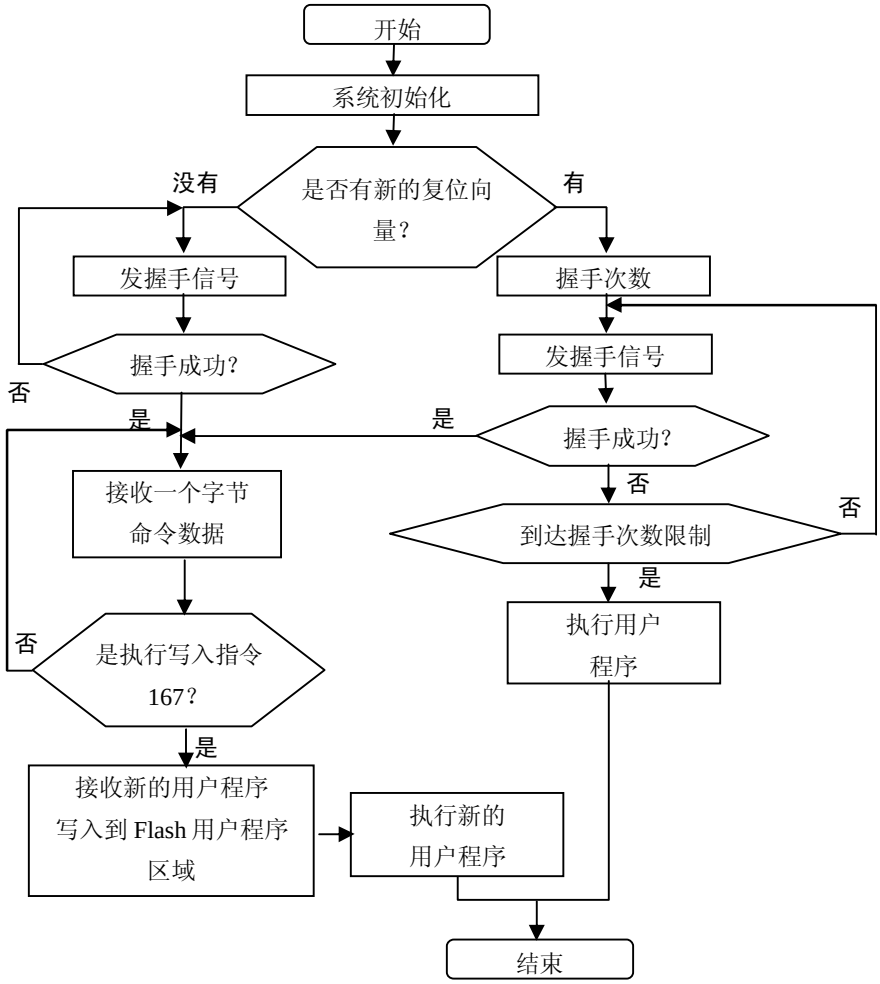


图 5-1 监控程序主流程图

5.2.1 主程序流程图

如图 5-1 所示为监控程序的执行主流程。对其描述如下：

(1) 系统初始化，其中包含了对堆栈的设置，晶振设置，串口设置以及 Flash 的初始设置。

```
;  ①  堆栈设置
   LDHX  #$107f      ;          重新定义栈顶
   TXS               ;SP<-(H:X-1)
;     ② 晶振设置，完全用外部晶振 4MHz.使用外部晶振，内部锁相环屏蔽
   LDA  #$22
   STA  ICGC2
   LDA  #$78
   STA  ICGC1
;  ③  串口初始化，波特率为 1200bps，发送和接收使能----
   ; baud=4MHz/16/BDH      1200bps=4MHz/16/B13
   LDA   #%00000000      ;          正常码输出、8 位数据、无校验等
   STA   SCI1C1
   ; baud = 19200    BDH=4MHZ/192/16=!13=$D
   LDA   #$00
   STA   SCI1BDH
   LDA   #$0D          ;          设定时，从“$D”改为“$C”
   STA   SCI1BDL
;  ④ 设 FLASH 时钟为 200kHz ---------
   LDX   #%00110000      ;          清 FLASH 状态寄存器
   STX   FSTAT           ;
   LDX   #initFCDIV
   STX   FCDIV           ;f(FCLK) = 192.308 kHz [4MHz xtal]
```

(2) 是否有用户程序的判断

此处的判断主要是通过判断用户程序复位向量区(UserReset，UserReset)是否$FFFF。复位向量区在写入器灌入监控程序时被擦除为$FFFF,当有用户程序写入时就被修改为用户程序入口的地址。因而通过此位的判断可以区分是否有用户程序被已经被写入到 Flash 中。

(3) 无用户程序的处理

当芯片中还没有用户程序写入时，则一直发握手信号$49 主动与 PC 握手，直到握手成功。握手成功后，等待接收 PC 方发出的准备写入指令 167。然后与 PC 方配合将新的用户程序写入到芯片中。写入成功后运行新的用户程序。

```
MainLoop1:
   JSR   CommunicateTest    ;          调用“通信握手子程序”
   BCC  MainLoop1          ;C          =0 表明握手未成功，继续握手
   BRA   MainLoop2         ;C          =1 表明握手成功
   ;    握手正确的情况，接收命令及数据(进入监控状态)
MainLoop2:
   STA   SRS
   NOP
```

```
    ;    接收命令
    JSR   GetChar        ;              接收一字节
    CBEQA #!167,MainSub1   ;#!167,       执行写入
    BRA   MainLoop2
    ;       命令=167,执行写入
MainSub1:                  ;167,                执行写入
    JSR  ReceiveAndWrite    ;         接收程序、擦除、写入
RunAPP:
    ;    转入执行用户程序
    STA   SRS
    NOP
    LDA  #$77
    JSR   PutChar
    LDHX  #!0
h_c_loop3:
    STX   testRAM,x
    AIX   #!1
    CPX   #!15
    BLO   h_c_loop3       ;              重新赋值操作结束
    LDA   UserReset       ;              取用户复位矢量
    PSHA
    PULH
    LDX {UserReset+!1}
    JMP ,X                ;              转用户程序执行
```

在以上的程序代码中调用了子程序 ReceiveAndWrite，其主要的功能是将新的用户程序代码以及中断向量写入到 Flash 的用户程序区，这两项内容将在下面两节中具体介绍。

(4) 有用户程序的处理

芯片中有用户程序时，上电复位后，先发一段时间的握手信号$49 与 PC 握手通信(暂时设定为 50 次，可以调整)。如果在此段时间内和 PC 握手成功，则以下执行流程同无用户程序时握手成功后的操作。如果握手不成功，则自动转移到用户程序入口处执行。

```
        ;------       有用户程序情况----------
HaveNewReset:
    LDX #!50          ;              握手时间参数
HaveNewReset1:
    PSHX
    JSR CommunicateTest    ;         调用“通信握手子程序”
    PULX
```

```
BCS  MainLoop2          ;C              =1 表明握手成功，转
DBNZX  HaveNewReset1    ;              继续握手
;    无监控信号返回,转入用户程序执行
BRA  RunAPP
```

5.2.2 接收和写入数据区流程图

图 5-2 所示为接收和写入数据区的流程图。上层软件对机器代码分析后将其按照一页一页的方式发送至 MCU 端，即机器人硬件平台的主控芯片中。主控芯片根据接收到到的页号信息将该页所对应的区域先擦除掉然后写入相应的内容。写完后再将写入的内容发送至上层软件，以备比较写入的内容与待写入内容是否相符。如此往复直到将所有的用户代码全部写入到数据区内。

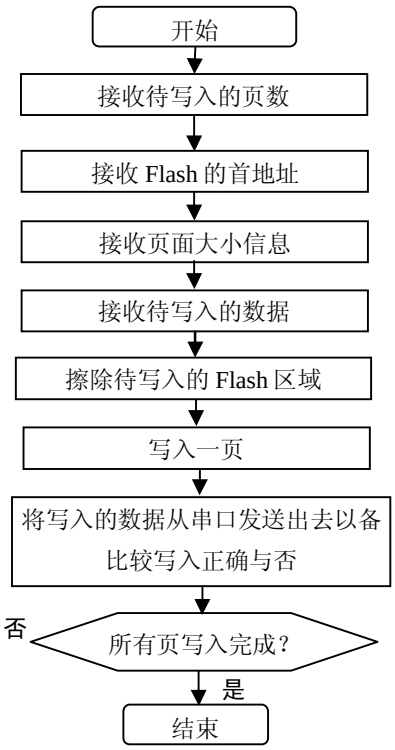


图 5-2 接收和写入数据流程图

5.2.3 接收和写入 Flash 向量区子程序

图 5-3 为接收和写入 Flash 向量区子程序的流程图。对于中断向量区内容的写入和上述的程序数据代码区的写入大致相同，但必须考虑一个复位向量的问题。因为在系统复位后必须从监控程序开始执行，而一旦将用户程序的 入口地址写入到芯片的复位向量区，则下次复位后将会直接运行用户程序，不再运行监控程序，如此执行则监控程序写入后只能写入一次用户程序，这一点与在线写入的思想相违背。为此在设计时采用了复位向量重定位的方式，即先将用户程序的复位向量放到另外开辟的一个 复位向量缓存中，写入用户程序时，仍然将监控程序的首地址写入到芯片的复位向量位置。这样

