Y1750246

西华大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：王春燕　　指导教师签名：杨孟泽

日期：2010.6.1　　日期　2010.6.1

西华大学学位论文版权使用授权书

本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，在校攻读学位期间论文工作的知识产权属于西华大学，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，允许论文被查阅和借阅，西华大学可以将本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、缩印或扫描等复印手段保存和汇编本学位论文。(保密的论文在解密后遵守此规定)

学位论文作者签名：王春燕　　指导教师签名：杨孟泽

日期：2010.6.1　　日期　2010.6.1

摘　要

为了提高晶振与时间继电器的测试效率，实现仪器的便携性与可程控，本设计研制了一种可对晶振频率和时间继电器的时间参数自动测量的仪器。可以同时接收八路输入信号，自动测试完毕后，在TFT液晶屏上实时显示出测量结果、绝对误差和相对误差。

本设计以Luminary公司Cortex-M3内核的单片机LM3S8962和Altera公司Cyclone II系列的FPGA芯片EP2C8Q208为核心构成硬件系统平台。采用高性能的Cortex-M3内核单片机，实现了智能化测量。系统的硬件设计包括单片机和FPGA外围电路设计、键盘与LCD接入设计、被测信号调理模块等。系统的软件设计有：用C语言实现单片机的控制，用混合描述方式实现FPGA内测试模块，用Lab Windows/CVI实现上位机软件界面的控制。测量和计算结果采用320*240液晶屏显示，界面友好，人机交互方便。

该作品做为一种智能程控仪器，可以进行本地控制和远程控制。远程控制时，通过仪器内部的通用接口模块，可以与实验室研制的TCP/IP、GPIB、USB等接口及接口驱动软件对接，组合为多接口兼容可程控仪器，由上位机发送SCPI指令完成测量过程，进而与计算机构成具备TCP/IP、GPIB、USB等接口的自动测试测试系统。

本论文对上述的各项工作进行了详尽的讨论，并以相应的实物模块为例，证明了该系统的可行性。参照国标要求，对继电器测试精度达到10^{-5}，对晶振的测试精度达到10^{-9}。突破了目前市场多种继电器测试仪，晶振测试仪，测试功能单一的特点。实验结果分析和测试应用的结果证明，该测试系统具有较小的测量不确定度、较高的工作效率。

关键词： 晶振；时间继电器；FPGA；LM3S8962；自动测试系统

Abstract

In order to enhance the test efficiency for oscillators and time relays,and make the tester programmable controlled and portable, the instrument used for testing frequency and Relay's time parameters has been designed. The tester equipped with eight channels for eight signals input simultaneously.When automatic test completed,the measurement results,absolute errors and relative errors are displayed on TFT liquid crystal panel in real time.

The hardware platform of this test instrument based on Luminary's LM3S8962 with Cortex-M3 core and Altera's Cyclone II series FPGA chips EP2C8Q208. Intelligent measurement is achieved since the use of the high-performance Cortex-core microcontroller.The hardware design includes microcontroller and FPGA external circuit design, the circuit of keyboard and LCD, the conditioning module for easured signal. System software design includes: useing C language to achieve MCU control,using hybrid description of FPGA to realize test module inside,using Lab Windows / CVI software interface to achieve the upper machine control. The Measured and calculated results are displayed on LCD liquid crystla panel with 320 * 240 pels, user-friendly interface to facilitate human-computer interaction.

As an intelligent programmable controlled tester,it can be local control and remote control. In remote control mode , through a common interface module designed in the instrument, combined whth TCP / IP, GPIB, USB interface and interface-driven software, a multi-interface-compatible device is integrated, host computer send SCPI commands to complete measurement process,Thus constitutes an automatic test system with computer and TCP / IP, GPIB, USB or other interfaces.

In this paper，the work of the above had a detail discussion，and using corresponding physical modules proved the feasibility of the system. Reference to national standard, Relay test accuracy achieved 10^{-5}, crystal achieved 10^{-9}. Breakthrough the character of a variety of relay tester, crystal tester only have single function in the current market.The experimental analysis and result of tests proved that this system offers smaller uncertainty of measurement and higher efficiency.

Key Word: Crystal;Time Relay;FPGA; LM3S8962; Automatic Test System

目　录

1 绪论

1.1 研究背景及意义

在国防航空、工业电子、科研计量等部门，常常需要了解所用的晶体振荡器的谐振频率、频率准确度等性能。采用传统的计数器或示波器等测试设备面临测试效率低、测试不方便的问题。对于低端谐振器、振荡器的测试，更是因为测试比较困难而无法得到测试确认，甚至会影响整套系统的性能。高稳定度晶振的准确度、老化等性能的测试校准，传统方法需要一套专门庞大的计量测试设备，成本高昂而且无法保障在实际应用中的性能和精度。

而在工业生产和制造业中常用的时间继电器如 JS14S、DH14S 等系列,随着使用中的磨损和老化，它的定时精度和可靠性都会降低。为了能够提前知道时间继电器的定时准确度，从而进行校正、修理或者在定时的时候把误差考虑进去，就需要定期的对时间继电器进行检定。在以往，时间继电器的检定装置主要是用电秒表。但是有几个方面的的原因限制了它的发展：一是有些电秒表使用的时间基准就是使用市电的 50HZ 作为时间基准，它的精度不高。二是它最多只有两个通道，效率不高，无法使用于大批量检定。三是它的操作不便，必须人工值守，无法实现测量的自动化。

通过参观成都某工业公司，发现他们的晶体振荡器测试技术还使用人工测试，工作效率低下，测试精度有随机性误差等弊端,鉴于这种现状，结合两种测量原理的相近之处，本设计研制了一台功能复用的高精度晶振与时间继电器测试仪，可以解决对可编程仪器标准命令 SCPI 的解析，和自动测试系统中可程控仪器通信协议生成，与国际仪器设计遵循的标准 SCPI 兼容，并可通过 TCP/IP、GPIB、USB 等接口与 PC 机构成自动测试系统，从而将计算机技术、软件技术、智能仪器、总线与接口技术等有机地结合在一起。

高速、高精确度、多参数、多功能的自动测试系统，是电子测量技术与自动控制和电子计算机技术密切结合的成果，是电子测量仪器数字化与数字信息系统相结合的产物，能避免了人为因素的误差，可获得十分良好的测试复现性；通过进行大量的冗余测量，进行判断、分析和折算，可以在很大程度上消除或削弱随机误差和系统误差，从而获得极高的测量精确度。

1.2 国内外发展现状

时间继电器属于低压电器控制类，其发展史可追溯到 70 年代，随着我国电子技术的不断发展和专用时间继电器芯片的研发和应用，使国内生产的时间继电器，从产品外观到性能上都得到了很大程度的发展。用户在使用时可通过面板外设的拨码或功能按键进行时间或控制方式的预置，从具体使用上有些产品基本上可与国外产品进行等同互换。然而在国内对这些大量应用的数显时间继电器的计时准确性的自动测试仪的研究却

没有得到同步发展，目前国内大多数使用单位仍然在大量采用传统的机械式或者是模拟式时间检测仪器来测量时间继电器的计时准确度，这种检测仪体积大、精度低、操作复杂对测试人员要求高。这严重的制约了智能时间继电器的推广应用，因而就迫切的需要研究一款体积小、重量轻、操作简便的数显时间继电器智能测试仪，能够同时对多款不同型号不同种类数显时间继电器的准确性进行快速检测。

晶振测试仪广泛应用于晶体行业、邮电、通信、广播电视、学校、研究所及工矿企业的科研和生产之中，但在 2004 年以前，晶振测试仪仍依赖于国外进口。目前市场上最精确的计时/计频仪器是产自美国福禄克的 PM6681R/PM6685R 计时/计频器.它提供了使所有的 PM6681R/PM6685R 的功能和稳定性精度都相当高的内置铷时基，可给出 10 位的可靠显示，价格高达 16 万人民币。国内市场上的晶振测试仪也都价格不菲，所以在提高精度、降底成本上下功夫是很有意义的。

总之，目前市场上涉及到测量精度的仪器，普遍成本较高，如果将晶体振荡器和时间继电器的测试利用一台仪器来完成，就可以很大程度上降底成本。本文提出的可程控的晶振与继电器高精度测试仪，不但是一台测试功能齐全的晶振测试仪，也可以仅通过软件菜单的选择，切换内部测试模式，形成一台高准确度的继电器测试仪，无需增添其他任何硬件设备。通过专利查询还没有发现相关的技术。

1.3 本课题研究的主要内容

本课题研究的晶体振荡器与时间继电器自动测试系统原理如下：由于在电子测量中频率信号抗干扰能力强，测量精度高，因而利用频率信号去测量时间，LM3S8962 完成控制、处理与显示，FPGA 完成分频、计数、闸门控制等模块。被测时间继电器的信号和基准频率信号在单片机的控制下进入 FPGA,由 FPGA 内部的闸门控制模块和计数模块完成继电器高脉冲下的时标计数，继电器吸合的同时 FPGA 内停止计数，然后单片机读回测量结果，并计算出设置值与测量值之间的绝对误差和相对误差，最后将测量和计算结果送到 LCD 上进行显示。

测量晶振精度时，外部基准频率源切换到铷标，此时基信号经过分频形成 1s 到 1000s 的精确闸门时间，这时做为计数脉冲的是被测晶振。其它原理与继电器的测试相同，在不增加成本的基础上实现了功能的扩展。

归纳起来本论文主要对以下几方面内容进行讨论：

- 研究了晶体振荡器与时间继电器自动测试系统的测量原理和结构；
- ARM、FPGA 硬件系统平台的设计，包括配置电路、LCD 电路、接口转换电路等一些外围电路的设计；
- 以单片机 LM3S8962 作为系统的主控部件实现了对系统的管理、控制和显示；
- 基于 Quartus II 和 VHDL 语言在 FPGA 芯片 Cyclone II 系列 EP2C8 上采用自上而下

的数字电子系统设计方法，实现了对频率测量的硬件及软件设计；

- 设计上位 PC 机虚拟仪器控制程序并将上位 PC 机虚拟仪器控制程序、接口转换卡、测时间继电器/测频的联调，实现系统的功能。

2 时间/频率测量方法

时间/频率的测量[1]是电子测量领域的最基本的测量之一，由于频率信号抗干扰能力强、易于传输、可以获得较高的测量精度，在数字化测量系统中对时间的测量通常都是转化为对频率的测量[2]，所以对频率测量方法[3]的研究越来越受到重视。下面将分别介绍测频领域几种常用的测频方式[4]，即直接测频法、间接测频法和等精度测频法。

2.1 直接测频法

直接测频法是根据频率的定义廷伸出来的一种测量方法，它的工作原理框图如图2.1所示。首先，被测信号通过信号调理，形成幅度一致，形状一致的计数脉冲，即时标。然后，将它加到闸门的一个输入端。由石英振荡器产生的振荡信号经放大整形，形成方波，在分频控制电路的控制下，形成不同周期的门控信号，加到闸门的另一个输入端。闸门由门控信号来控制其关闭时间,只有在闸门打开时间 T 内，通过计数器记录被测信号的脉冲个数 N。根据频率的定义计算出被测信号的频率为：

$$Fx = \frac{N}{T} \tag{2.1}$$

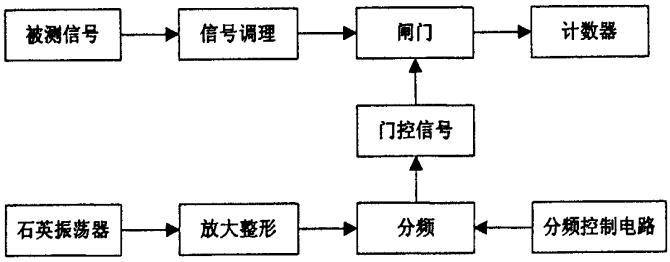


图 2.1 直接测频法原理图

Fig 2.1 Schematic diagram of the direct frequency measurement method

2.2 间接测频法

间接测频法[5]是通过测量被测信号的周期来计算频率的，即通过测量被测信号的周期 Tx（即被测信号两间隔脉冲间的时间），并由周期为频率的倒数来确定被测向信号的频率。其测量电路框图如图 2.2 所示。

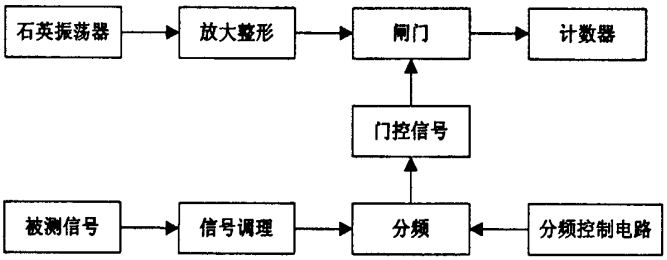


图2.2 间接测频法的原理图

Fig 2.2 Schematic diagram of the Indirect frequency measurement method

被测信号经信号调理电路变成方波信号后，经分频形成门控信号 Tx，控制闸门的开闭。当闸门打开时，周期为 To 的时基信号通过闸门送到计数器计数。闸门关闭时，停止计数。设记录下闸门由打开到关闭期间输出的计数脉冲 N，则有：

$$Tx=Nto \tag{2.2}$$

$$F_X=\frac{1}{T_X}=\frac{1}{NT_O}=\frac{F_O}{N} \tag{2.3}$$

2.3 等精度测量法

等精度测频方法[6]是在直接测频方法的基础上发展起来的。它的闸门时间不是固定的值，而是被测信号周期的整数倍，即与被测信号同步，因此，去除了对被测信号计数所产生±1个字误差，并且达到了在整个测试频段的等精度测量。等精度测频的最大特点就是在整个频率范围内都能达同样的测量精度，且与被测信号频率大小无关。原理框图如图2.3。

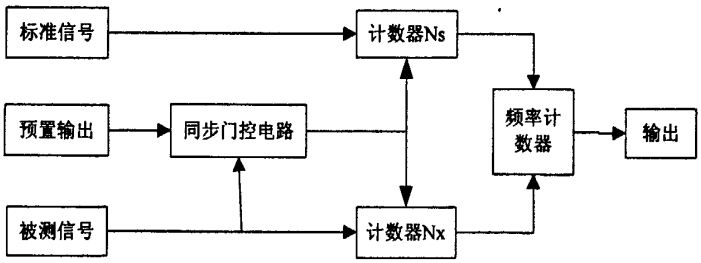


图 2.3 等精度测频法原理框图

Fig 2.3 Schematic diagram of the Precision frequency measurement method

在测量时间内，被测信号由计数器 Nx 计数，时基脉冲信号做为标准信号由计数器 Ns 计数。预置门的打开和关闭由被测信号和预置的测量时间控制，两个计数器的开关由控制预置输出的时间预置电路和同步门电路来完成，使闸门的开启和关闭和被测信号的有效跳变同步，从而得到完全相同的闸门开门时间。其原理波形如图 2.4 所示。