图 5-3 接收和写入中断向量区流程图

在系统复位后仍然从监控程序处开始执行，系统检测没有新的用户程序要求写入时转到复位向量缓存地址所对应的用户程序入口处执行。如此即可解决复位向量问题。

5.3 驻留子程序

在SD-HCS08-Robot开发平台在设计初期发现系统软件在编译和写入时均要花费大量的时间，一个长一点的程序通常要花上一分多钟的时间来完成这两项的操作。经过分析发现，在系统中有很多常用的子程序在每次编译和写入时都要重复的写入，而且占去了大量的时间。为此设计的后期调整了设计思路将一部分常用的子程序先编译好一次性的写入到芯片中，并驻留在芯片中，这样执行的效率得到了大大的提高，本来写入和编译的时间要近一分钟的时间，现在只要十几秒钟甚至几秒即可完成。同时上层软件也得到了极大的简化，屏蔽掉了很多本来就应该最终用户不可见的内容，使得他们在使用时不必纠缠于一些细节，也基本上感觉不到由于编译和写入延时带来的不舒服，提高编程的效率。以下将具体介绍目前SD-HCS08-Robot开发平台底层软件驻留在芯片中的子程序的设计。

5.3.1 子程序设计原则

在子程序的设计中应该遵循两个原则，一是降低子程序之间的耦合性，二是尽量少占用系统资源。

降低子程序之间的耦合性是为了使各子程序相对独立，这样设计的好处是能够使子程序具有更强的通用性，这样也为以后的程序开发做了积累。低耦合性的了另一个好处是可以使得其中某一子程序的修改不会影响其他子程序的功能。这一点也尤为重要，耦合性太强的程序很难修改，而且可读性较差，可以说是比较失败的程序。降低程序间的耦合性一方面是要使子程序的功能相对单一，不要把很多功能糅合到一个子程序中，这样子程序设计时也比较简单，由于子程序的结果而引起的操作尽量用参数在出口处带出去，留给调用该子程序的上层程序来解决。降低程序间的耦合性另一方面则是在设计时少用全局变量，此处的全局变量是指在芯片的内存区开设的固定变量，此固定变量既被该子程序调用，在其他子程序中也被用到。当这种变量使用很多时，程序间的关系很复杂，很难理清楚，看起来显得很混乱，以后修改也很困难。

减少子程序对系统资源的占用主要是减少对内存资源和Flash空间资源的占用。

在子程序设计过程中不可避免的会用到很多变量，而这些变量很多情况下是仅仅在在程序内部用到的，即在高级语言中所提到的临时变量的概念。这种变量如果都开设在内存区将会极大地占用内存资源，因而设计时应当考虑将此类的变量放到堆栈中去，这样在子程序调用时，从堆栈中开辟一块空间出来作为临时变量区，子程序退出时再将这块开辟的区域释放出来，这样就不会占用很多内存资源了。另一方面是减少对 Flash 资源的占用。Flash 资源是用于存放程序代码的，每种芯片这项资源都是有限的，而驻留在芯片中的子程序是一直保留的，减少驻留程序对 Flash 资源的占用就可以给用户程序留下更多的空间，如此用户程序就可以写的相对较大了。减少对此项资源的占用最有效的办法是对程序进行优化，同样功能的程序，优化后和优化前的代码量差别很大。优化工作就是对程序中的语句进行斟酌，发现是否有更简单的方法，更有效的指令，以及更合理的结构。在此要着重提出的是尽量能使用汇编语言来写这部分的代码，这样可以大大的减少代码量。

5.3.2 参数传递和调用方式

由于这段驻留的程序是留给上层软件来调用的，而上层软件设计是基于嵌入式 C 来进行的，所以在此必须解决程序的调用方式和参数传递的问题。以下将介绍针对这两个问题在子程序设计时所做的处理。

在系统资源的内存区中预留了一段空间$0090~$009F 作为系统保留空间，用户程序不能直接操作，即把用户可用的内存区设置从$00A0 开始。预留的这段内存区用来解决参数传递问题，由于嵌入式 C 中没有直接针对寄存器操作的指令，但可以操作具体的物理地址。将所有子程序的入口参数和出口参数均定在$0090，$0091，$0092，程序开始时将此三个变量分别放入寄存器 A，H，X，程序退出前再将 A，H，X 再回送到这三个地址中。上层调用时只要先将变量放入这三个物理地址中，调用完毕再从这三个地址中读取数据。当参数个数超过 2 个以上的参数时，参数就从 HX 传递过去，即实际参数存放在以 HX 为入口的 N 个内存区域，N 为参数的个数。

对于上层调用某子程序时必须能明确的知道该程序的入口地址，而当某子程序作修改后重新编译，则所有的子程序均有可能要调整，这样对于上层的调用很不方便。为此在设计过程中，对某个子程序设计时先将其程序代码编译，然后根据得到的机器代码大小给其分配一个适当大小的空间(比其实际的代码量稍大)，并将其位置固定。

这样如果以后对该子程序修改编译后就不会影响到其他子程序了。

5.3.3 子程序设计

经过分析主要有以下的子程序：AD 采样子程序，串行通信子程序，延时子程序，液晶显示子程序，PWM 输出子程序，直流电机方向控制子程序，蜂鸣器子程序，通用 IO 读写函数等八个通用的子程序。在设计时将给它们分配好各自具体的地址空间，以它们对应的程序入口地址作为调用时的函数名。

(1) AD 采样子程序

AD 采样子程序主要是用在对外部传感器数据的采样上，主控芯片中提供了一个 8 通道 10 位 AD 采样模块。为了能够更准确的采集到外部的数据，同时也给上层提供方便的调用方式，在设计时，共使用了四个子程序来协作完成，具体介绍如下：

单通道基本 AD 采集函数：AD10Z。此函数用于对单路的某通道进行最简单的采样。这个函数的返回值就是采样时刻 AD 通道上的原始数据。

8 通道 16 次均值采样函数：AD8Lu。这个函数利用上一个基本函数 AD10Z 来通过求平均的方式来得到 8 个通道经过 16 次累加求平均后的采样值。函数返回值就是 8 个通道上各自的值。

16 次均值后单通道的值：AD10Z1。此函数调用 AD8Lu，结合入口的通道号给出 16 次采样后某一通道上的值。

(2) 串行通信子程序

串行通信子程序是用来实现主控芯片通过串行口和外界通信的。为了使双方在通信时不致于死等造成死机，因此在主控芯片接收数据时增加了一个延时等待时间参数，在规定时间内收不到数据即视为通信失败，不再继续等待。设计中共使用了四个函数，即接收一个字节函数 H08SCI1Rec1，接收 N 个字节函数 H08SCI1RecN，发送 1 个字节函数 H08SCI1Send1,发送 N 个字节函数 H08SCI1SendN。以上函数功能很明确，在此不再一一赘述。

(3) 延时子程序

延时子程序在系统中使用较多，不仅上层要调用，底层函数也可能会使用到。设计时，做了两个不同数量级的延时函数，具体介绍如下：

周期级延时函数：Delay_Nus。其延时的时间为 12+8A 个指令周期，A 为入口参

数，取值范围 0~255。

毫秒级(4000C)延时函数：Delay_ms。其延时时间为 HX*4000 个指令周期，HX 为入口参数，取值范围为 0~65535。若指令周期为 0.25 微秒，延时一个单位时间为 1 毫秒。

(4) 液晶显示子程序

液晶显示是为了将系统运行中的某些数据量直观的反映出来。在目前的系统中只提供了英文显示的功能，但已经将点阵字符显示的接口留出，随时可以添加。在实现英文显示时共设计了三个函数来协作工作，具体介绍如下。

液晶执行指令函数：LCDCommand。其功能是将入口给出的指令送到 LCD 进行命令执行或字符显示。

液晶初始化指令：INIT_ELCD。功能是初始化 LCD 的主控芯片 HD44780[22]。

液晶显示指令：Show_ELCD_Buffer。其功能是将入口 HX 对应的 16 个内存变量根据入口 A 中的参数显示在液晶的上行或下行上。

(5) PWM 输出子程序

PWM 功能在目前系统中实现三个功能：给直流电机调速，给步进电机打相位指令，另外是给蜂鸣器打声音脉冲使其发出不同的声音。具体实现时由两个函数来实现，具体介绍如下：

PWM 初始化函数：PWMInit。此函数用于实现 PWM 的初始化，包括边沿对齐方式和 PWM 的脉冲宽度。

PWM 配置函数：GB60PWMNConfig。此函数用于配置具体某一通道上 PWM 的占空比。

(6) 直流电机方向控制子程序

直流电机控制子程序主要是 通过控制直流电机上两根控制线上电压的高低来完成对电流转动方向的控制。通过两个固定于机器人硬件平台上两边的轮子配合工作来完成机器人的行进动作。具体实现时由直流电机方向控制函数 DCMotor 来完成，其入口参数为电机号和运转的模式，用以实现对单个电机的控制[24]。

(7) 蜂鸣器子程序

蜂鸣器子程序是用于控制蜂鸣器的鸣叫的次数。具体实现函数为 Beep_Bell，入

口参数为 A，表示鸣叫次数，取值范围为 0 到 255。当 A=0 时蜂鸣器不鸣叫。

(8) 通用 IO 读写子程序

通用 IO 读写子程序用于控制系统留出的通用 IO 口的读写操作，设计过程中将没有使用的IO口全部留出来做为通用的IO。具体实现时用2个函数分别完成，即IORead和 IOWrite。IORead 函数用于读选中 IO 通道上的电平值，入口参数为 A，取值范围为 0~8,用于选取系 IO 通道号；出口参数为 A，取值范围为 0 或 1，用于表示选中通道上的电平值。IOWrite 函数用于向选中的通道上写入指定的电平值，入口参数为 A 和 X，A 表示通道号，取值范围 0~8，X 表示写入的电平值，取值范围为 0 或 1；无返回参数。

以上介绍了在内存区中驻留的子程序，其使用的空间是$2000~$2780，而给驻留子程序分配的空间是$2000~$4FFF，所以目前仅使用了分配空间的 1/6，还有很大的可扩充性。

5.4 用户程序

用户程序是指用户通过上层的图形化设计界面或类 C 设计界面设计而成的程序，它编译后通过串行线下载到硬件平台的主控芯片中。这部分程序是主控芯片中的核心程序，它决定了机器人实际的运行方式，是对机器人的直接控制。它一方面可以由用户根据实际情况，从元件库中选择相应的控制元件和执行元件共同搭建而成，另一方面也用户由可以根据系统中提供的系统函数库和编成规范直接编写 C 语言程序来完成控制。关于如何设计用户程序将在 PC 方软件一章中将会给出详细介绍，在此不再赘述。

传感器是机器人感知外界环境的唯一途径，用户程序的设计过程其实就是对传感器的应用过程[25]。用户程序设计的好坏一方面取决于用户设计程序的技巧，另一方面取决于用户对传感器知识的了解[26]。为此在这一章中来介绍一下设计机器人应用程序时可能用到的各种传感器。传感器根据获得的外界信息方式不同分为两大类，即模拟量传感器和开关量传感器。模拟量传感器是指获得的数据是一个连续变化的值的传感器；而开关量传感器是指获得的数据只有真假两个量传感器。但是这二者之间的界限也是很模糊的。比如说灰度传感器，它本身是一个模拟量传感器，它可以测量出投射在反射面上光线的强度。白色物体反射强度最好，因而转化后的模拟量也就最大；黑

色物体吸收能力最好，因而放回值最小。为此可以将灰度传感器用于循线，寻找铺设在地面上的色带，有色带处计为真，无色带处计为假，此时也可以说它是一个开关量传感器。下面将介绍一些常用的其他传感器的知识。

(1) 接触传感器，压力传感器

这两个传感器是用于测试机器人与外界的接触关系[26]。接触传感器传递给机器人的信息是是否有物体与机器人直接接触，它只有真假两种状态，因而是个标准的开关量传感器。压力传感器可以反映出机器人和物体的接触程度，即压力大小，压力越大得到的输出模拟量越大。因而压力传感器可以用于机器人手掌的设计，用来判断机器人在取物时使用了多大的力。

(2) 红外传感器

这个传感器是用户感知肉眼看不到的红外信息。一方面它可以做成开关量传感器用于探测机器人周围是否有红外的光源，另一方面也可以用于红外测距或者是红外测障，作此用途时要用一对发射管和接收管来实现，二者操作的红外频率必须相同[27]。

(3) 声音传感器

这一传感器主要是用于测试外界的声音大小的，最常见的是噪音传感器。除此之外还有专门录制动物语音的传感器。

(4) 超声波传感器

它主要是用于探测外界的超声波[28]。由于自然界中的超声波很少，因而使用超声波传感器也是成对使用的，一个用于发，一个用于收，根据发射和收到的时间差可以估算出机器人与反射源之间的距离，因而可以用于测距和避障操作[29]。

(5) 方位传感器

此传感器主要用户探测机器人所在的方位，使机器人具有方向感。使用时可以根据机器人正对的方位和南北方向偏移来计算出机器人的方位，从而决定下一步要执行的操作。

(6) 加速度传感器

加速器传感器用于判断行进中物体的加速度。这个传感器主要用于测试快速运动的物体的加速特性和减速特性。

(7) 气体传感器

这个传感器可以判别出机器人所在环境的有毒气体成分。由于该传感器的探头技术各不相同，因而气体传感器能识别的气体的种类也各不相同，高级探头可以检测十几种，而普通的探头只能检测几种。

这些传感器就像人的眼耳口鼻一样把外界的信息传递给机器人[30]，以便机器人综合全面的信息作出反应提供条件。要根据实际应用的情况有选择性的使用传感器才会设计出一个合理的机器人运行程序[31]。

5.5 嵌入式软件编程规范总结与体会

经过这段时间毕业设计工作以及三年的研究生学习，本人对嵌入式软件编程规范有了一些感性的认识，在此作以下总结。

5.5.1 注释

注释是软件编程设计的一个重要组成部分，一个注释清晰的程序可以使得一个原本对此段程序不了解的人员很快的了解程序的功能和结构。这一点对于程序设计者本身也是很重要的，一个大的软件设计周期可能很长，当设计者在设计后期再回头看最初设计的东西时，如果注释不清楚则仍然要花费很多时间去重新来了解自己编写的程序。当然注释内容也不是越多越好，注释太多一方面使得程序太长显得很拖沓冗长，另一方面也使得程序结构不适很清晰。对于嵌入式的底层软件程序编程注释应该做到以下两点：

(1) 程序头的注释

程序头的注释中应该能够包括以下内容：程序名，功能描述，入口参数，出口参数，占用堆栈空间大小，占用的系统寄存器，内部调用的函数，设计人员，编程时间，备注，调用实例。有了这些内容，调用此段程序时，不必详细了解程序的实现细节也可以放心的使用。

(2) 语句的注释

语句的注释包括两个内容，其一是单条语句的注释，另一种是整段功能的注释。对于单条语句注释时要力求简单，对于那些一目了然的语句就没有必要加注释了。对于整段的注释应该添加在这段语句之前，如果在程序中前后有顺序关系，最好能加上

标号，这样会使得程序看起来结构更清晰。对于那些已经在整段注释中描述过的内容就不要再出现在单条语句中了。

注释在编程时固然比较麻烦，但是对于后期设计，以及加强程序的可读性，提高自己的编程能力都是很有好处的，在编程过程中应该得到足够的重视。

5.5.2 命名规则

一个好的命名规则可以使得程序中所有的程序名，变量名显得很有意义，从命名的方式中也可以看出编程功底。命名中强调两点：一是命名中尽量避免中英文的混合命名，二是子程序中标号命名时将子程序名带入其中，这样可以避免标号的重复。

5.5.3 程序分割

对于一个大的软件开发，不同的功能分割可以使得实现时的难度差别很大。这种实现的难度在设计初期表现并不明显，到了后期这种由于设计时功能划分的混乱造成的问题会越来越多地暴露，此时不断地修改，不断地调整，可能最终还是无济于事，导致最终的失败。因而在设计的初期就要对整体做好把握，各部分功能进行合理的切割，定好明确的出口入口，如此才能保证最终得到一个完美的软件产品，即使某一方面出了问题，也可以在其内部解决掉而不要牵扯到其他的内容。

5.5.4 程序测试

程序测试是软件设计的一个 重要的环节。嵌入式系统的设计中应该包括两层测试：首先是各模块本身的测试，在各部分程序都设计完成最终进行整合前，各部分内容应该是得到了充分的测试，并保证没有问题，如果出现问题仍然能够回到该模块中重新测试，迅速确定是否的确是模块内部有问题；另一项就是系统整合后的测试，测试软件并不一定是最终的交付给用户的程序，而仅仅是用于测试各功能模块协作工作的稳定性，健壮性等。当然最终交付的程序当然也要进行测试，这一点是所有设计者都知道的。对于前期的测试也是不容忽视的，前期进行充分的测试可以大大减少后期综合测试的出错可能性，而且后期测试相对也比较麻烦，尽量不要把本该前期所做的工作带到后面，这样会成倍的增加工作量。

5.5.5 版本控制

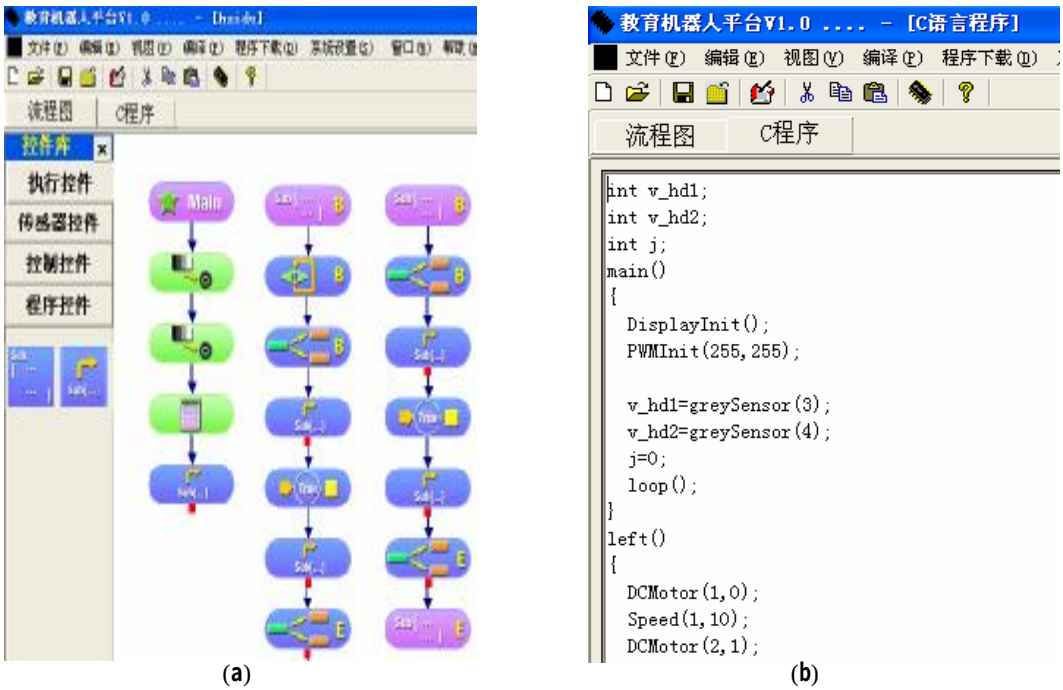
之所以要提到版本控制问题是强调一下不要因为版本的混乱使得做好好的程序被冲掉，使得很多工作都白白浪费掉。在设计过程中一般建议使用三个目录：当前工作目录，历史目录，临时工作目录。历史目录中保留近三天到一周的工作内容，更长时间的可以一周或半个月保存一个版本；当前工作目录中保存着正在修改的工作，但是并不在此目录下进行修改，真正要操作时把此部分内容拷贝到临时工作目录中，在那里面进行操作，操作完成后将其拷贝回当前工作目录，并及时的把当前工作目录内容备份到历史目录中。备份工作时在文件夹上加上日期以防混淆。养成这样良好的习惯，对于编程工作还是很有益处的。