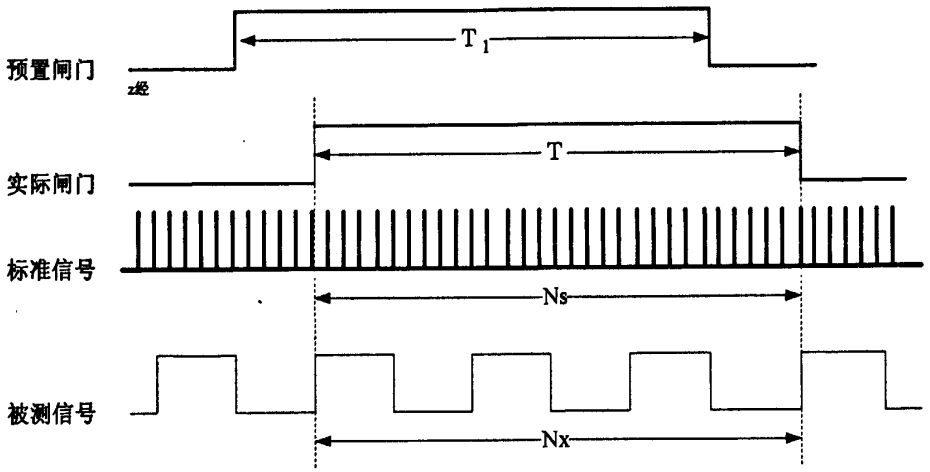


图 2.4 等精度测频原理波形图

Fig 2.4 The waveform of the Precision frequency measurement method

在测量过程中，两个计数器分别对标准信号和被测信号同时计数。首先给出闸门开启信号（预置闸门上升沿），此时计数器并不开始计数，而是等到被测信号的上升沿到来时，计数器才真正开始计数。然后预置闸门关闭信号（下降沿）到时，计数器并不立即停止计数，而是等到被测信号的上升沿到来时才结束计数，完成一次测量过程。可以看出，实际闸门时间 T 与预置闸门时间 T_1 并不严格相等，但差值不超过被测信号的一个周期。

2.4 测试仪设计方案论证

依据对前面介绍的时间/频率测量原理的理解，考虑到本测试仪要完成对晶振和继电器各自独立的测试，结合实验室的具体情况，最后定下的方案是：在共用软硬件的基础上，通过在 FPGA 内设置的一个选择接口电路，针对不同的测试功能，灵活地选择相对精确适用的测量方法。

具体做法是，对时间继电器测试时，考虑到它属于低频信号，故采用间接测频法，即测周期法。在继电器与主板之间，设计一个外围电路，实现的功能为：在继电器的定时时间间隔内，同继电器输入主板内测量模块的被测信号为高电平，定时时间到或不工作时，此被测信号变为低电平。也就是说，此时充当闸门信号的是继电器的高电平输出信号，充当时基脉冲的是恒温晶体振荡器(OCXO)的输出，它的标称频率为 5M，在系统开机 5min 后晶振频率准确度达 2.6×10^{-6}。

对晶振进行测试时，采用适用于高频段的测量方法，即直接测频法，为了尽可能保证闸门时间的精确度，基准源采用铷标，频率准确度为 5×10^{-10}，铷时标株子共振理论从本质上说，其稳定性是机电结构晶体振荡器的 100 倍，由它引进的量化误差（±1 误差）基本上可以忽略。基准源信号经过分频电路形成可以选择的不同时间间隔的闸门输

出，被测晶振经整形放大后充当计数脉冲，符合直接测频法的定义。

这样做的好处在于即节省了硬件资源，又保证了测量精度，仅通过软件上程序的执行就可方便地切换测试功能。对于前面提到的等精度恒误差测频原理由于是近几年才发展起来的所以没有在本设计中采用，但是如果要对系统的改进或升级则需要优先考虑这种方法，这也是本设计下一步需要改进的地方。

本测试仪[7]采用美国 Luminary 公司的 LM3S8962 作为系统的主要控制部件，实现对整个电路的测试信号控制、数据运算处理、键盘扫描和控制液晶显示器的显示输出等。以一块现场可编程逻辑器件 FPGA 芯片(CycloneII 系列的 EP2C8)，完成时基分频、时序逻辑控制、计数、输出等功能[8]。基于 QuartusII，用 VHDL 语言编程对 FPGA 进行设计、编译、调试、仿真和下载，实现了测试仪的模块化设计。这样相对于分离器件来说大大的缩小了体积、减轻了重量，提高了系统的集成度和可靠性[9]。在 ARM 控制下，当打开闸门时，被测器件时间继电器的信号和时间基准信号分别被送入各自计数器的输入端开始计数，当闸门信号关闭时计数器都同时停止计数，单片机将 FPGA 内的 12 位十进制计数器的计数值读入其内存进行处理后，并将计数结果送 LCD 显示[10]。通过对本地键盘或远地可程控面板操作，可以分别对时间继电器和时基信号计数器的开启、停止计数功能进行控制，也可以对各个计数器进行初始化。系统将单片机的控制灵活性及 FPGA 芯片(ACEX 1K30)的现场可编程性相结合，不但大大缩短了开发研制周期、降低了设计成本，而且使本系统具有结构紧凑、体积小、重量轻、可靠性好、精度高、易于升级等优点。

3 系统的硬件设计

3.1 硬件总体设计

整个系统的设计主要包括：单片机与 FPGA 组成的主机测试系统、键盘、液晶显示、通用接口模块、被测继电器组/晶振组以及上位机虚拟仪器控制模块等。总体框图如图 3.1 所示，其中单片机与 FPGA 组成的主机测试系统是本仪器的主体部分，键盘实现启动/停止测量、初始值设置、测时间继电器或者测晶振的功能选择，以及测时间继电器的时标设置和测晶振时的闸门时间设置等。测试仪虚拟显示控制面板可以进行本地和远程测试设置及显示，实现本地键盘所具有的一切控制功能，达到可程控智能化测试的目的。

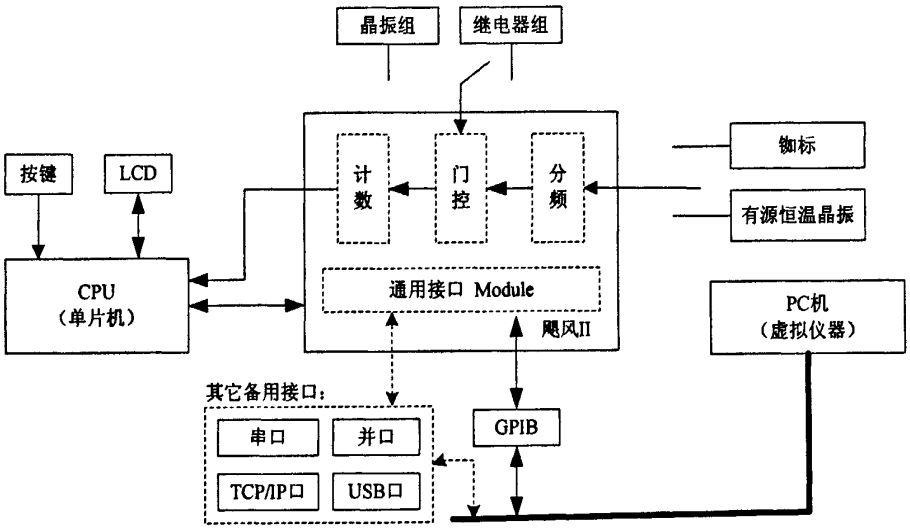


图 3.1 系统总体框图

Fig.3.1 The overall block diagram of system

系统复位后，各部分都处于准备工作状态。在启动信号到来后，被测时间继电器的信号和基准频率信号在单片机的控制信号的控制下进入 FPGA 中的计数器计数，计数结束后其结果送到缓冲器，当单片机接收到计数结束信号后到缓冲器取出数据进行处理后送到液晶显示器进行显示输出。键盘控制命令通过单片机的 GPIO 口读入单片机，实现启动/停止测量、初始值设置、测时间继电器或者测频选择功能以及测时间继电器的时标基设置和测频率时的闸门时间设置等。

3.2 单片机主控模块设计

3.2.1 LM3S8962 单片机介绍

本设计的 CPU 选用美国 Luminary 公司的 ARM Cortex-M3 内核芯片 LM3S8962。Cortex-M3 是首款基于 ARMv7-M 架构的处理器[11]，其包含的处理器基于 ARMv7 架构的三个分工明确的部分，A 部分面向复杂的尖端应用程序，用于运行开放式的复杂操作系统；R 部分针对实时系统；M 部分为成本控制和微控制器应用提供优化。是专门为了在微控制器，汽车车身系统，工业控制系统和无线网络等对功耗和成本敏感的嵌入式应用领域实现高系统性能而设计的，它大大简化了可编程的复杂性。

(1) Cortex-M3 内核介绍

ARM 内核的发展如图 3.2 所示，图中都是较为常见的 ARM 内核，目前最新的已经不是 Cortex 系列了，在过去的十年里，ARM7 系列处理器被广泛应用于众多领域。Cortex-M3 在 ARM7 的基础上开发成功，为基于 ARM7 处理器系统的升级开辟了道路。它的中心内核效率更高，编程模型更简单，它具有出色的确定中断行为，其集成外设以低成本提供了更强大的性能。下面我们重点介绍本项目中使用的 Cortex 内核。

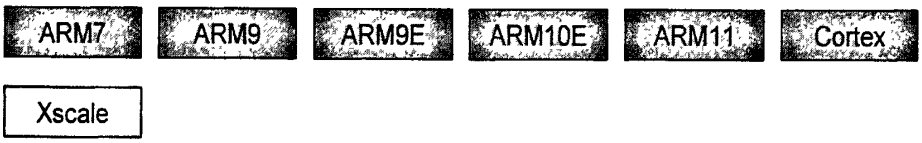


图 3.2 ARM 内核的发展

Fig.3.2 The development of ARM core

Cortex-M3 内核[12]的特点主要有：具有 ARMv7M 架构，在 ARMv4T 架构基础上拓展了 36 条新指令；具有更高的性能，Thumb-2 指令集架构（ISA）使其具有业界领先的代码密度；具有存储器保护单元（MPU），MPU 加强了优先权和访问规则，在多任务操作系统中通过分离代码、数据和堆栈来实现安全的优先级，同时通过控制存储地址的读、写和执行来实现访问的限制；具有标准化的存储器映射，高密度的内存使用，支持 8 位、16 位、32 位等非对齐访问和位操作（Bit-Bending）；支持最大 8 层的硬件中断嵌入，自动压栈和出栈，高优先级中断可以迟来中断，多挂起中断占先出栈，大大降低了中断延迟，并集成了嵌套向量中断控制器（NVIC）和内建系统时钟；增强了调试和跟踪的能力；更加节省了功耗，具有两种睡眠模式；能够更好地支持带有 CoreSight 以及其它系统特性的多核系统。Cortex-M3 内核比 ARMTDMI 有着更加优良的性能，具体比较如表 3.1 所示。

表 3.1 Cortex-M3 与 ARM7TDMI 性能比较所示

Table 3.1 The performance comparison between Cortex-M3 与 ARM7TDMI

特性	ARM7TDMI	Cortex-M3
架构	ARMv4T(冯·若依曼)	ARMv7-M(哈佛)
ISA支持	Thumb/ARM	Thumb/Thumb-2
流水线	3级	3级+分支预测
中断	FIQ/IRQ	240个物理中断
中断延时	24-42个时钟周期	12个时钟周期（末尾连锁仅6个周期）
休眠模式	无	内置
存储器保护	无	8段存储器保护单元
硬件除法	无	2-12个时钟周期
运行速度	0.95 DMIPS/MHz	1.25 DMIPS/MHz
功耗	0.28 mW/MHz	0.19 mW/MHz
面积	0.62 mm^2	0.86 mm^2(内核+外设)

Cortex-M3 内核的内部方框图如图 3.3 所示，它主要分为以下几个组件：处理器内核、嵌套向量中断控制器(NVIC)、存储器保护单元(MPU)、总线接口、低成本调试解决方案。下面逐一介绍[13]。

① 处理器内核，特点：门数目少，中断延迟短。

- ARMv7-M：Thumb-2 ISA 子集，包含所有基本的 16 位和 32 位 Thumb-2 指令
- 只有 SP 是分组的，寄存器集比 ARM7 简单
- 硬件除法指令，SDIV 和 UDIV（Thumb-2 指令
- 处理模式（handler mode）和线程模式（thread mode）
- Thumb 状态和调试状态
- 可中断-可继续（interruptible-continued）LDM/STM，PUSH/POP，实现低中断延迟
- 自动保存和恢复处理器状态，可以实现低延迟地进入和退出中断服务程序（ISR）
- 支持 8 位、16 位和 32 位等非对齐访问