以上即是作者对嵌入式软件编程规范的一些体会。

第六章 PC 方软件设计

6.1 功能概述与系统结构

PC 方软件是系统平台中提供给最终用户设计机器人控制程序的高端软件，其中包含了一个图形化设计界面和类 C 设计界面[32],如图 6-1 所示。图形化的设计界面中提供了一个使用了结构化的程序设计思想来设计机器人控制程序的设计平台。设计时用户不需要直接编写程序代码，只要将编程的思路用结构化的流程图搭建起来就可以了，软件平台会自动将流程图转化为 C 代码。对 C 语言比较熟悉的用户也可以直接使用 C 来编程，设计时可以直接调用底层封装的驻留子程序来完成编程。



(a) (b)

图 6-1 PC 方软件界面

PC 方软件的实现思想是将所有的控制内容抽象为统一的控件对象，每一个控件对象中包含了与之相关的所有信息，包括控件对象的类型，在图形化设计界面中的位置，与控件对象相关联的父子关系，控件属性等。有了这些内容可以把图形化设计界面上设计的控制程序相关的图形和属性信息保存下来，下次再次打开时可以重新显示出来，同时也可以根据属性信息生成相应的类 C 代码用于编译和下载。以下将具体介绍 PC 方软件的实现。

6.2 图形化设计语言

图形化界面的设计是围绕着控件对象展开的。如图 6-2 所示，设计时将控件对象分为四大类：执行控件、传感器控件、控制控件和程序控件。执行控件包含了机器人的各种动作指令，包括直流电机和伺服电机的控制和调速、系统的延时、液晶显示以及变量的四则运算等。传感器空间中包含了机器人用于感知外界环境的各种传感器，包括红外传感器，灰度传感器及超声波传感器等。控制控件中包含了各种控制流程的操作指令，包括单分支、多分支、条件循环和计数循环。程序控件包含了对子程序的操作指令，包括子程序创建和调用。通过以上控件对象的组合就可以完成图形化程序设计。

图 6-2 控件库

6.2.1 控件对象数据结构

如表 6-1 所示为控件对象的数据结构。其中共包含了 8 个基本字段和四个综合字段。基本字段由一个简单的数据类型构成，各自表示一个特定的含义；综合字段由定长的字符串构成，通过对字符串的解析可以得到的字段信息。

表 6-1 控制对象数据结构

类型名：Icon_object		
字段名称	数据类型	说明
Index	Integer	控件对象序号
ControlType	Integer	控件对象类型
RectX	Singer	控件对象横坐标
RectY	Singer	控件对象纵坐标
ConnectVisble	Boolean	连接点是否可见
LineVisible	Boolean	连接线是否可见
Parents	Integer	父节点
Child	Integer	子节点
DataDefine	String*5	新变量声明
ExecSentence	String*100	C 执行代码
IconPara(15)	String*5	属性参数
PromptMsg	String*100	提示信息

(1) Index：控件对象序号

控件对象序号是标识动态生成的控件对象的唯一标识符。它在该控件对象被生成时创建，在该控件对象被卸载之前都是不变。它和控件对象数组下标是相同的。

(2) ControlType：控件类型

控件对象类型是用来标识动态生成的控件对象的类型。在现有的平台中共有 21 个控件对象类型。如表 6-2 所示为控件对象类型。执行控件和传感器控件仅对应一个控件类型，控制控件和程序控件每个对象包含了两个控件类型，其中双分支控件对象包含了三个控件类型。

表 6-2 控件对象类型

类型号	说明	类型号	说明
1	直流电机	12	双分支开始
2	伺服电机	13	双分支真结束
3	电机调速	14	双分支结束
4	延时	15	计数循环开始
5	四则运算	16	计数循环结束
6	液晶显示	17	条件循环开始
7	红外传感器	18	条件循环结束
8	灰度传感器	19	子程序调用
9	超声波传感器	20	子程序开始
10	单分支开始	21	子程序结束
11	单分支结束		

(3) RectX,RectY：控件对象位置

控件对象位置是标识控件对象在图形化设计界面中相对于左上角的位置。当在 PC 屏幕上显示时要做相应的转化。初始设置根据它的父节点所在的位置作设定，下移一个控件对象的位置。

(4) ConnectVisiable,LineVisble：连接线和连接点可见标识符

连接线和连接点是两个互斥的概念。当某控件对象连接点可见时，连接线不可见，此时可以在此空间对象连接点处增加新的控件对象；反之当连接线可见而连接点不可见时说明该控件对象已经连接了另外的一个控件对象，不可以再添加新的控件对象到其连接点处。ConnectVisiable 初始化为 True,LineVisble 初始化为 Flase。

(5) Parents,Child：控件对象的父子关系

控件对象的父子关系标识了该控件对象的连接关系，其中存放的是父子控件对象

的 Index 值。控件对象生成时父子关系值均初始化为-1。

(6) DataDefine：数据变量定义

数据变量定义字段定义了新的数据变量。当某个数据类型中包含了新定义的变量时，即可存放在该项内容。此项内容存放时已经将数据类型和数据变量一起放在该项内容中了。例如定义一个整型变量 J，DataDefine 内容为”Int J;”。

(7) ExecSentence：C 执行代码

C 执行代码是指根据控件对象类型和 IconPara(15)中定义的参数来共同生成的类 C 执行代码。将图形化设计界面中主程序开始的所有控件对象连接起来就是一个完整的类 C 代码。具体实现将在下面章节中介绍。

(8) IconPara(15)：属性参数

属性参数是指每个控件对象所包含的各参数的值。每个控件对象中可包含 16 个控制参数，每个参数可以有 5 个字节的字符串构成。通过对此属性参数的解析就可以复原出这个控件对象的所有内容。

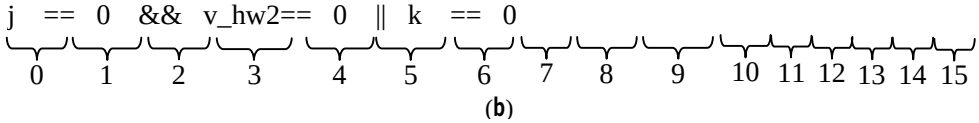
(9) PromptMsg：提示信息

提示信息是用于在某个控件对象被选中时显示其包含参数信息和控件类型信息的字符串。

6.2.2 控件对象属性的生成

控件对象属性参数存放在一个字符串数组中。由于每个控件对象所包含的参数个数不同，各个参数的信息也各不相同，在解析时要区别处理，为此为每一个控件对象

(a)

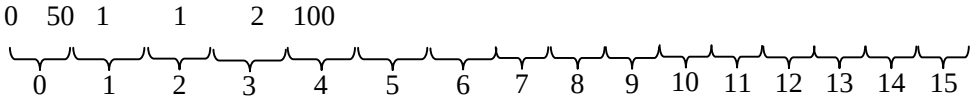


(b)

图 6-3 条件循环条件参数

都设计了一个独立的属性设置界面。如图 6-3 和 6-4 分别为条件循环条件参数和直流电机控制参数。

(a)



(b)

图 6-4 直流电机控制参数

图 6-3 所示的条件参数设置中使用了 10 个属性参数空间，从这个参数设置中很容易看出属性中包含的含义。图 6-4 所示的电机控制参数设置包含了 4 路电机的控制

表 6-3 控件对象参数列表

类型号	说明	参数个数	说明
1	直流电机	8	四个电机的运行模式和速度
2	伺服电机	1	伺服电机的转动角度
3	电机调速	1	电机的 PWM 值
4	延时	1	延时时间
5	四则运算	10	操作数和操作符
6	液晶显示	8	液晶提示信息
7	红外传感器	1	红外传感器变量名称
8	灰度传感器	1	灰度红感器变量名称
9	超声波传感器	1	超声波传感器变量名称
10	单分支开始	10	条件参数
12	双分支开始	10	条件参数
15	计数循环开始	1	循环次数
17	条件循环开始	10	条件参数
19	子程序调用	1	子程序名
20	子程序开始	1	子程序名

信息：每个电机由两个参数来设置，前一个是用来表示电机运行模式，电机有三种运转的模式，即正转(0)，停止(1)和反转(2)；后一个用来表示电机运转的速度，它是通过调节控制信号的高低电平的占空比来控制的,参数设置中的数据为低电平的百分比，所以该数值越小，电机转动越快。图中所示的 2,3 电机的运行模式均为 1，即为停止模式；电机 1 为正转方式速度为 50；电机 4 为反转方式，速度为 100，即控制信号全为低，所以电机也是停止状态。表 6-3 中给出了控件对象各自所包含的参数的个数以及参数的说明。

6.2.3 控件对象类 C 代码的生成

控件对象的类C代码在软件平台中是根据属性参数的信息和控件对象类型来设置的，它存放在 ExecSentence 字段中。由控件对象的类型来决定使用哪些通用的 C 语句，由属性参数的信息来设置动态生成的该控件对象的特定属性。以条件循环为例来介绍一下控件对象的类 C 代码的生成。如图 6-5 所示为 C 语言条件循环的结构体，其中

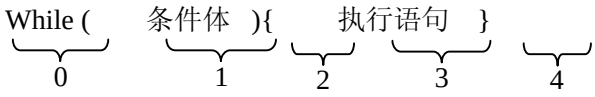


图 6-5 条件循环结构体

包括 5 个部分：0、2、4 部分为 C 语言关键字部分，它们在所有的程序中是不变的；1、3 部分为变化部分，它在各个不同的程序体中是不相同的。在此平台软件中将条件循环语句分为三部分来实现，即条件循环语句开始，条件循环语句体，条件循环语句结束。条件循环语句开始包含 0、1、2 部分，其中 0 和 2 部分为固定不变内容，可以直接生成，1 部分需要通过属性参数来设定；条件循环语句体是第 3 部分，它是在其他实际操作的控件对象中生成的；条件循环语句结束只包含第 4 部分，其中只有一个“}”字符，可以直接生成。所以当动态生成一个条件循环控件对象时将产生两个控件图标，一个为开始，另一个为结束，在开始后面的连接点可见，此时可以在其后面插入实际的语句体；右击开始图标就会弹出属性设置窗口，在其中可以设置结构体中第 2 部分的条件体。这样就完成了一个条件循环的设计。其他的控件对象也是按照此方法来设计的。当要生成总的类 C 程序代码时则要将主程序开始的所有控件对象类 C 代码根据各控件对象的父子关系连接起来，当然在此过程中为了程序的美观还要考虑缩进问题。

6.2.4 控件对象提示信息的生成

控件对象的提示信息是在生成 C 程序代码时同时生成的，因为它也要对属性参数进行解析才能得到具体的参数内容。这些提示信息是在图形化界面设计时鼠标移动到该控件对象上时作出提示的。

6.2.5 控件对象操作

以上介绍了控件对象节点内部的各个字段的设计方法，对于一个控件对象整体而言它包括了下面一些操作：创建，移动，删除和拷贝，存储。在PC 软件平台设计时，将所有的控件对象存储为一个动态的控件数组。但数组中前后的元素并没有关系，它们的连接关系由控件对象字段中的父子关系构成的双向链表来完成。由于并不是所有节点都是连接在一起的，所以动态的控件数组中可能不止包含了一个双向链表，这些链表的操作是相对对立的，彼此并没有影响。以下具体介绍对控件对象的操作。

(1) 创建

创建一个新的控件对象时必须先从控件库中选择一个控件类型，然后在图形化界面中创建一个此类型的对象，因而所有的控件对象都包含在控件库中。为了使开发平台的功能更强大，就必须不断地扩充控件库。

(2) 连接

当两个节点靠近时(移动节点为从节点，固定节点为主节点)，此时就要判断是否将两个节点连接起来。有两个条件来决定是否连接：一是是否进入连接有效区域，而是主节点的连接点标识 ConnectVisible 是否设为可见。当两个条件都满足时将主节点的 ConnectVisible 设为 False，LineVisble 设为 True。并将从节点设置为主节点的子节点，二者设定父子关系。

(3) 移动

移动是指用户可以直接拖动一个单独的节点在图形化界面中移动。如果节点和其他的节点相连的话，则将其作为一个整体一起移动。如果节点有父节点，拖动时其父节点不移动，而其子节点和它一起移动。当拖动到相应位置并松下鼠标左键时，还要判断是否拖动到了某个连接点的有效区域内，如果是，并且这个连接点有子节点，还要同时移动子节点的位置。

实现方法：

① 移动节点的方法。首先根据引起移动的那个节点的原来的坐标和新的坐标，

计算出它在 X 轴和 Y 轴方向所移动的位移量；然后根据此位移量，移动需要移动的所有节点。

② 普通节点和特殊节点。考虑到流程图的单链结构，为了便于处理，把单分支、双分支、计数循环、条件循环等的开始到结束部分作为特殊节点考虑；其他节点作为普通节点。

③ 对特殊节点的不同处理。如果拖动的是特殊节点，并且有子节点，则只移动特殊节点，然后其子节点上移到特殊节点的位置处。

(4) 删除

删除即是删除不需要的节点。通常的方法是把所要删除的控件节点的父子节点的父子关系修改即可完成。但是这样做会使动态控件对象节点数组变的越来越庞大。因而在设计时采用另外一种手段，即首先将动态数组的最后一个节点拷贝到当前要删除的节点位置，然后卸载掉最后一个节点。这样动态数组中只包含了所有有用的节点，没有冗余。删除操作和移动操作一样也要考虑那些控制节点，由于它们不是独立的一个节点，所以删除时应该把与之相关的所有节点作为一个整体全部删除掉，删除时重复调用删除一个节点的方法将所有的内容都删除掉。

(5) 拷贝

拷贝操作是根据选中的控件对象的类型复制一个相同的控件对象，然后将原来的控件对象的属性参数拷贝到新生成控件对象中。当为特殊节点时也是把所有相关内容作拷贝。

(6) 存储

图形化程序最终存储为一个.dat 文件。它是一个纯二进制文件，其中存放了图形化界面设计的动态控件对象数组的所有元素。重新打开时再次将存储的内容恢复为动态控件对象数组，就可以在图形化界面中再次显示和编辑保存过的文件。

6.3 类 C 语言

SD-HCS08-Robot 开发平台最终用于编译器编译的是 Freescale 公司 08C 语言。它是一种嵌入式的 C 语言，直接使用它编程必须先了解嵌入式 C 的编程规范，嵌入式的 C 包含了很多对底层端口的控制，这些内容对于没有嵌入式开发经验的人来说理解起来较为困难。因此在平台 PC 方软件设计时在原有的嵌入式 C 语言的基础上做了一次封装产生了一个中间语言类 C 语言，它语法结构和标准 C 相同，但是它又可以

利用系统封装的底层函数对底层的端口和模块进行操作，因而使用起来比直接用嵌入式的 C 语言要简单，程序代码看起来也更加直观。

6.3.1 类 C 语言语法

类 C 语言的语法结构和标准 C 语法相同，其中的关键字，控制流程方式都是一样的。类 C 语言增加了一个控制函数库，此函数库中包含了控制底层端口和模块的操作函数。因此在使用此类 C 语言时基本语法的使用是比较简单的，关键问题是如何调用底层封装的驻留子程序。

6.3.2 底层接口函数访问

对函数访问应该应该了解两项内容，即函数名和接口参数。由于封装在底层的驻留子程序不是和上层 PC 软件同时设计和编译的，因而就不能够直接用设计时使用的函数名来访问，而只能访问函数在地址空间所在的绝对地址。由于上层的 C 不支持直接对累加寄存器 A 以及变址寄存器 HX 的访问，因而不可以用它们作为接口传递参数，而只能通过内存的绝对地址的访问来实现参数的传递。

(1) 内存地址访问

用汇编程序直接设计底层的控制程序时，一般会使用寄存器 A，HX 来传递参数，而嵌入式的 C 可以访问内存中的地址。为此在设计时将底层程序稍作调整，将内存区的前 16 个字节，$0090~$009F 保留下来不让用户直接使用，专门用来传递参数。$0090 对应寄存器 A，$0091 对应 H，$0092 对应 X，这样所有用 A,HX 传递参数的程序改为用$0090~$0092 的绝对地址来传递参数，进入程序后首先将相应的地址变量存储到对应的寄存器中，出口时作相反操作，其他部分则不需要改变。在 PC 方程序中用以下方式定义了 6 个变量用于函数调用时传递参数，para1~para3 为入口参数，rvalue1~rvalue3 为出口参数。定义时使用 volatile 是为了保证定义的参数随着对应的内存地址中的值同步变化。

```
#define para1     *(volatile unsigned char *)0x90
#define para2     *(volatile unsigned char *)0x91
#define para3     *(volatile unsigned char *)0x92
#define rvalue1      *(volatile unsigned char *)0x90
#define rvalue2   *(volatile unsigned char *)0x91
#define rvalue3   *(volatile unsigned char *)0x92
```

(2) 驻留子程序的调用

在底层共提供 7 个独立的功能子程序，在 PC 方软件平台中并不能直接调用，为了实现对这些内容的访问在 PC 方软件中采用了一个变通的方法，具体介绍如下：

假如底层定义的子程序 SubA 对应的存储地址为 AddressA，首先在 AddressA 处用以下方式定义一个子程序_SubA，_SubA 中并不做任何事情。

```
#pragma abs_address: AddressA
 void _SubA {}
#pragma end_abs_address
```