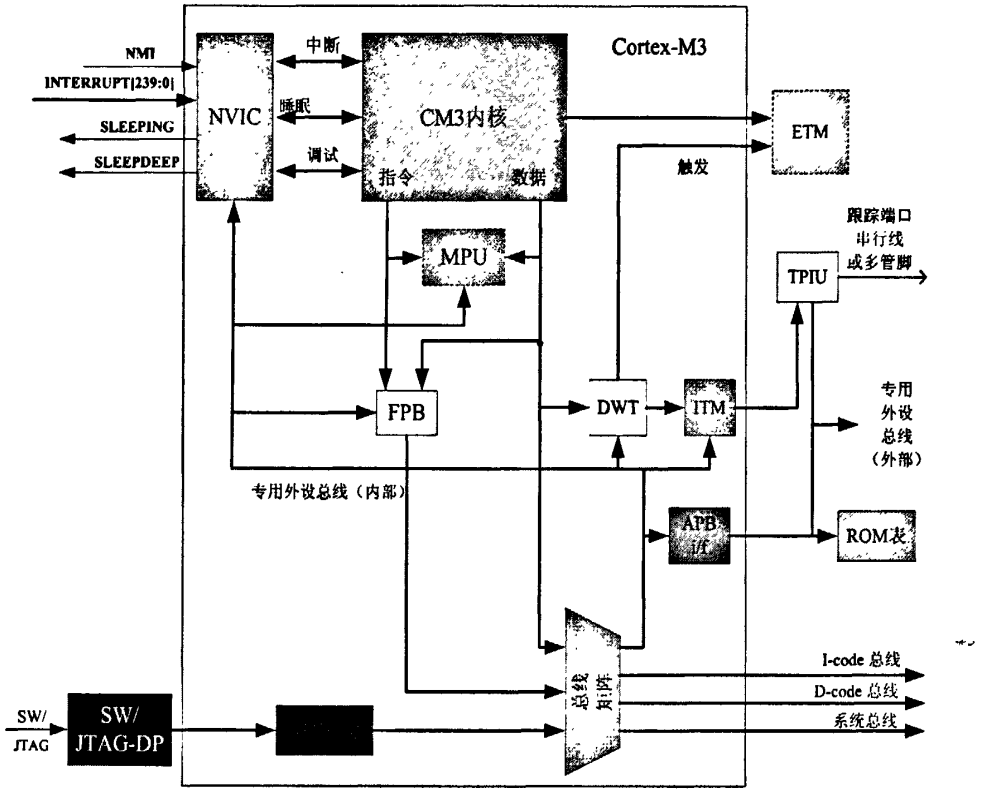


图 3.3 ARM Cortex-M3 内核内部方框图

Fig.3.3 The internal block diagram of ARM Cortex-M3 core

② 嵌套向量中断控制器（NVIC），与处理器内核紧密结合实现低延迟中断处理。

- 外部中断可配置为 1～240 个
- 优先级位可配置为 1～8 位
- 中断优先级可动态地重新配置
- 优先级分组。分为占先中断等级和非占先中断等级
- 支持末尾连锁（tail-chaining）和迟来（late arrival）中断。这样，在两个中断之间没有多余的状态保存和状态恢复指令的情况下，使能背对背中断（back-to-back interrupt）处理
- 处理器状态在进入中断时自动保存，中断退出时自动恢复，不需要多余的指令

③ 存储器保护单元（MPU）。MPU 功能可选，用于对存储器进行保护。

- 8 个存储器区
- 子区禁止功能(SRD)，实现对存储器区的有效使用
- 可使能背景区，执行默认的存储器映射属性

④ 内部总线接口

- AHBLite ICode、DCode 和系统总线接口

- APB 专用外设总线（PPB）接口
- Bit band 支持，bit-band 的原子写和读访问
- 存储器访问对齐
- 写缓冲区，用于缓冲写数据

⑤ 成本调试解决方案

- 当内核正在运行、被中止、或处于复位状态时，能对系统中包括 Cortex-M3 寄存器组在内的所有存储器和寄存器进行调试访问
- 串行线（SW-DP）或 JTAG（JTAG-DP）调试访问，或两种都包括
- Flash 修补和断点单元（FPB），实现断点和代码修补
- 数据观察点和触发单元（DWT），实现观察点，触发资源和系统分析（system profiling）
- 仪表跟踪宏单元（ITM），支持对 printf 类型的调试
- 跟踪端口的接口单元（TPIU），用来连接跟踪端口分析仪
- 可选的嵌入式跟踪宏单元（ETM），实现指令跟踪

(2) Cortex-M3 内核工作流程

Cortex-M3 中央内核基于哈佛架构，指令和数据各使用一条总线。与 Cortex-M3 不同，ARM7 系列处理器使用冯·诺依曼（Von Neumann）架构，指令和数据共用信号总线以及存储器。由于指令和数据可以从存储器中同时读取，所以 Cortex-M3 处理器对多个操作并行执行，加快了应用程序的执行速度。

内核流水线分3个阶段：取指、译码和执行。当遇到分支指令时，译码阶段也包含预测的指令取指，这提高了执行的速度。处理器在译码阶段期间自行对分支目的地指令进行取指。在稍后的执行过程中，处理完分支指令后便知道下一条要执行的指令。如果分支不跳转，那么紧跟着的下一条指令随时可供使用。如果分支跳转，那么在跳转的同时分支指令可供使用，空闲时间限制为一个周期。Cortex-M3内核包含一个适用于传统Thumb和新型Thumb-2指令的译码器、一个支持硬件乘法和硬件除法的先进ALU、控制逻辑和用于连接处理器其他部件的接口。Cortex-M3处理器是一个32位处理器，带有32位宽的数据路径，寄存器库和存储器接口。其中有13个通用寄存器，两个堆栈指针，一个链接寄存器，一个程序计数器和一系列包含编程状态寄存器的特殊寄存器。Cortex-M3处理器支持两种工作模式（线程（Thread）和处理器（Handler））和两个等级的访问形式（有特权或无特权），在不牺牲应用程序安全的前提下实现了对复杂的开放式系统的执行。无特权代码的执行限制或拒绝对某些资源的访问，如某个指令或指定的存储器位置。Thread是常用的工作模式，它同时支持享有特权的代码以及没有特权的代码。当异常发生时，进入Handler模式，在该模式中所有代码都享有特权。此外，所有操作均根据以下两种工作状态进行分类，Thumb代表常规执行操作，Debug代表调试操作。

(3) LM3S8962最小系统电路

CPU 芯片的最小系统电路[14]如图 3.4 所示，该电路除了 CPU 以外还包括晶体振荡器、复位电路、CPU 电源以及以太网物理层接口的电路。其管脚有 100 个，TQFP 封装。

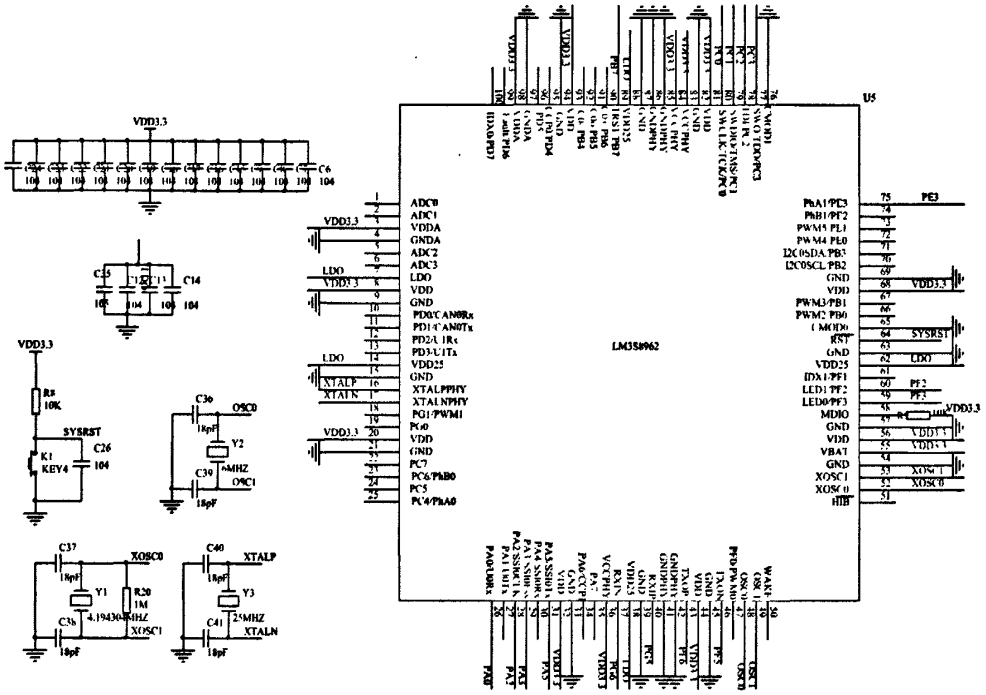


图 3.4 LM3S8962 最小系统电路

Fig.3.4 The minimum system circuit of LM3S8962

CPU 共接了 8 组数字电源，其作用用于多点输入和多点接地，这样提高了 CPU 的抗干扰能力。对于每一组输入电源都接了 0.1μF 的高频电容，起到高频旁路作用。LDO 引脚外接 0.1μF 的滤波电容，可以输出稳定的 2.25—2.75V 的电压，因此它可以接到 VDD2.5 端给内核提供电源，这样可以少设计一路电源，从而简化了电源电路的设计。为了增加以太网通信的可靠性，以太网的电源与地间也要加上 0.1μF 的滤波电容。

晶体振荡器标配 6.000MHz 的石英晶体作为主振荡器，电路中还有 CPU 在休眠模块下使用的晶振，标配 4.194304MHz 的石英晶振，以太网接口配有 25MHz 的石英晶振。晶体振荡器的输入引脚不可用手触摸，所以在设计 PCB 时，时钟输入引脚与晶振的走线短些更好。

(4) LM3S8962 基本功能外设

LM3S8962 处理器性能良好，并提供了丰富的外设,包括以太网接口、CAN 接口。I^2C 接口，UART 接口、ADC 接口、CCP 接口、SSI 接口、模拟比较器、PWM 接口等。这里我们主要用到了 GPIO，定时器和以太网控制接口这三个功能外设，下面就主要讲解这三个外设。

① GPIO

GPIO 模块由 8 个物理 GPIO 模块组成，每个对应一个独立的 GPIO 端口（端口 A，端口 B，端口 C，端口 D，端口 E，端口 F，端口 G 和端口 H，）。GPIO 模块遵循 FiRM (Foundation IP for Real-Time Microcontrollers)规范，并且支持 0-60 个可编程的输入/输出管脚，具体取决于正在使用的外设。

GPIO 模块具有以下的特性：

- 可编程控制 GPIO 中断
  - 屏蔽中断发生
  - 边沿触发（上升沿，下降沿，上升、下降沿）
  - 。(高或低)电平触发
- 输入/输出可承受 5V 电压
- 在读和写操作中通过地址线进行位屏蔽
- 可编程控制 GPIO 引脚（pad）配置
  - 弱上拉或下拉电阻
  - 2-mA，4-mA 和 8-mA 引脚驱动
  - 8-mA 驱动的斜率控制
  - 开漏使能
  - 数字输入使能

除了 5 个 JTAG/SWD 管脚（PB7 和 PC[3:0]）之外，所有 GPIO 管脚默认都是三态管脚（GPIOAFSEL=0，GPIODEN=0，GPIOPDR=0，且 GPIOPUR=0）。JTAG/SWD 管脚默认为 JTAG/SWD 功能（GPIOAFSEL=1，GPIODEN=1 且 GPIOPUR=1）。通过上电复位（POR）或外部复位（RST），可以让这两组管脚都回到其默认状态。每个 GPIO 端口都是同一物理块中的独立硬件的实例化（hardware instantiation）。LM3S8962 微控制器含有 8 个端口以及 8 个物理 GPIO 块。对数据传送控制，数据传送方向，中断设置等的控制都是通过调用驱动库函数来完成的，在此不做详细介绍。

② 定时器

可编程定时器可对驱动定时器输入管脚的外部事件进行计数或定时。Stellaris® 通用定时器模块（GPTM）包含 4 个 GPTM 模块（定时器 0，定时器 1，定时器 2 和定时器 3）。每个 GPTM 模块包含两个 16 位的定时/计数器（称作 TimerA 和 TimerB）。用户可以将它们配置成独立的定时器或事件计数器，或将它们配置成 1 个 32 位定时器或 1 个 32 位实时时钟（RTC）。定时器还可以用来触发模数转换（ADC）。由于所有通用定时器的触发信号在到达 ADC 模块前一起进行或操作，因而只需使用一个定时器来触发 ADC 事件。

支持以下模式：

- 32-位定时器模式

- 可编程单次触发（one-shot）定时器
- 可编程周期定时器
- 使用 32.768-KHz 输入时钟的实时时钟
- 事件的停止可由软件来控制（RTC 模式除外）

■ 16-位定时器模式

- 带 8 位预分频器的通用定时器功能模块（仅单次触发模式和周期模式）
- 可编程单次触发（one-shot）定时器
- 可编程周期定时器
- 事件的停止可由软件来控制

■ 16-位输入捕获模式

- 输入边沿计数捕获
- 输入边沿定时捕获

■ 16-位 PWM 模式

- 简单的 PWM 模式，可通过软件实现 PWM 信号的输出反相。

每个 GPTM 模块的主要元件包括两个自由运行的先递增后递减计数器（称作 TimerA 和 TimerB）、两个 16 位匹配寄存器、两个预分频器匹配寄存器、两个 16 位装载/初始化寄存器和它们相关的控制功能。GPTM 的准确功能可由软件来控制，并通过寄存器接口进行配置。在通过软件对 GPTM 进行配置时需用到 GPTM 配置（GPTMCFG）寄存器 、GPTM TimerA 模式（GPTMTAMR）寄存器和 GPTM TimerB 模式（GPTMTBMR）寄存器。当 GPTM 模块处于其中一种 32 位模式时，该定时器只能作为 32 位定时器使用。但如果配置为 16 位模式，则 GPTM 的两个 16 位定时器可配置为 16 位模式的任意组合。对定时器的操作主要也是通过调用驱动库函数来完成的，在此也不做过多的介绍。

③ 以太网控制器接口

以太网接口是本项目中最关键的一个外设，也是我们选用此芯片的主要原因。Stellaris® 以太网控制器由一个完全集成的媒体访问控制器（MAC）和网络物理（PHY）接口器件组成。以太网控制器遵循 IEEE 802.3 规范，完全支持 10BASE-T 和 100BASE-TX 标准。其方框图如图 3.5 所示。

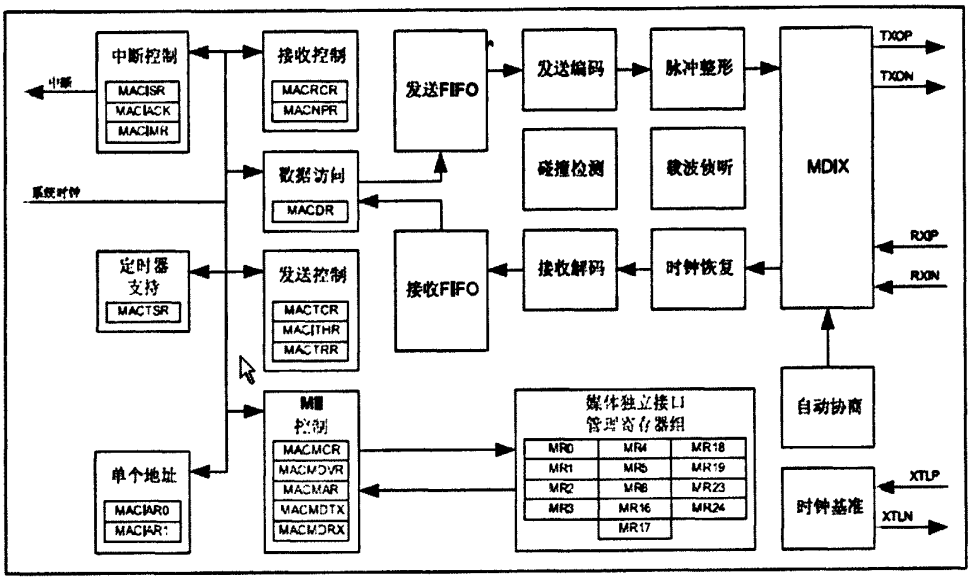


图 3.5　以太网控制器方框图

Fig.3.5　The block diagram of ethernet controller

以太网控制器模块具有以下特性：

■ 遵循 IEEE 802.3-2002 规范

- 遵循 10BASE-T/100BASE-TX IEEE-802.3。只需要一个双路 1:1 隔离变压器就能与线路相连
- 10BASE-T/100BASE-TX ENDEC, 100BASE-TX 扰码器/解扰器
- 全功能的自协商