这样在调用_SubA 时实际上就是跳转到地址 AddressA 处执行，这和底层调用 SubA 是一样的。但是这样操作后虽然在子程序_SubA 的程序体中并不做任何操作，但是经过编译后仍然会产生一个子程序返回指令，如果这个指令被写到地址 AddressA，则跳转到该处执行时会立即返回，而不是执行子程序 SubA 代码。因而在实际将机器代码写入到芯片中时，并不把这部分定义内容编译后的代码写入芯片，这样用户程序写入以后地址 AddressA 处存放的仍然是 SubA 程序，在调用_SubA 时也是跳转到地址 AddressA 处执行，但是执行的却是 SubA 程序。这样就实现了 C 调用底层封装函数的目的。调用之前把所需的参数存放到入口参数 para1~para3 处即可，需要返回值时从出口参数 rvalue1~rvalue3 中取得。底层提供的七个控制函数如表 6-4 所示。

表 6-3 底层函数和 PC 方接口函数对应表

底层函数	存储地址入口	C 对应函数	说明
Delay()	0x2000	_Delay()	延时子程序
Sensor()	:0x2250	_Sensor()	传感器子程序
Display()	0x22C0	_Display()	显示子程序
DisplayInit()	:0x2300	_DisplayInit()	显示初始化子程序
PWMInit()	0x23A0	_PWMInit()	PWM 初始化子程序
Speed()	:0x2400	_Speed()	调速子程序
DCMotor()	:0x2650	_DCMotor()	直流电机控制子程序

(3) 子程序的二次封装

通过上一节的操作已经实现了在 C 语言中访问底层封装的操作函数，例如当要实现让电机 x 做 y 模式运转时，可以使用以下方式：

```
para1=x;
para3=y;
_DCMotor();
```

以上语句虽然能实现对电机的控制，但其代码书写形式和通常的 C 书写形式感

觉上总有点不同，特别是其参数传递方式。为此利用一个宏定义将以上三句话封装起来，实现如下：

```
/*1.直流电机*/
#define DCMotor(x,y)  para1=x-1;para3=y;_DCMotor()
```

这样在同样实现上述功能只用使用 DCMotor(x,y)一句代码就可以实现。对于没有返回值的底层程序都可以使用这种定义的方式来重新定义。在目前的软件平台中按照以上方式定义的子程序除了上面的例子还有四个：

```
/*2 PWM 初始化*/
#define PWMInit(x,y) para2=x;para3=y;_PWMInit()
/*3  伺服电机*/
#define ServerMotor(x,y)   para1=x+1;
                           para2=(PWMPulseCycle/100*(y/18+85))/256;
                           para3=(PWMPulseCycle/100*(y/18+85))%256;
                           _Speed()
/*4  直流电机调速*/
#define Speed(x,y)       para1=(x-1)/2;
                         para2=(PWMPulseCycle/100*y)/256;
                         para3=(PWMPulseCycle/100*y)%256;
                         _Speed()
/*5  延时*/
#define Delay(x)         para2=x/256;para3=x%256;_Delay()
```

对于那些有返回值的底层函数或者传递的参数在 2 个以上的就不好用宏定义的方式来封装了，必须用子程序来完成。例如对于液晶显示的控制操作，它必须传递 17 个参数，1 个为显示的上下行参数，16 个为要显示的字符，封装形式如下：

```
/*6LCD 输出*/
/6-1 显示上行*/
void LCDLine1(char *p,int i1,int i2)
{
  int i;
 for(i=0;i<16;i++)  LCDBuffer[i]=32;
  for(i=0;i<strlen(p)&&i<16;i++)
  LCDBuffer[i]=*(p+i);
  if (i1==1)
  { intTOstring(i2); }
  para1=0x80;
  para2=(unsigned char)(((unsigned int)(LCDBuffer))>>8);
  para3=(unsigned char)(((unsigned int)(LCDBuffer)));
  _Display();
```

```
}
```

除了以上例子还有四个子程序也是按以上方式封装的：

```
/6-2 显示下行*/
void LCDLine2(char *p,int i1,int i2)
/*7 红外传感器*/
 int infraredSensor(int x)
 {   para1=x;
    _Sensor();
    return (unsigned int)((rvalue2 << 8) | rvalue3);
 }
 /*8  灰度传感器*/
 int greySensor(int x)
 /*9  超声波传感器*/
 int ultraSonic(int x)
```

有了这些封装的函数就可以像常用的 C 语言设计一样来设计机器人的程序了。类 C 语言编译时要将这些子程序得封装文件包含到编译文件中。

6.4 编译和下载

编译和下载工作是程序设计的最后一项工作。当用户对编译器的编译方法比较熟悉时，用户可以自己制作编译文件，按照自己的要求进行编译。另外编译工作也可以把设计好的 C 程序拷贝到包含编译器的嵌入式 C 集成开发环境中自动编译，但是在使用集成开发环境之前要求用户必须了解集成开发环境的使用，完成编译的初始的设置。这两种编译方式对于本开发平台所面向的对象都是不适合的。最好的方式是平台中设计好编译的流程，用户通过一个按键就可以完成编译工作，繁琐的设置工作交给平台软件去完成。SD-HCS08-Robot 设计平台的编译工作正是基于这种考虑而设计的。

6.4.1 编译过程

在软件平台中完成机器人程序设计以后，保存图形化语言文件时便会自动生成嵌入式 C 的源代码。点击编译按钮就可以完成所有的编译工作，实际的编译过程包含了以下几个步骤：

(1) 编辑编译脚本文件：软件平台根据主控芯片设置(FLASH 地址、RAM 地址、堆栈指针)、源程序文件列表生成编译的脚本文件(.mak 文件)，该文件中包含了编译指令及相关的参数。

(2) C 程序向汇编程序的转化：启动 08C 编译器，根据.mak 文件的编译脚本将所

有.c 文件编译成.s 文件及.lst 文件。

(3) 汇编代码向机器码的转化：汇编程序将所有的汇编文件编译成.o 的中间目标文件，即所有的汇编语句都编译成机器码，此时机器码是分开存放的，没有统一连接在一起构成可执行的目标代码。

(4) 连接：启动连接器，连接器根据.mak 文件中的连接脚本，把所有的.o 文件统一连接成一个.s19 文件，该文件可以下载到目标机器上执行，同时还可根据用户需求生成.lst 文件和.mp 文件。

6.4.2 编译参数

在生成编译脚本文件时要使用到许多的编译参数，以下为在本平台软件设计编译命令时使用到的编译参数，下面将对这些参数作简要的注解[33]。

(1) –I：包含的头文件和连接文间所在目录。

(2) -g ： 带 Debug 调试信息 。

(3) -l ： 输出 lst 文件。

(4) -o ： 最终连接的主文件名。

(5) –c：包含的子文件所在的位置。

(6) -blit ： Flash 的起始地址。

(7) -bvregs ： 内存区的地址空间。

```
CC = E:\毕业设计\software\NewStyle3-19\bin\sdcc08
CFLAGS =  -IE:\ 毕业设计\software\NewStyle3-19\include\ -e -l
ASFLAGS = $(CFLAGS)
LFLAGS =  -LE:\ 毕业设计\software\NewStyle3-19\lib\
-blit:0x5000 -bvregs:0xA0.0x107F -dinit_sp:0x107F -fmots19
FILES = sub.o  4.o
4: $(FILES)
 $(CC) -o 4 $(LFLAGS) @4.lk
sub.o:
sub.o:
 $(CC) -c $(CFLAGS) E:\      毕业设计\software\NewStyle3-19\tempC\sub.c
4.o:
4.o:
      $(CC) -c $(CFLAGS) E:\毕业设计\software\NewStyle3-19\tempC\4.c
```

图 6-6 mak 文件

(8) -dinit_sp ：栈底地址。

(9) -fmots19 ：最终编译成为 motorola 的 s19 文件格式。

(10) -wf-xmem： 编译专用参数,使用间接寻址方式编译。

图 6-6 为一个 C 文件编译前由平台系统生成的.mak 文件，其中包含了一个 sub.c 的子文件和 4.c 的主文件。编译后输出 motorola 的 s19，同时产生 lst 文件和包含错误信息 err 文件。

6.4.3 编译设置

(1) 堆栈，Flash 空间，内存空间

主控芯片内存空间范围为 4KB 地址为$0080~$107F，前 32 个字节平台系统设计时保留，所以内存空间设置为$00A0~$107F，同时将堆栈空间的栈底设置为$107F；Flash 空间为 60KB ，地址空间为分为两段，$1080~$17FF 和$182C~$FFFF，前半部分已经用于存放了监控程序，后半部分的前端已经用于存放了驻留子程序，所以将用户使用的 Flash 空间定义从$5000 开始。

(2) 直接页与间接页设置

由于主控芯片的内存空间较大，因而在设置编译参数时为了能够访问到所有的内存空间，因而应该将编译参数设置为间接页编译。

6.4.4 机器代码下载

编译完成后就生成了可以写到芯片中的 S19 机器代码文件。目前平台上是使用串行口下载程序的。下载之前要对编译器编译出的 S19 文件做一番预处理。首先要将 S19 文件中的内容按照地址空间的大小进行排序。然后去处掉在$5000 以前的所有机器代码，因为在前一节中提到要求把定义的用于访问底层函数的子程序所编译出的代码在写入时去掉。接着根据最大地址和最小地址计算出待写入数据所占的页数。最后通过串口控件将机器代码写入到芯片中，实现对机器人的控制。

第七章 后续工作与总结

经过近一年的时间，终于完成了 SD-HCS08-Robot 教育机器人基础开发平台的所有设计，此开发平台已经可以完全实现最初的设计构想：提供了图形化的设计界面，开放的硬件环境，自由的机械结构。用户已经可以利用此平台进行各种机器人的设计，参加各种的比赛。