■ 多种工作模式

- 全双工和半双工 100 Mbps
- 全双工和半双工 10 Mbps
- 节电和掉电模式

■ 高度可配置

- 可编程 MAC 地址
- LED 活动选择
- 支持混杂模式
- CRC 错误拒绝控制
- 用户可配置的中断

■ 物理媒体操作

- 自动 MDI/MDI-X 交叉校验
- 寄存器可编程的发送幅度
- 自动极性校正和 10BASE-T 信号接收

以太网控制器在功能上被划分为两层或两个模块——媒体访问控制器（MAC）层和网络物理（PHY）层。它们与 ISO 模型的第 2 和第 1 层相对应。以太网控制器的基本接口是到 MAC 层的一个简单总线接口。MAC 层提供了以太网帧的发送和接收处理。MAC 层还通过一个内部的媒体独立接口（MII）给 PHY 模块提供接口。功能框图如图 3.6 所示。

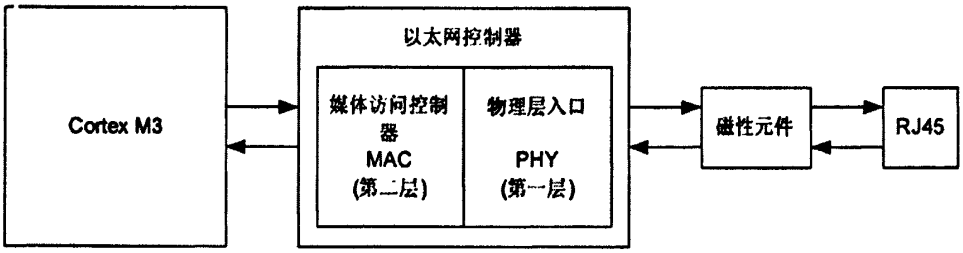


图 3.6　以太网模块功能框图

Fig.3.6　The block diagram of ethernet module functional

3.2.2　单片机控制电路设计

由于 LM3S8962 没有总线操作方式，但有丰富的 GPIO，所以可以用 IO 口模拟总线进行单片机与 FPGA 的之间的通信，考虑到有些 GPIO 具有功能复用的特性，将这些资源预留了出来，以备后期产品升级使用。所以数据总线和地址总线跨越几个端口间隔分配如下表：

表 3.2　16 位地址总线对应的 IO 口

Table 3.2　16-bit address bus corresponding to the IO port

地址信号	Addr15	Addr14	Addr13	Addr12	Addr11	Addr10	Addr9	Addr8
对应IO口	PA7	PA6	PA5	PA4	PA3	PA2	PE1	PE0
地址信号	Addr7	Addr6	Addr5	Addr4	Addr3	Addr2	Addr1	Addr0
对应IO口	PD7	PD6	PD5	PD4	PD3	PD2	PE1	PE0

表 3.3　8 位数据总线对应的 IO 口

Table 3.3　8-bit data bus corresponding to IO port

数据信号	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0
对应IO口	PC7	PC6	PC5	PC4	PF3	PF2	PF1	PF0

表 3.5 保留的特殊功能管脚

Table3.5 Special function pin to retain

特殊功能脚	ADC0	ADC1	ADC2	ADC3	PWM3	I^2COSCL	I^2COSDA	CANORx	CANOTx
对应 IO 口	PE4	PE5	PE6	PE7	PB1	PB2	PB3	PD0	PD1

总线控制信号中写信号$\overline{WR}$对应 IO 口中的 PG0，读信号$\overline{RD}$对应 PG1。PB0，PB4，PB5，PB6 分配为键盘的控制信号。PD5、PD6 作为从 FPGA 输入到 CPU 的单向通讯信号，基中 PD6 为计数结束的标志信号，PD5 为触发 CPU 中断的中断信号输入。

如图 3.7 所示，单片机以总线形式与 FPGA 相连[15]，并并完成整个测量电路的测试控制、数据处理、键值输入和液晶显示输出的控制、管理等工作。FPGA 内部集成了门控电路、计数电路和分频电路等，从而完成如下几种测试功能。

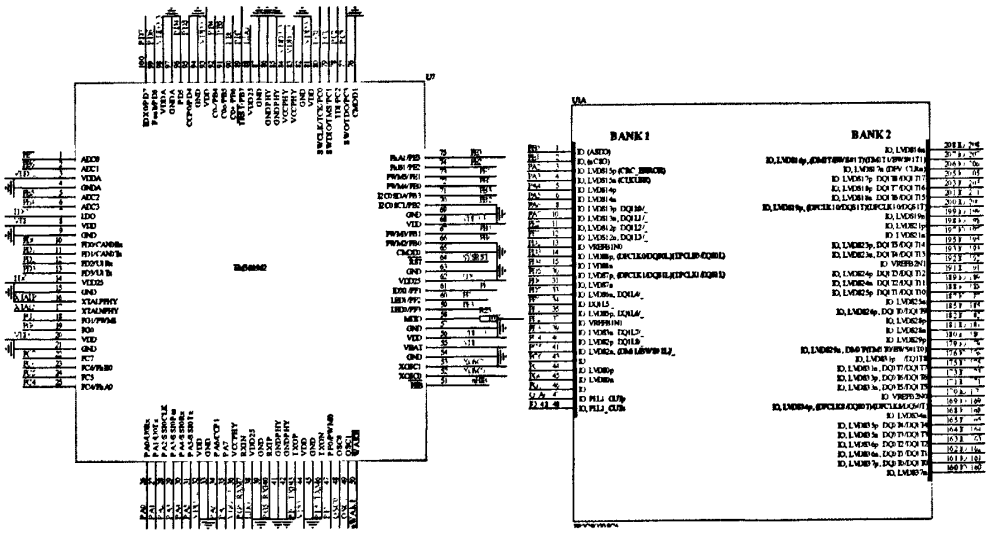


图 3.7 单片机控制电路

Fig.3.7 The MCU control circuit

(1) 完成键盘扫描。输出扫描信号，根据行线的状态，判断按键的状态和键值，然后根据键值执行相应的功能。

(2) LCD 驱动及显示。完成 LCD 的初始化工作，并在 FPGA 内的测量模块计数结束后，通过总线将测量结果读回到单片机，在程序内部经过计算，送到 LCD 上进行显示。

(3) 对 FPGA 的控制[16]。根据人机对话中的分频设置，发出分频命令，锁存到 FPGA 的中，由分频器执行；根据测试对象的设置，发出选择命令，通过 FPGA 中的门电路切换到不同的测试平台，使被测信号与基准信号进入不同的通道，自动选择与测试功能相匹配的更精确的测试方法。

(4) 在进行时间继电器测量时，闸门的开启、关闭时刻是通过时间继电器本身控制的，由单片机根据启动键的键值决定，是否进行本次测量；在进行晶振的测量时，闸门

的周期是通过键盘设置并由单片机发出分频命令控制的。

(5)完成对多通道的测量结果的轮循。完成控制系统的复位与清零，保证连续测量的准确性。当因各种故障引起死机时，通过软件复引自动重启。

3.3 FPGA 内部的测试模块

3.3.1 FPGA 芯片介绍

现场可编程门阵列 FPGA (Field Programmable Gate Array)和复杂可编程逻辑器件 CPLD (Complex Programmable Logic Device)同属于近年来发展迅速的大规模可编程专用集成电路(ASIC)。FPGA 器件的现场可编程技术[17]和 CPLD 器件的在系统可编程技术 ISP(In System Programmable)使可编程器件在使用上更为方便，大大缩短了设计周期，减少了设计费用，降低了设计风险。现场可编程门阵列 FPGA 是一种高密度的可编程逻辑器件，其集成密度最高达 100 万门/片，系统性能可达 200MHz。CPLD 主体结构是与或阵列，具有 ISP 功能的 CPLD 器件由于具有同 FPGA 器件相似的集成度和易用性，在速度上还有一定的优势，使其在可编程逻辑器件技术的竞争中与 FPGA 并驾齐驱，成为两支领导可编程器件技术发展的力量之一。

（1）CPLD 与 FPGA 性能特点比较

FPGA 与 CPLD 的辨别和分类主要是根据其结构特点和工作原理。通常的分类方法是：将以乘积项结构方式构成逻辑行为的器件称为 CPLD，如 Lattice 的 ispLSI 系列、Xilinx 的 XC9500 系列、Altera 的 MAX7000S 系列和 Lattice(原 Vantis)的 Mach 系列等。将以查表法结构方式构成逻辑行为的器件称为 FPGA，如 Xilinx 的 SPARTAN 系列、Altera 的 FLEX10K 或 ACEX1K 系列等。尽管 FPGA 和 CPLD 都是可编程 ASIC 器件，有很多共同特点，但由于 CPLD 和 FPGA 结构上的差异，具有各自的特点：

①CPLD 更适合完成各种算法和组合逻辑，FP GA 更适合于完成时序逻辑。换句话说，FPGA 更适合于触发器丰富的结构，而 CPLD 更适合于触发器有限而乘积项丰富的结构。

②CPLD 的连续式布线结构决定了它的时序延迟是均匀的和可预测的，而 FPGA 的分段式布线结构决定了其延迟的不可预测性。

③在编程上 FPGA 比 CPLD 具有更大的灵活性。CPLD 通过修改具有固定内连电路的逻辑功能来编程，FPGA 主要通过改变内部连线的布线来编程；FP GA 可在逻辑门下编程，而 CPLD 是在逻辑块下编程。

④FPGA 的集成度比 CPLD 高，具有更复杂的布线结构和逻辑实现。

⑤CPLD 比 FPGA 使用起来更方便。CPLD 的编程采用 E2PROM 或 FASTFLASH 技术，无需外部存储器芯片，使用简单。而 FPGA 的编程信息需存放在外部存储器上，使用方法复杂。

⑥CPLD 的速度比 FPGA 快，并且具有较大的时间可预测性。这是由于 FPGA 是门级编

程,并且CLB之间采用分布式互联,而CPLD是逻辑块级编程,并且其逻辑块之间的互联是集总式的。

⑦在编程方式上,CPLD主要是基于E2PROM或FLASH存储器编程,编程次数可达1万次,优点是系统断电时编程信息也不丢失。CPLD又可分为在编程器上编程和在系统编程两类。FPGA大部分是基于SRAM编程,编程信息在系统断电时丢失,每次上电时,需从器件外部将编程数据重新写入SRAM中。其优点是可以编程任意次,可在工作中快速编程,从而实现板级和系统级的动态配置。

⑧CPLD保密性好,FPGA保密性差。

⑨一般情况下,CPLD的功耗要比FPGA大,且集成度越高越明显。

(2) CycloneII系列 FPGA器件

FPGA主要生产厂有Altera、Xilinx、Actel、Lattice,其中Actel主要提供非易失性FPGA,产品主要基于反熔丝工艺和FLASH工艺。Altera和Xilinx主要生产一般用途的FPGA,其主要产品采用RAM工艺。Altera公司的可编程逻辑产品可以分为高密度FPGA、低成本FPGA和CPLD3类,每个产品类别在不同时期都有其主流产品。在Altera近几年的产品系列中,高端高密度FPGA有APEX和Stratix系列;低成本FPGA有ACEX和Cyclone系列;CPLD有MAX7000B、MAX3000A和MAXⅡ。基于各个器件的特性及本系统[18]的实际需要,我们决定使用Altera公司低成本FPGA CycloneII系列EP2C8Q208C6芯片。下面对Cyclone系列FPGA进行简单介绍。

第一代Cyclone FPGA是2002年12月份推出的,主要用于对成本敏感的设计中,如数字终端、手持设备、工业类和汽车市场。对于在大批量应用中需要更低成本、更大容量和更多功能的设计者,可使用Cyclone ⅡFPGA。Cyclone FPGA是基于成本优化的,130nm全铜工艺的1.5V SRAM工艺。容量为2910~20060个逻辑单元,拥有最大288K位的RAM;除此之外,Cyclone FPGA还集成了许多复杂功能。Cyclone FPGA提供了全功能的锁相环(PLL),用于板级的时钟网络管理和专用的I/O接口,这些接口用于连接业界标准的外部存储器器件。Cyclone FPGA具有[19]以下特性:

① 新的可编程构架通过设计实现低成本;

②嵌入式存储资源支持各种存储器应用和数字信号处理(DSP);

③专用外部存储接口电路集成了DDR FCRAM和SDRAM器件以及SDR SDRAM存储器件;

④支持串行总线和网络接口及各种通讯协议;

⑤使用PLL管理片内和片外系统时序;

⑥支持单端I/O标准和差分I/O技术,支持高达311 Mbps的LVDS信号;

⑦支持Nios Ⅱ系列嵌入式处理器;

⑧采用新的串行配置器件的低成本配置方案。

Cyclone FPGA支持各种单端I/O标准,如LVTTL、LVCMOS、PCIS和STL-2/3。通过

LVDS 和 RSDS 标准提供多达 129 个通道的差分 I/O 支持，每个 LVDS 通道高达 640 Mbps。Cyclone 器件具有双数据速率（DDR）SDRAM 和 FCRAM 接口的专用电路。Cyclone FPGA 中有 2 个锁相环（PLL）提供 6 个输出和层次时钟结构，以及复杂设计的时钟管理电路。

Altera 在大获成功的第一代 Cyclone 系列的基础上，开发了全铜层 90nm 低 k 绝缘工艺，1.2VSRAM 工艺，在 300mm 圆晶片上生产 Cyclone II FPGA。Cyclone II FPGA 具有很高的性能和极低的功耗，而价格和 ASIC 相当，能够提供多种功能，为价格敏感的应用提供大批量产品解决方案。Cyclone II 器件是汽车、通信、消费类、视频处理、测试和测量以及其他终商市场解决方案的理想选择。