7.1 SD-HCS08-Robot 开发平台开发的关键技术

SD-HCS08-Robot 开发平台的开发过程中使用了两个关键性的技术：即底层软件的分块设计思想和 PC 方软件的图形化设计思想[34]。

7.1.1 底层软件的分块设计思想

底层 MCU 方软件设计时使用了分块设计思想，即把底层主控芯片中的程序分为三个相对独立部分分别设计，即监控程序、驻留子程序和用户程序。在设计主控制程序之前先为它做一个预处理，完成了功能性子程序的设计和写入，使得主程序开发变的相对简单，不必过多注意到内部细节的实现，从而也使设计主程序时更容易把握。这样一方面降低了设计和调试的难度，另一方面由于部分程序已经编译写入到芯片中，节约了上层软件编译的效率。这个设计思路对于设计类似的具有一定功能性要求的集成开发环境的项目有借鉴的价值。

7.1.2 图形化设计思想

图形化界面设计思想是指界面设计最终达到这样一个效果：这个图形化界面设计好以后，用户在设计程序时不需要写任何的一句程序代码，只要通过拖动图标的方式就可以完成程序的设计；程序的下载也不要用户过问，只要在设计完成后点击编译下载的图标就可以完成直接的下载，不用用户去做配置。这个设计思路对于开发中小学生使用的软件有一定的参考价值。由于在设计时关于图形化界面的设计是相对对立的，所以很容易移植到其他的应用系统中，为以后做类似的开发打下了基础。

7.2 SD-HCS08-Robot 开发平台开发的不足之处

但由于毕业设计的时间有限，因而在 SD-HCS08-Robot 开发平台设计过程中仍然有很多不尽完善的地方，还有很多对平台进行改进和功能完善的后继工作要完成，以

下将列出几点下一步将要完成的工作。

7.2.1 底层用户空间的利用

主控芯片 HC9S08GB60 的 Flash 空间为 60KB 字节，除去驻留子程序所占的 10KB 左右的空间，还有近 50KB 的空间。用户空间如此大，而一个用户程序通常比较小，远远不能放满全部的用户空间，这样就造成了用户空间的浪费，因此应当考虑在其中同时放入几个用户程序，让用户可以有选择的运行其中的某一段程序。如此一来，正常情况下向其中可以烧入不同的程序，以备选择；调试过程中可以把几个版本的程序都放进去，选择性的运行，这样可以使得用户方便的判断它们之间的运行差别，而此过程中则不需要每次都重新烧写程序，节约了调试的时间。这样就可以充分的利用这段 Flash 资源了。提高用户空间的具体方案的解决方案还有待进一步研究。

7.2.2 元件库的丰富

元件库是用于的搭建机器人的砖块，它决定了机器人最终能完成工作类别的多少。在目前的 SD-HCS08-Robot 开发平台只能完成一些简单的操作，例如执行基本的行进指令，采集外部温度、湿度情况，探测红外源等。在以后的工作中做一些比较高级的元件添加到其中，比如利用超声波技术探测外部的障碍物，利用语音芯片完成机器人说话，利用无线通信技术实现机器人程序的下载和机器人间的通信，利用图像处理技术增加机器人对于外界环境的识别能力。这些都是有待扩充的内容。

7.2.3 调试技术的完善

和其他的开发平台一样，目前的 SD-HCS08-Robot 开发平台缺乏 C 的调试环境，只能支持编译和下载，这使得用户在用 C 设计出现错误时只能人工的去判断出错的地方。然而本平台设计时所选用的主控芯片 MC9S08GB60 自带了 BDM 调试模块，我们只是在设计写入器的时候使用基本的写入指令，功能强大的调试功能并没有使用，为了使 SD-HCS08-Robot 开发平台变得更加完美，在后继的工作中，可以利用 BDM 模块完成调试环境的设计。

7.2.4 PC 方软件界面的美化

SD-HCS08-Robot 开发平台的 PC 方软件虽然已经按照预定的要求实现了图形化程序设计界面，但是其界面不是很美观：一是表现在其控件图标的直观表达效果不好，

控件图标不能随着控件对象属性的变化而变化；另一方面由于其采用了从上而下的直线型生成流程图的形式，所以在表达上不是很直观，因而在以后将对此作出修改，使用标准流程图的方式来显示图形化的语言。

7.2.5 C 程序与图形化语言程序的联动修改

在 SD-HCS08-Robot 开发平台中只能实现从图形语言到 C 语言的转化，而反过来则并没有实现。然而使用图形化的语言表达编程思想显得更加清晰直观，因而实现 C 程序与图形化语言程序的联动修改可以大大的提高编程效率。

7.2.6 智能化的分析

机器人教育其实也是人工智能在教育中的应用[35]，如何让机器人运行的更人性化，更智能化也是机器人教育平台设计的一个重要环节，虽然这部分内容应该由用户来考虑，但是如果能在设计时已经考虑体现了人工智能的思想，则在后期用户使用时则会更加方便。目前在 SD-HCS08-Robot 开发平台中尚未引入人工智能的思想，因而从智能化的角度去改善本开发平台的性能也将是下一步必须完成的工作。

致　谢

三年的时间转瞬即逝，在即将结束我学生生涯之际首先感谢我的导师王宜怀教授和他的夫人张建英老师。感谢你们在这三年中给我在学业上和生活中无微不至的关怀。王老师，你用渊博的专业知识将我领进科学的大门，你用严谨的教学态度教我如何做学问，你用豁达的处世方式让我明白如何待人接物。正是你的言传身教使我在人生路上得到了又一次的提高，这一切都使我受益匪浅，我也将在以后漫长的人生路上铭记在心。

感谢陆晓峰老师，你在电路方面丰富的知识使得我的硬件水平又上了一个台阶。

衷心感谢我的晓升师兄——刘老师！在我毕业设计的整个过程中都得到了晓升师兄极大的帮助。谢谢毕业了的大师兄阿蔡和戴晓静师姐，是你们在我入门时给了我的足够指导。感谢同级的汤龙梅和徐丽华，还有那帮可爱的师弟和师妹们，是你们使得我的研究生生活充满了快乐，特别感谢郭继伟小师弟在我毕业设计中对我的帮助，以及陈帅和徐清师弟帮我做论文的校对工作。

感谢我的父母，我的家人，是你们这么多年来一直为我提供了良好的家庭环境，毫无保留的物质支持和精神鼓励，使得我能够全身心的投入的学习中。谢谢阿猫，你在我研究生生活的最后阶段走进我的生活，使得我的学生生活更加的圆满。

感谢所有关心我、帮助我、鼓励我的老师、朋友和同学，我会用我的努力去回报大家对我的帮助，回报社会。

二零零五年四月二十日于苏州

蒋建武

参考文献

[1] (美)Saeed B. Niku.机器人学导论:分析、系统及应用 分析、系统及应用[M].电子工业出版社，2004.

[2] (美)Fred G. Martin.机器人探索:工程实践指南[M].电子工业出版社，2004.

[3] (USA)Bostock,Mike.EDUCATIONAL ROBOT[J].IEE Colloquium,1985,(48):p51–53.

[4] (JAN)Suzuki,H.Summary report of survey on the actual conditions of robot [J].education Robot,1995,(106):p 27-35.

[5] 彭绍东.信息技术教育的新发展: 机器人教育 [R].http: //hunandaxuesheng.blogchina.com/blog/article_67996.787509.html 2005.2.

[6] (USA)Murphy, R.R."Competing" for a robotics education [J]. IEEE Robotics & Automation Magazine June 2001,8(2):p 44-55.

[7] 教育部文件.教育部关于 2002 年普通高等学校招收保送生工作的通知[Z].教学[2002]6 号.

[8] 中央电化教育馆函件.关于举办第六届全国中小学电脑制作活动的通知[Z].教学馆[2005]13 号.

[9] (PL)Bak,R.Jedwabny,T.; Kasinski, A.; Majchrzak,J.; Niwczyk, G. Computer-aided task planning and control system for an educational robot ROBO L-2 [J]. Studia z Automatyki i Informatyki,1995,(2):p7-28.

[10] 教育部文件. 关于在中小学普及信息技术教育的通知[Z].教基[2000]33 号.

[11] SUNNY618 教育机器人的功能与特点[Z]. http://www.sunny618.com 2003.4.

[12] 陈晋龙，元魁等.一种智能教育机器人的研制 [J].《测试技术》2003,22(6):p42-45,p50.

[13] Wang, WeiZhuang,Yan;Yun, Wei-Min Innovative control education using a low cost intelligent robot platform[J].Robotica,2003,21(3):p283–288.

[14] 费仁元，张慧慧等.机器人机械设计和分析[M].北京工业大学出版社,1998.

[15] 马香峰.机器人机构学[M].机械工业出版社,1991.9.

[16] MC68HC908GP32 technical data[Z].www.motorola.com/semiconductors.

[17] MC9S08GB60.pdf[Z].Motorola Inc.,2003.6.

[18] L298[Z].STMicroelectronics Group of Companies,1999.

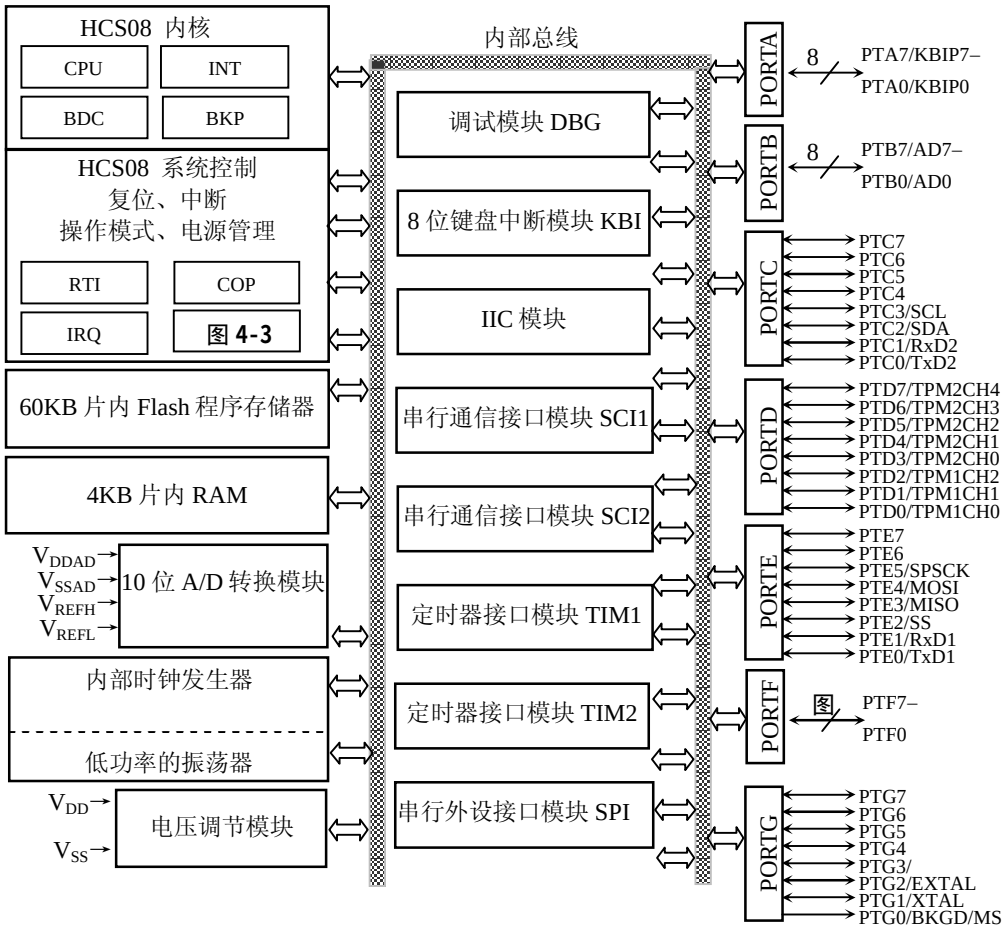
[19] SerroManual.pdf[Z].中鸣数码科技有限公司,2003.10.

[20] 15KV ESD-Protected +5V RS-232 Transcecivers[Z].MAXIM,1996.

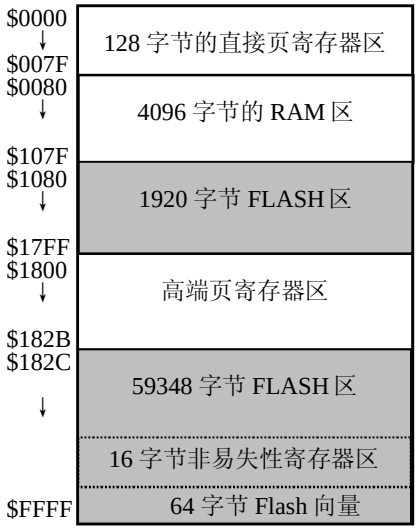
[21] HT1620[Z].Holtec Semiconductor Inc.,1999.7.
[22] HD44780[Z].南京国显电子公司,2003.
[23] 王宜怀.单片机原理及其嵌入式应用教程[M].北京希望电子出版社,2002.8.
[24] 严宏穗.ET-18 英雄机器人控制系统的研制[D].上海: 上海大学,2002.
[25] 杨东勇,蒋静坪 等.机器人分 布多传感 器系统的 设计[Z].《仪 器仪表 学报》,2002,23(3):p130-131.
[26] (美)迈克·普瑞德科.机器人控制器与程序设计[M].科学出版社,2004.5.
[27] 樊明轩.全区域覆盖移动机器人避障策略与多传感器系统的设计研究[D].南京: 南京理工大学, 2003.
[28] 10digital_gnomon.pdf[Z].上海广茂达, 2003.
[29] 彭志.机器人无碰撞轨迹规划[D].沈阳: 沈阳工业大学,2003.
[30] (日)城井田胜仁.机器人组装大全[M]. 科学出版社,2002.
[31] 罗志增, 蒋静坪等.机器人感觉与多信息融合[M].机械工业出版社,2002.
[32] (日) 西原主计,山藤和男等.机器人C语言机电一体化接口[M].科学出社,2002.
[33] iccmot.hlp[Z]. ImageCraft Creations Inc.,2000.
[34] (ISR)Verner,I.M. Waks, S. Educational features of robot contests the RoboCup-98 survey [J]. Advanced Robotics,2000,14(1):p 65-74.
[35] (美)Robin R. Murphy.人工智能机器人导论[M].电子工业出版社,2004.

附录 A　MC9S08GB60　相关资料

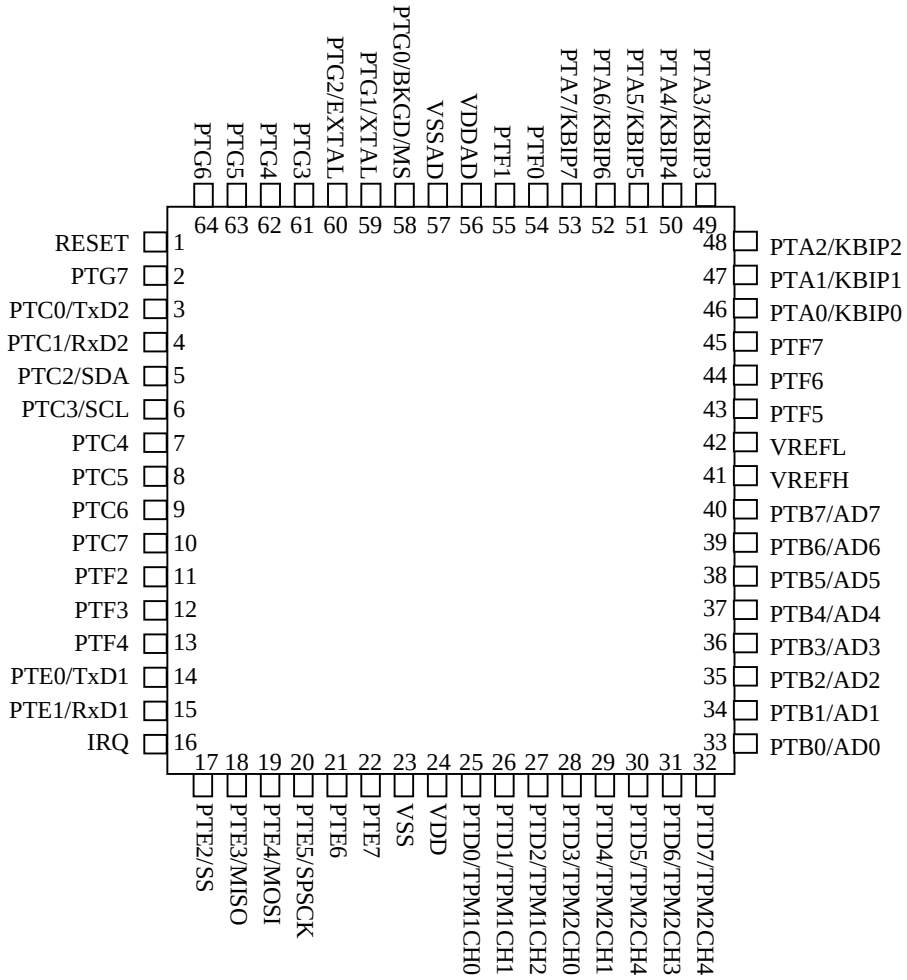
A.1 MC9S08GB60 结构框图



A.2 MC9S08GB60 存储器映像图



A.3 MC9S08GB60 管脚图



附录 B 控件对象节点结构体

```
Public Type Icon_object
   ContronlType   As Integer    '        控件类型
      RectX        As Single     '        控件横坐标
      RectY        As Single     '        控件纵坐标
      ConnectVisible  As Boolean   '     连接点是否可见
   LineVisible     As Boolean    '        连线是否可见
      Parents      As Integer    '        连线的出发点(父结点)，初始化时其值为-1，
                                          ‘当和某一点连接时，改变父子关系
      Child        As Integer    '        连线所指点(子结点)，初始化时其值为-1，
                                          ‘当和某一点连接时，改变父子关系
      DataDefine    As String * 50  '     设定变量的名称－－转换成 C 代码时，
                                          ’作数据声明用
   ExecSentence  As String * 100  '       执行语句
   IconPara(15)   As String * 5    '      属性设定的参数
      PromptMsg    As String * 80   '     提示信息(要初始化)
End Type
```

攻读学位期间公开发表的论文及参与的科研项目

[1] 蒋建武、王宜怀、章建民．MCU 串行异步通信的几种实现方法与编程实例[M]．军民两用技术与产品 2004 年第 1 期 总第 188 期。

[2] 参与《RFID 卡系列写入器的研制》项目，该项目已于 2004 年 12 月 17 日通过江苏省科学技术厅鉴定。

[3] 参与王宜怀、刘晓升编著的《嵌入式应用技术基础教程》第 16 章 08 系列编程器的开发的撰写，该书将于 2005 年 7 月在清华大学出版社出版。