Cyclone II 器件提供了 4608 至 68416 个逻辑单元（LE），并具有一整套最佳的功能，包括嵌入式 18 位×18 位乘法器、专用外部存储器接口电路、4K 位嵌入式存储器块、锁相环（PLL）和高速差分 I/O。具体到 EP2C8 这款芯片，它的内部资源有 56 个逻辑单元，36 个 M4K RAM 块（4K 位+512 位校验位），总位数为 165888，18 个嵌入式 18×18 乘法器，2 个 PLL，最多 182 个用户 I/O 引脚，77 个差分通道。

（3）Altera 可编程逻辑器件开发[20]设计

Altera 公司在推出各种可编程逻辑器件的同时，也不断升级其相应的开发工具软件。目前，主流的开发软件为 Quartus II。使用 Quartus II 的设计过程包括以下几步，若任一步出错或未达到设计的要求则应修改设计，然后重复以后各步。设计流程如图 3.8。

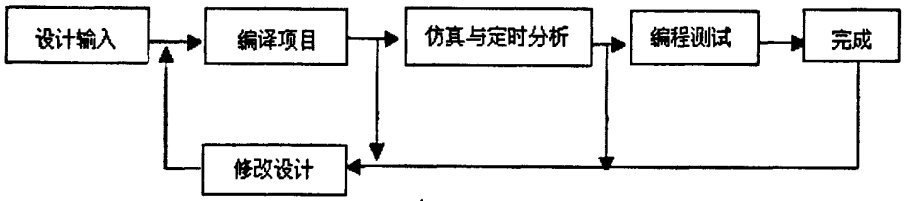


图 3.8 FPGA 芯片的设计流程

Fig 3.8 The design flow chart of FPGA

3.3.2 完成测试的各功能模块设计

(1) 单片机对 FPGA 的测试功能选择及分频的控制[21] [22]

这个模块主要用到了锁存器和分频器。首先介绍一个锁存器，锁存器(Latch)是一种对脉冲电平敏感的存储单元电路，输出端的状态不会随输入端的状态变化而变化，只有在有锁存信号时输入的状态被保存到输出，直到下一个锁存信号。如图 74373 模块是从函数库里调用的一个锁存器，它有两个使能控制端：OEN 和 G，仅当 OEN 为低电平，G 为高电平时，输入的状态才被保存到输出，在其它任意时刻，输

出保持不变。主要作用是缓存，完成高速的控制与慢速的外设之间的不同步问题。

总线的写信号/WR 与译码出来的地址信号 I800DH 经过与非门形成 G 端的控制信号，根据与非门的真值表可知，仅当/WR 与 I800DH 均为低电平时，输出才为高电平，此时，使能端 G 有效，锁存器输入端状态传递到输出端。因为/WR 与 I800DH 都是低电平有效的信号，也就是说，当译码器选通 I800DH 地址，单片机向 I800DH 写数据时，数据总线上的 8 位数据被 74373 锁存。

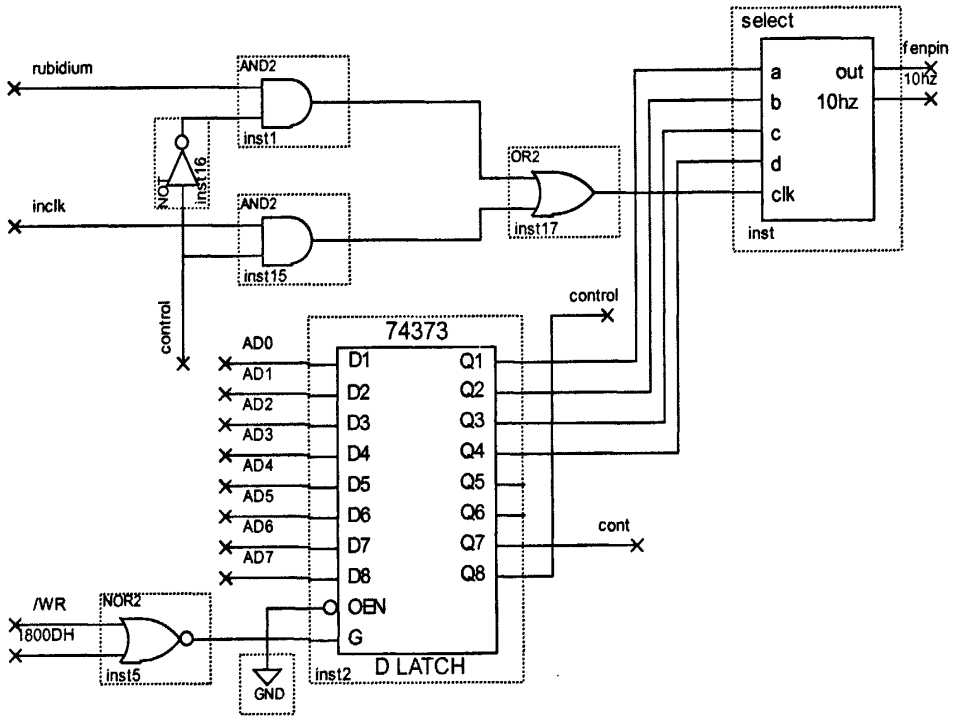


图 3.9　测试功能选择模块

Fig. 3.9　The module of selecting test function

图 3.9 中 select 模块封装[23]的内容如图 3.10，其中 74151 是 TTL 8 选 1 数据选择器。所谓数据选择是指经过选择，把多个通道的数据传送到唯一的公共数据通道上去，实现数据选择功能的逻辑电路称为数据选择器，它的作用相当于多个输入的单刀多掷开关。74151 使用 3 位地址码 A、B、C 产生 8 个地址信号，任何时候只能有一个地址信号有效，在输入使能端 GN 有效的前提下，使对应的那一路数据通过，送达到 Y 端。

select 模块的主要功能是完成对输入时钟信号 clk 的分频，分频模块由 fenpin5 和多个 fenpin 模块级联而成，其中 fenpin5 模块完成对 clk 的五分频，fenpin 模块完成对输入信号的十分频。各个分频单元的输出分别连接到 74151 的输入通道上。

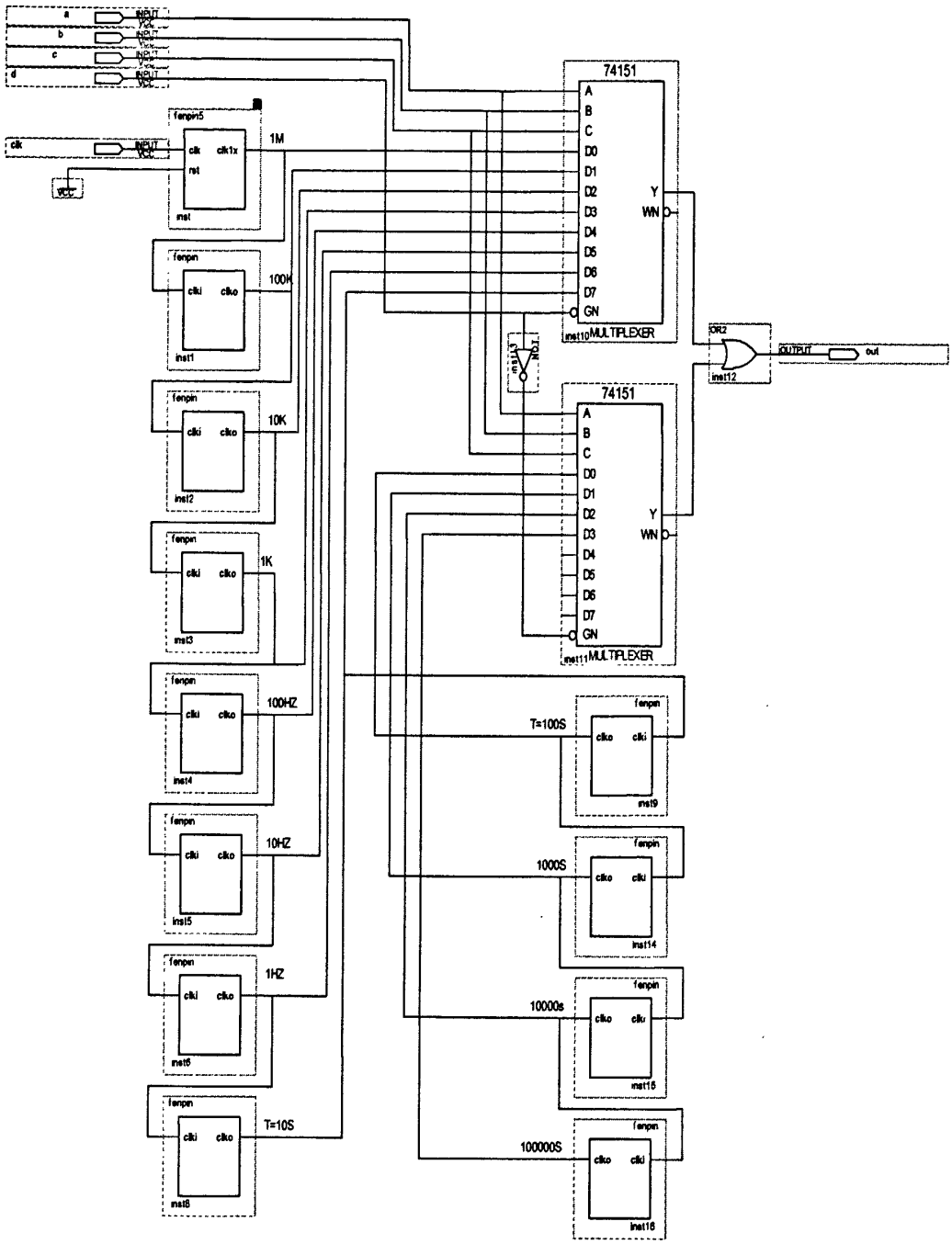


图 3.10　分频模块

Fig. 3.10　The module of frequency divider

下面，结合图 3.9 和图 3.10，分析图 3.9 中输入到锁存器 74373 的 8 位数据各自代表的含义。

AD0、AD1、AD2 分别控制 select 模块的 a、b、c，作为 select 模块内的数据选择器 74151 的地址码，选择时钟信号 clk 的分频输出。

AD3 将作为 74151 的输入使能端，决定选择 select 模块中哪一个 74151。测量晶振时采用直接测频法，将输入的铷标信号分频成精确的闸门，闸门时间越宽，测量结果越准确；测量继电器时采用的是间接测频法，恒温晶振的输入分频后作为时基信号，时基信号频率越高，测量结果越准确。所以对两种对象的测量所用的频段不同，所以在 select 模块内设计了两个数据选择器，根据需要切换到不同的测量频段。

AD6 作为启动本次测量的标志信号 cont，平时为低电平，由按键控制产生一个高电平脉冲，标志着可以对随后出现的被测信号进行测量。如果 AD6 没有出现高电平脉冲，即使出现符合条件的被测信号也不会进行测量。还需要考虑的一个问题是，当 cont 信号和被测信号同时为高时，若开始了本次计数，计数模块就会因为开始计时时间晚，而得到小于被测信号实际长度的值，造成与真实值偏差比较大测量结果。解决方案是本次不能计数，但启动信号 cont 仍然有效，只是等到下一次被测信号为高时，才开始计数。这样设计，提高了系统的抗干扰能力并保证了精度。

AD7 决定进入 select 模块被分频的时钟是来自恒温晶振还是来自铷基时标。当 control=1（为高电平）时，图 3.9 的与门 inst15 打开，inst1 关闭，可知，两路与门的输出信号通过或门 int17，连接到到 select 模块 clk 管脚上的是 inclk 信号，inclk 信号是主板上既为 FPGA 提供全局时钟又为继电器测量提供时基的恒温晶振的输出信号；control=0 时，图？的与门 inst15 关闭，inst1 打开，从或门 int17 输出的是信号 rubidium，即此时被 select 模块分频的是铷标的时钟。

AD4、AD5 为系统预留，以后可以扩充系统其它功能。

(2)测量控制模块设计

以对继电器测量时，第一个通道的数据采集为例，简单阐述控制原理如下。选择继电器测量时，单片机发出 control=1 的命令，经过前级的门电路逻辑和分频电路，进入分频器被分频的时基信号是恒温晶振的输出，这个信号连接到 jingdu 模块的 fenpin 引脚；无论有没有被测信号，jingdu 模块的 jidianqi 引脚和 pinliv 引脚分别连接对应测试对象插座上的输入，当 control 信号的值确定下来后，jingdu 模块内部会自动选择哪一路被测信号有效；cont 对应前级的启动信号；测试完毕后，输出 kk 高电平信号，单片机检测到 kk 的跳变后，就会通过总线读回测量结果；计数结果为 12 位，分 6 次由 datal～data6 的地址选通信号控制，先后送到数据总线 y1[8..1]上。

本测试仪能一次采集 8 路通道的数据，这 8 个通道分别由 8 个 jingdu 模块控制。在由一路扩展到 8 路的设计里，需要做两个工作，第一，作为了使单片机能够识别各个通道，每个 jindu 模块所对应的地址选通信号 datal～data6 各不相同，具体分配如表 3.6 所示。

表 3.6 各通道对应的地址选通信号

Table 3.6 Address signal corresponding to each channel

	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7	通道 8
data1	I8010H	I8016H	I801CH	I8022H	I8028H	I802EH	I8034H	I803AH
data2	I8011H	I8017H	I801DH	I8023H	I8029H	I802FH	I8035H	I803BH
data3	I8012H	I8018H	I801EH	I8024H	I802AH	I8030H	I8036H	I803CH
data4	I8013H	I8019H	I801FH	I8025H	I802BH	I8031H	I8037H	I803DH
data5	I8014H	I801AH	I8020H	I8026H	I802CH	I8032H	I8038H	I803EH
data6	I8015H	I801BH	I8021H	I8027H	I802DH	I8033H	I8039H	I803FH

这些地址信号都是由总线经译码器产生的，当要采集相应通道的测试结果时，单片机会依次产生对应的地址选通信号，使计数器的结果先后传到数据总线上。第二，单片机 I/O 资源有限，除去保留的特殊功能管脚，再除去键盘、LCD、总线占用的 I/O 口，只剩下一个可用的 I/O。这一个 I/O 要完成对 8 个测量模块各自的测量结束标志信号 kk 的检测，采用中断或轮询均不能解决问题，经反复思考，决定采用模仿总线的方式，将 8 个 jingdu 模块的 kk 输出连接起来，每一个模块的的 kk 连到这条单总线上先经过一个三态缓冲门电路，其使能端由一个地址选通信号控制。这样，单片机通过一个 I/O 口就可以监测 8 个通道的测量状态，实时地处理各个通道的测量结果，相互之间不会造成干扰。各个通道的 kk 标志信号所对应的三态缓冲器的使能端分配的地址选通信号如下：

表 3.7 三态缓冲器的使能端对应的地址选通信号

Table 3.7 Address signal corresponding to enable terminal of 3-state buffer

三态门	1	2	3	4	5	6	7	8
使能端	I8001H	I8002H	I8003H	I8004H	I8005H	I8006H	I8007H	I8008H

具有 8 个通道的测量模块如图 3.11 所示。

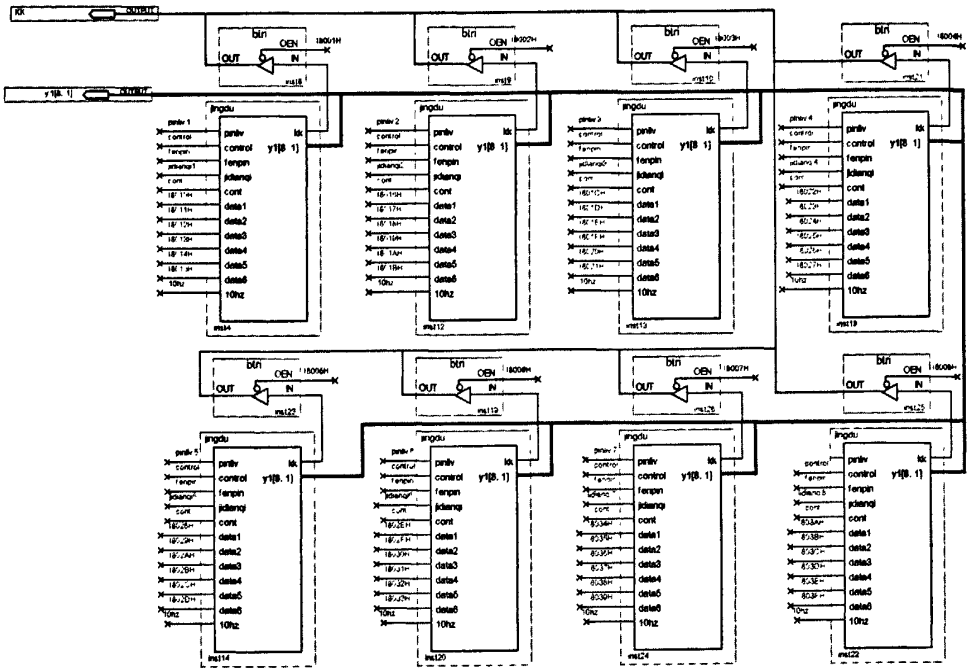


图 3.11 8 通道测量模块图

Fig 3.11 The model of 8-channel measurement

下面分三小节对 jingdu 模块内部各功能单元展开介绍。

①control 对测试对象的选择电路

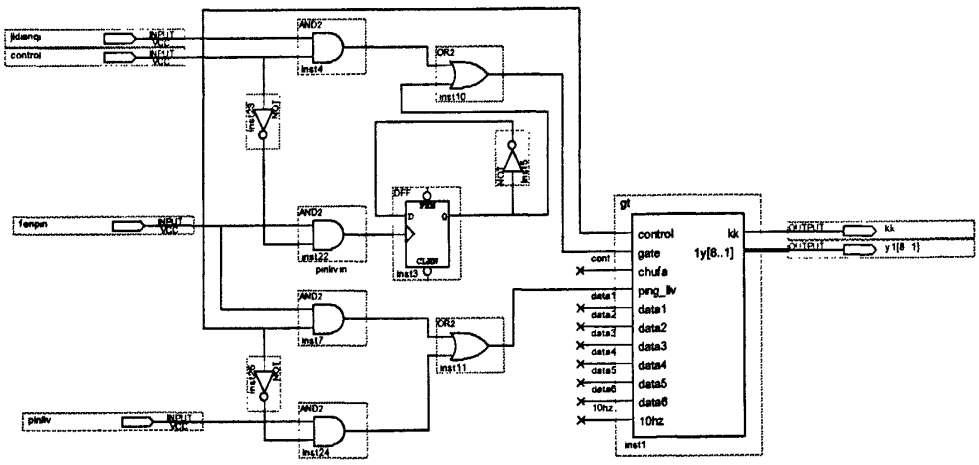


图 3.12 测试对象选择电路

Fig.3.12 The select circuit of Test object

当 control=1 时，由前面的叙述可知，图 3.12 中的 fenpin 输入信号是来自恒温晶振经 select 模块分频后的输出。control=1 时，与门 inst7 打开，fenpin 信号通过；inst24 关闭，pinliv 信号被过滤掉，所以，通过后级或门 inst11 连

接至 gt 模块 ping_liv 引脚上的，充当时基的信号是恒温晶振分频后的信号。与此同时，与门 inst4 打开，jidianqi 信号通过；与门 inst22 关闭，fenpin 信号被过滤掉，通过后级或门 inst10 连接至 gt 模块 gate 管脚上的，充当闸门的信号是被测继电器的输出。也就是说，当 control=1 时，进入 gt 模块的信号是被测时间继电器与恒温晶振分频后的信号，两者以符合间接测频法的原理递交给 gt 模块进行后续处理。

当 control=0 时，结合图 3.9 中的说明可知，此时的 fenpin 信号来自锄标经 select 模块分频后的输出。control=0 时，与门 inst7 关闭，fenpin 信号被过滤掉；inst24 打开，pinliv 信号通过，通过后级或门 inst11 连接至 gt 模块 ping_liv 引脚上的，充当时基的信号是外部被测晶振 pinliv 信号。同时，与门 inst4 关闭，jidianqi 信号被过滤掉；与门 inst22 打开，fenpin 信号通过。因为只有高电平时间才能做为被测信号的闸门时间，所以在 inst22 之后，用 D 触发器和非门构成一个二分频电路，对 fenpin 信号进行二分频，使闸门时间等于 fenpin 信号原来的周期，目的是使晶振与继电器的测量计算方法相同，简化软件程序设计。处理过的 fenpin 信号经后级或门 inst10 连接至 gt 模块 gate 管脚上的，充当闸门信号。符合晶振测量时采用的直接测频法。

②门控模块设计

下图是 gt 模块所封装内容的部分截图，主要包括门控模块和计数模块。首先介绍门控模块 d，门控模块负责响应 CPU 的控制信号，并对被测信号的测量时机在苛刻的条件下进行调控，它的逻辑功能的实现是整个系统的关键。以对继电器测量为例，简述其工作原理，当键盘上的启动键被按下时，单片机会在 d 模块的 chufa 管脚上产生一个高电平脉冲，启动本次测量，同时，chufa 信号通过一个非门 inst23 产生一个低电平脉冲对计数模块整体清零，防止前次测量数据对即将进行的下次测量产生影响；继电器的输入做为门控信号 gate，平时为低电平，继电器定时间隔内输出高电平；由恒温晶振产生振荡信号，经放大整形形成方波，分频后产生幅度一致，形状一致的计数脉冲（时标）输入到 d 模块的 ping_liv 引脚；count 为闸门时间下的计数脉冲输出，连接到后级级联产计数模块。kk 为测量结束标志信号，平时为低电平，完成测量后变为高电平，单片机取走本次测量结果后又变低，为下一次测量做准备。整个测量流程是：当门控信号 gate 发生高电平跳变时，count 输出计数脉冲，后级计数模块开始计数；gate 变为低电平时，count 输出随之变低，kk 位被置高并停止计数。计数模块记录下了闸门由打开到关闭期间输出的计数脉冲，通知并等待单片机来读取测量结果。

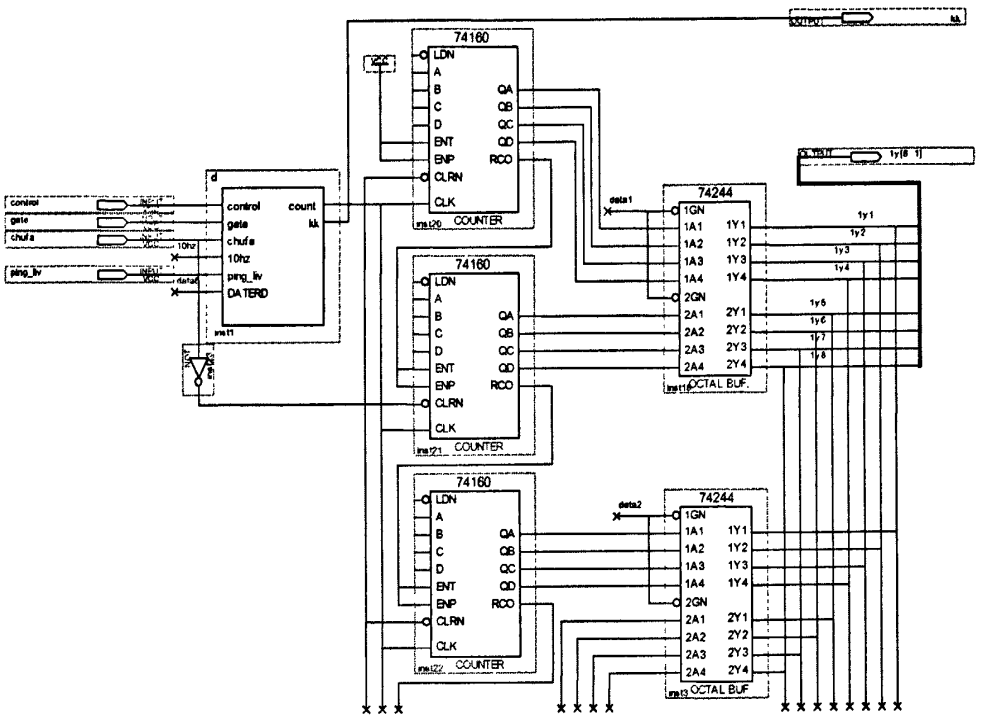


图 3.13　门控模块和计数模块

Fig.3.13　The gating module and the counter module

③实现高精度的计数模块设计[24]

影响本系统精度的主要因素是基准频率源，计数模块提供的计数位数，以及有效闸门时间下计数脉冲的同步输出。最后一个因素已在门控模块的设计中解决。对前两个因素的处理如下：为了消除温度对基准源振荡频率的影响，本系统采用了恒温晶体振荡器(OCXO)，它的标称频率为 5M，在系统开机 5 分钟后晶振频率准确度达 2.6×10^{-6}，能够满足企业标准的精度要求。测量晶振时，基准源采用铷标，频率准确度为 5×10^{-10}，铷时标株子共振理论从本质上说，其稳定性是机电结构晶体振荡器的 100 倍，由它引进的量化误差（±1 误差）基本上可以忽略。

计数模块采用了可预置数的十进制同步加法计数器 74160，它具有数据输入端 A、B、C 和 D，同步预置端 LDN、异步清除端 CLRN 和计数控制端 ENT 和 ENP，为方便级连，设置了进位输出端 RCO。74160 具有超前进位功能，当计数溢出时，进位输出端 RCO 输出一个高电平脉冲，其宽度为 Q_A 的高电平部分。当置数端 LDN=0、CLRN=1、CLK 脉冲上升沿时预置数。当 CLRN=LDN=1 而 ENT=ENP=0 时，输出数据和进位 RCO 保持。当 ENT=0 时计数器保持，但 RCO=0。CLRN=LDN=ENT=ENP=1 时，电路工作在计数状态。详细功能见功能表 3.8。

表 3.8 74016 真值表

Table 3.8 The truth table of 74016

输入					输出
CLRN	LDN	ENT	ENP	CLK	Q_n
0	x	x	x	x	异步清除
1	0	x	x	↑	同步预置
1	1	1	1	↑	计数
1	1	0	x	X	保持
1	1	x	0	x	保持

通过调用 12 个十进制集成计数器 74160，可提供最长 12 位长度的测量结果，能够满足对晶振精度的要求。为防止异步时序造成的竞争冒险，12 个 74160 的工作方式采用同步时序，它们的时钟输入端均接到门控模块的输出上，用前一个计数器的进位端连到后一个计数器的使能端实现级联。具体工作流程是：门控模块的输出平时为底，当有效的闸门时间来临时，同步输出基准频率源的时标脉冲，计数模块开始计数，闸门关闭时停止计数计数结果存储在多个 74160 的输出端。因为数据总线是 8 位，为了提高传输效率，将两个 74160 的输出连接到一个 74244 上，74244 的使能端由一个地址选通信号控制，这样，使用 6 个地址信号控制 6 个 74244 就可以传输 12 个 74160 的测量结果。最后一个地址信号 data6 被选通后，意味着所有数据已经通过总线被读到单片机里，这时门控模块的各测量状态应该进行一次复位，为下一次测量做好准备，这个动作会在 data6 出现低电平脉冲时由门控模块内的逻辑自动完成。下一次启动信号 chufa 到来时，计数模块被同步清零，为开始新的测量做好准备。

3.3.3 通用接口设计

若想将带有某一总线接口的仪器接入其他总线网络中通常采用的方法是：将带有不同总线接口的程控仪器配合相应的接口转换卡，如PCI-GPIB，PXI-LXI等。为了使仪器厂商在生产不同仪器的时候不需要由于仪器应用的普遍性或成本而取舍某一种总线接口方式，而是采用一种统一的接口平台，配合不同的接口转换卡，本设计中开发了一种新的接口平台，使用户可以根据自己的需要配备不同的接口转换卡，就可以方便的连接到现有的测试网络中；如果测试网络改变，只需更换接口转换卡即可，而测试仪器则完全不用改变或全套更换，这样，不仅方便省事，而且不会造成不必要的浪费，更有利于用户紧随科技发展的脚步不断的更新自己的测控网络。

本设计中，晶振与继电器测试仪和各类接口转换部分分别做在不同的板子上，都留有相同的 20 芯口，使用时用排线连接。通用接口模块的协议由仪器主板上的 FPGA 内部逻辑电路完成。主要设计思想是，命令和数据分开传递，复用 RAM 的地址线。

利用51单片机外部存储空间，8000H至8FFFH区域是作为IO口地址使用，用来传输双MCU的操作命令，相当于是命令区。9000H至9FFFH区域是作为内部RAM口地址使用，该区域用来访问FPGA内部设计的RAM，相当于是数据区。下面分命令传输和数据传输两部分来介绍接口模块完成的协议。信号线的命名，以I开头的代表与仪器主板CPU相关的信号，以O开头的代表通用模块与接口板通讯的信号。测试仪器与上位机的双向通讯，无论使用哪种协议，仪器主板向接口模块发送的命令和数据，都是没有经过封装的原始数据，双方通讯的数据在模块内所开辟的一个2K的RAM里存取，协议栈的完成由具体的接口转换板独立完成。明确的分工增强了仪器的通用性。

①命令传输

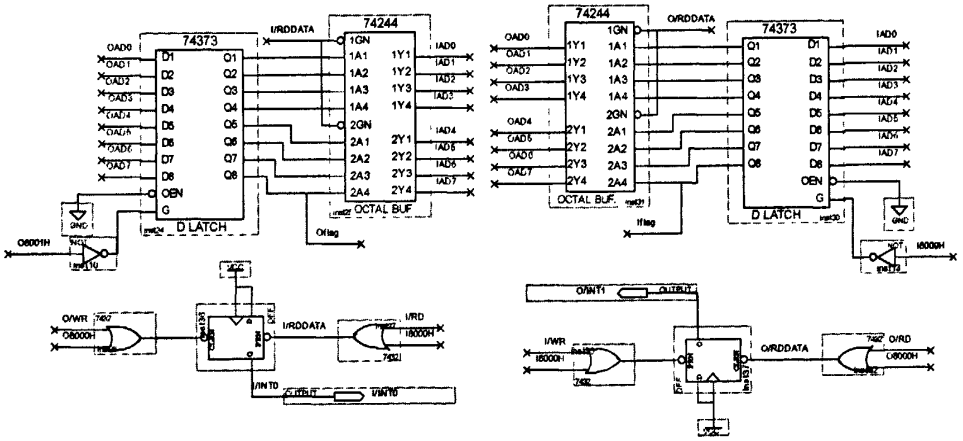


图3.14　通用接口模块第一部分

Fig.3.14 The first part of the Common Interface Module

如图3.14所示，左右两边都是双向口，都可以输入输出。现在假设左边为输入端（I端），右边为输出端（O端），即数据由上位机经过接口板发送到测试仪器。左边发送的8位数据，在地址信号08001H有效时，被inst24锁存，这8位数据分别代表不同的命令，目前仅使用一位，用OAD7信号的高、低电平两种状态代表外部接口板对RAM的占用与释放。为了防止对RAM访问的冲突，双方在占用RAM时，先要置高命令字节的第八位，表示自己要操作RAM了，对RAM读写数据完毕后，要置低命令字节的第八位，表示对RAM的释放。这样，对方在对共用的RAM读、写时，先读RAM的占用标志位，仅当RAM处于空闲状态时才可以操作。结合图3.14，对08001H写入0x01,即Oflag为高电平，占用RAM；随后，上位机发送的数据依次写入以9000H地址开始的RAM里；写完数据之后，对08001H写入0,即Oflag为低电平，释放RAM；接着对08000H地址进行一次写操作，通过D触发器inst36产生一个低电平输出，这个输出管脚连接到测试仪单片机的的中断管脚上，有效地触发了中断，使单片机即时从RAM里取走接口板发送过来的数据。

在单片机中断服务程序里，首先对I8000H地址进行一次读操作，产生有效的

I/RDDATA信号，完成两个目的,第一，通过使D触发器inst36的置位端有效,使输出变为高电平,即清除中断标志位，为CPU下一次响应中断做好准备。第二，使74244缓冲器inst25的使能端有效，接口板锁存在锁存器inst24的命令字状态就会传输到测试仪主板的数据总线上,读取并判断命令字的第八位,如果为高,等待;如果为低,则向地址I8009H写入0x80,使Iflag信号为高,占用RAM。随后主板从RAM里读走数据,完毕后向地址I8009H写入0,使Iflag信号为低，释放RAM。至次，完成了一次上位机到仪器的单向数据传输。

仪器的测量结果返回上位机时,完成对称的传输。主板锁存命令字到锁存器inst30,置高标志位信号Iflag，然后写数据到RAM，写完置低标志位，并触发接口板的中断0/INT1，在接口板单片机中断服务程序里，先清除中断标志位，读取RAM占用标志为空时，从RAM里取走数据，将数据根据对应的协议封装成包，返回到上位机。

②数据传输

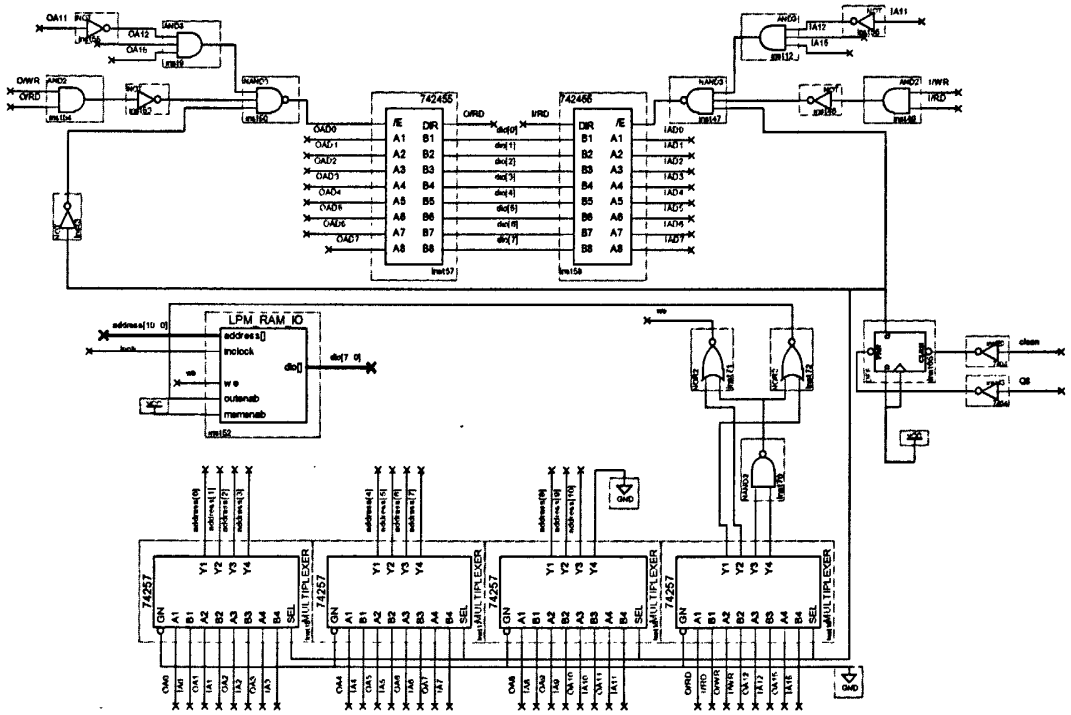


图3.15　通用接口模块第二部分

Fig.3.15 The second part of the Common Interface Module

图3.14和图3.15合成完整的通讯接口模块的框图，现结合图3.15介绍RAM的双向存取数据功能。RAM双向地址线由四2选1数据选择器74HC257控制，当SEL引脚为低时，A通道的数据传到输出端，即接口板的地址线控制RAM；SEL为高时，A通道的数据传到输出端，即主板的地址线控制RAM。而SEL的信号是通过D触发器inst65控制的，inst65的清零和置位端分别经反向器连接至Oflag和Iflag,也就是说，若接口板要占用RAM，会通过

置0flag位为高，使inst65输出低电平，从而使7425的SEL=0，选择由接口板的地址线控制RAM。在此，为使inst65的清零和置位有效进行，务必保证两个信号不能同时有效，否则D触发器输出状态不会翻转。

RAM的8位数据总线经过742455缓冲器inst57接至接口板数据总线，同时经过缓冲器inst58接至主板数据总线，具体走向由两个缓冲器的使能端/E控制。742455是从函数库里调用双向总线驱动器件，没有锁存功能。方向控制管脚DIR=0时，数据由B传向A，DIR=1时，由A传向B；使能端/E由三输入的与非门控制，仅当三个输入均为高时，与非门的输出才为低，使能742455。以与测试仪主板相连的缓冲器inst58为例，分析控制/E的与非门inst47的三个输入，从上而下，第一个输入为高的条件是IA15=1，IA12=1，IA11=0，确定了RAM相对于主板CPU的外围地址是9000H~9FFFH；第二个输入为高的条件是主板的CPU进行一次读写操作；第三个输入信号为高的条件是D触发器inst65输出为高，即要使Iflag=1，也就是说主板要通过对标志信号置高占用RAM。

与接口板相连的742455缓冲器inst57工作原理同上，当接口板单片机置高0flag位占用RAM，并对以9000H开始的地址进行一次读或写操作时，inst57的使能端/E有效，RAM与接口板数据可以双向通讯，传输方向由74255的DIR引脚信号决定。

最后，分析RAM的读写控制线。写信号we由或非门inst72控制，读信号outenab由或非门inst71控制，仅当或非门两输入为0时，才输出有效的读、写高电平信号。A15=A12=1时，满足或非门的一个输入为低，这个条件决定了只有对9000H～9FFFH地址操作时，才可能产生RAM的读、写信号；当SEL=0时，inst71另一个输入是O/RD，即接口板产生有效的读信号时，inst71就会输出接口板对RAM的读信号，当SEL=1时，读信号we在测试仪主板I/RD的控制下对RAM读取。读信号outenable的控制与写信号类似，不再赘述。

这种双 MCU 通讯的方法的优点是命令和数据分开传输，握手严谨，传输准确，误差率低，速度比较快。而且充分利用 FPGA 内部资源，设计方法灵活，完全根据用户需要自行设计。对于不同的 MCU 来说，原理都是一样的，只需略微改动电路模块即可。

3.3.4 GPIB 接口与 SCPI 协议

为了组建现代化、标准化的自动测试系统，本测试仪通过各种接口如 RS232、GPIB、TCP/IP、将控制器和仪器组合起来进行通信，上位机可以发送 SCPI 到仪器实现标准化控制，至于软件编程环境本设计选用的是 LabWindows/CVI。本文将以 GPIB 接口为例讲解晶振与时间继电器自动测试系统[25]的流程。

(1)GPIB 接口简介

GPIB（General Purpose Interface Bus）自动测试总线技术于 1978 年正式定为国际测试总线标准[26]，已有近 30 年的历史。GPIB 测试总线[27]方便易用，可灵活组建各种功能的自动测试系统，获得国内外的广泛应用，至今方兴未艾。近年来虽然出现了 VXI、PXI 等测试总线，但 VXI 总线测试系统成本较高，特别在微波波段目前还没有更多的

VXI 微波模块可供选用。因此，目前的自动测试系统[28]多数都采用 VXI/GPIB 混合结构测试系统，GPIB 总线仍具有很强的生命力。

GPIB 接口高速传输性能以及完整的控制协议，使得基于 GPIB 接口的数据采集，温度监控，波形测量等便携式设备获得了广泛的应用。采用面向对象的软件编程使系统功能易扩展和维护，增强了它的生命力。各种程控仪器与计算机通过 GPIB 接口相连接，让计算机能同时控制这些程控仪器，并发出各种程控命令，接收数据、字节等[29]。GPIB 接口结构简单，通用性能强，使用灵活方便，

GPIB 接口板硬件原理图如图 3.16，接口电路由 89C51、74LS373、CPLD、SN75160、SN75162 等芯片组成。CPLD 接在微处理器与母线收发器之间，SN75160 和 SN75162 为接口电路的母线收发器。这两个芯片将输入的信号驱动到 GPIB 母线所要求的驱动电流，并控制信号传输的方向，它们的输出直接连接到 GPIB 母线，保证了 GPIB 接口物理性能的实现。

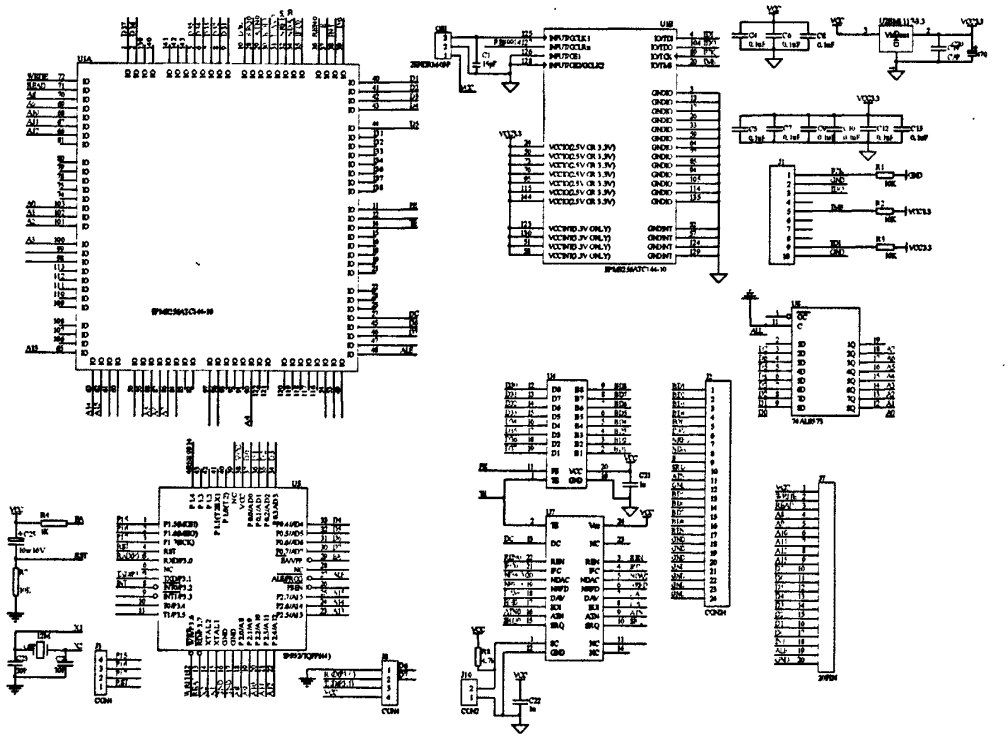


图 3.16　CPLD 实现的 GPIB 接口电路

Fig.3.16 The GPIB interface circuit based on CPLD

在本设计中 CPLD 实现的 GPIB 控制器芯片接口功能是通过 8051 单片机读取/写入其内部寄存器来实现的。CPLD 无内部时钟发生器，本电路通过外接 8MHz 有源晶振，产生 8MHz 时钟与 CPLD 的 CLK 端相连；采用 89C51 单片机，74LS373 作为其地址锁存器。CPLD 实现的 GPIB 控制器芯片的寄存器选择端 RS0-RS2 通过地址锁存器 74LS373 的 Q0-Q7 与单片机的 P0.0-P0.7 相连，这样就可以通过改变地址来选择对 CPLD 内部寄存器的操

作。8051 的中断端直接与控制器芯片的 INT 中断申请线相连，使控制器芯片可以通过中断方式实现接口功能。20 引脚的插针 J7 与测试仪预留的通讯端口相对应，工作时直接用排线连接即可。

(2) SCPI 解析模块

随着测量技术和仪器技术的发展，越来越多的测量设备加入到智能标准测试仪器行列中。智能仪器的特征之一是带有标准的接口，如 RS232、GPIB 以及 VXI 等。通过这些标准的接口，不同厂家、不同功能的仪器可以被集成到同一个测试系统[30]中，也就是说具有了硬件上的兼容性。但是，各厂家功能语言却不具兼容性，相同的命令在不同厂商提供的仪器中具有不同的意义，系统开发变得混乱。1990 年 4 月国际上九家仪器公司提出了程控仪器标准命令 SCPI。

SCPI 是一种基于 ASCII 码的仪器语言，供测试和测量仪器使用，其目的就是为了减少开发应用程序的时间，它要求各仪器生产商必须遵守该标准，程序员就不必再针对某一种仪器学习指令，保证了仪器系统间的兼容性和易用性[31]。对厂商来讲，SCPI 大大减小了厂商为不同仪器设计命令集的负担，还使仪器固件的重用成为可能；对用户来讲，SCPI 简单易学，用户可以选择自己熟悉的开发工具进行开发，缩短了开发的时间，测试程序更为易懂并且维护简单，大大增加了仪器的可互换性。

仪器接收到 SCPI 消息后进行响应：接收字符串消息、词法分析、语法分析、中间代码生成、优化和目标代码生成，语法分析模块的性能直接影响到程控执行效率。为了进一步简化仪器内语法分析模块、提高程控执行效率，本设计在接口电路中加入解析模块的思想，可将控制器发送到仪器的 SCPI 消息即复杂的 ASCII 码字符串转变为简单的二进制代码。

解析模块的实现程序烧写在图 3.16 的单片机里，单片机采用 AT89，其内部存储器为 64K，满足 SCPI 解析过程中数据库的建立、查找等过程要求。解析方法采用顺序查表法，所查的表是一个数据库，此数据库将所有的 SCPI 命令与唯一的二进制代码对应起来，具体的二进制代码值由自己分配。

GPIB 接口板中的单片机工作流程如下：首先，对 CPLD 进行初始化，通过软件配置来实现专用 GPIB 接口芯片的功能；上位机发送 SCPI 消息时，通过 GPIB 协议接收并保存，由于 SCPI 消息实质为 ASCII 码的字符串，单片机接收到后将其保存在定义好的一个字符数组即可；然后，进行格式修正和格式判断，如果是符合条件的 SCPI 消息，则递交给查找程序，去定义好的数据库里找到对应的二进制代码；最后，将解析结果即解析后的字符串发送到仪器以控制仪器。采用此解析模块将大大简化仪器设计者的软件工作，既能实现仪器语言标准化又能提高仪器对远程控制的响应速度。

3.4 外围电路设计

3.4.1 键盘接口电路设计

键盘接口电路如图 3.17 所示。键盘由串入并出移位寄存器 74HC595、连接到单片机的 4 个 IO 口共同组成。当单片机扫描键盘时，把扫描数据通过 PB4 送到 74HC595 的串行数据输入端，保证了并行输出端在任一时刻只一个端口为 0，这样，当某一键盘按下时，PB5、PB6 将有一个变为低电平，通过单片机读入判断出对应的行，行与列信号的交汇键就是按下的键。将键盘值读入单片机，从而实现对键盘动态扫描，实时将键盘命令交单片机处理。

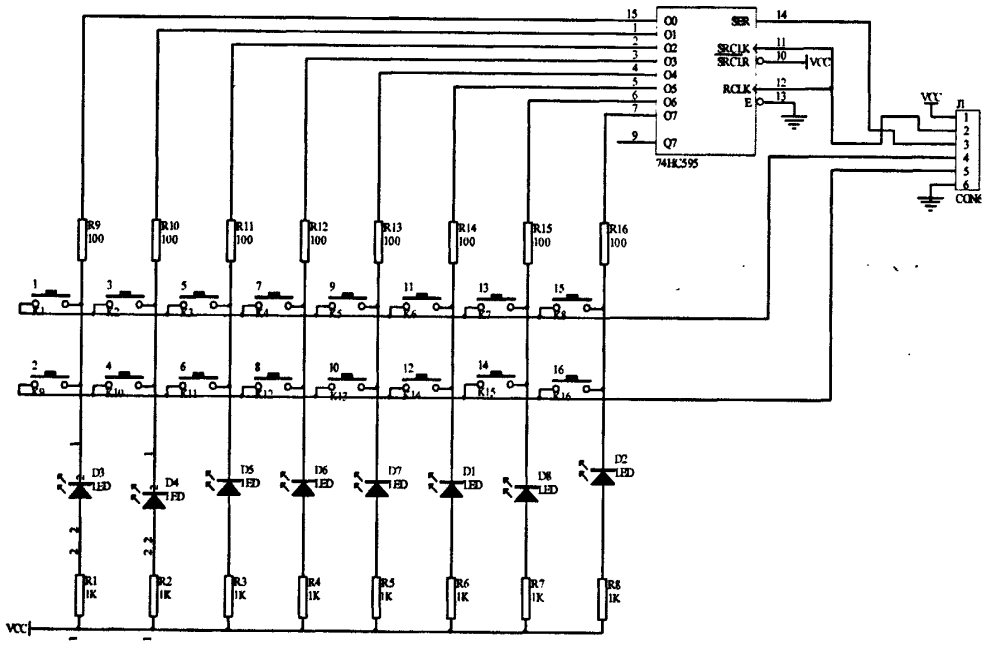


图 3.17 键盘接口电路

Fig.3.17 The circuit of keyboard interface

串行转换成并行的芯片采用 PHILIPS 公司生产的 74HC595, DIP 封装，它是高速的硅结构的 CMOS 器件， 兼容低电压 TTL 电路，遵守 JEDEC 标准。74HC595 具有一个 8 位移位寄存器和一个存储寄存器，三态输出功能。 移位寄存器和存储寄存器是分别的时钟。数据在 SHcp 的上升沿输入移位寄存器，在 STcp 的上升沿由移位寄存进入存储寄存器中去。如果两个时钟连在一起，则移位寄存器总是比存储寄存器早一个脉冲。 移位寄存器有一个串行移位输入（Ds），和一个串行输出（Q7'），和一个异步的低电平复位，存储寄存器有一个并行 8 位的，具备三态的总线输出，当使能 OE 时（为低电平），存储寄存器的数据输出到总线。工作时序如图 3.18。

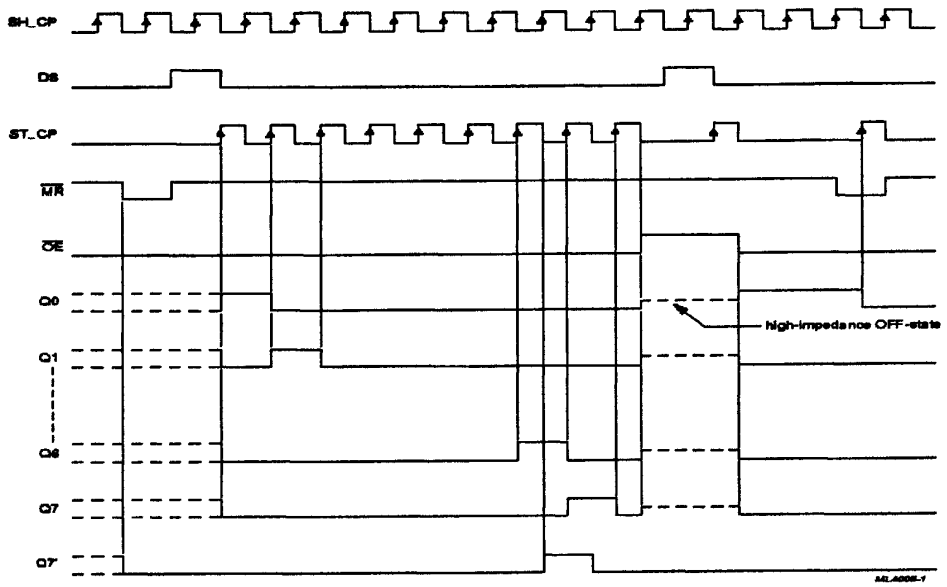


图 3.18 74HC595 时序图

Fig.3.18 The timing diagram of 74HC595

3.4.2 液晶显示电路设计

本方案的显示部分采用 5.7 英寸的 HS320240D 图形点阵模块,内置功能强大的 SED1335F 作为控制器。液晶显示器 HS320240D 的对外接口实质上就是控制器 SED1335F 与单片机进行间接连接，SED1335F 是 EPSON 公司生产的一款图形及字符显示控制器，适用于中等规模的点阵型液晶显示器。它能够提供液晶显示驱动器及外部显示存储器所需的全部控制信号，并且它还有一个内置的字符库，因此只需极少的外部器件就可获得一个组织灵活的低功耗显示[32]系统。

功能特征：传输数据迅速，具有强大的作图功能，支持文本显示，图形显示以及图形和文本混合显示；具备简捷的 MPU 接口和功能齐全的控制指令集，可编程实现控制显示光标及移动，灵活的滚动显示，字符与图形的两层复合显示或三层图形的复合显示；有丰富可选的存储器系统，支持 64K 内存，其中包括 4K 的用户自定义字符库和 60K 的显示内存，内部 CGROM 固化有 160 种 5×7 点阵的字符，支持外部字符 ROM 或 RAM，可选择 8*8 或 8*16 字符字体，并且允许 ROM 与 RAM 字型共同使用。

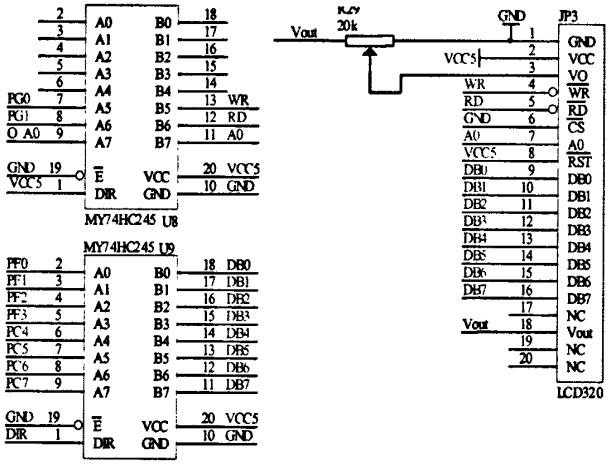


图 3.19 LCD 接口电路

Fig.3.17 The circuit of LCD interface

硬件接口电路如图 3.19 所示。由于 HS320240D 需要 5V 的电压驱动，而来自 CPU 和 FPGA 的控制信号与数据信号均是 3.3V,所以需要两片 74HC245 作电平转换。

74HC245:三态八总线收发器，可将信号功率放大。20 管脚插针封装的 74HC245，具备数据双向传输功能，第 1 脚：DIR，为输入输出端口转换用，DIR=“1”（高电平）时，信号由“A”端输入“B”端输出，DIR=“0”（低电平）时信号由“B”端输入“A”端输出。第 2~9 脚：“A”信号输入输出端，A1=B1、、、、、A8=B8，A1 与 B1 是一组，如果 DIR=“1”G=“0”则 A1 输入 B1 输出，其它类同。如果 DIR=“0”G=“0”则 B1 输入 A1 输出，其它类同。第 11~18 脚：“B”信号输入输出端，功能与“A”端一样，不在描述。第 19 脚 G，使能端，若该脚为“1”A/B 端的信号将不导通，只有为“0”时 A/B 端才被启用，该脚也就是起到开关的作用。第 10 脚 GND，电源地。第 20 脚 VCC，电源正极。

第一个 74HC245 即 U8 传输读、写、及片选信号，完成指令从单片机到 LCD 的单向传输，DIR 管脚可直接接高，使数据由“A”端输入“B”端输出。LCD 上除了显示测量结果与误差，还要将测晶振的标称值或被测继电器设定的闸门时间通过按键设置上去，以增加对比使显示更直观。为了将设定值和测量值进计算，要读回按键在 LCD 上的设置值，所以，第二个 74245 即要完成数据的双向传输，方向脚由 FPGA 里引出的 DIR 信号控制，当 DIR=1 时，单片机往 LCD 发送显示数据；当 DIR=0，读回被测对象的设置值并存储，等待与测量结果进行计算，测试结束时，送出测量结果、绝对误差、相对误差的值。

LCD 引出 20 根插针与单片机通讯，DB0~DB7 为 8 位数据线；VO、Vout 为背光调节管脚，调节滑动器 R29，可以看到背景色由白到蓝的渐变；$\overline{WR}$、$\overline{RD}$是对 LCD 的读写管脚； A0 是数据和指令控制脚，LCD 的读写有命令和数据之分，在对 LCD 操作时，命令字地址和数据地址是分开的，当 A0＝0 时，所写的数据为命令字，当 A0＝1 时，

所写的数据是要显示的数据代码。

3.4.3 JTAG 及 AS 配置电路设计

本系统提供两种配置方法：

(1) 调试时，使用运行在计算机上的 Quartus II 软件，通过 JTAG 电缆连接到目标板上 10 针 JTAG 接口直接下载配置数据到 FPGA。用户随时可以进行 JTAG 模式的配置；但要注意 JTAG 配置模式是直接对 FPGA 中 SDRAM 单元编程，掉电后数据丢失，因此重新上电要重新下载。

(2) 在脱机运行时，采用串行配置器件 EPCS4 进行主动配置(AS)。主动串行配置模式(AS)是将配置数据存储在串行配置器件 EPCS4 中，在每次系统上电时 FPGA 会自动使用 EPCS4 中的配置数据进行配置。在脱机运行之前，要事先将配置数据通过下载电缆写入 EPCS4 中。

用户可以通过 FPGA 上的 MSEL0、MSEL1 两个引脚的状态来选择表 3.9 中所列模式中的一种来进行 FPGA 配置，由于目标板只使用主动配置和 JTAG 配置模式，所以可将 MSEL0 和 MSEL1 引脚接地。

表 3.9 配置模式设置

Table3.1 Configuration mode setting

MSEL0	MSEL1	配置模式
0	0	AS
0	1	PS
0	0 或 1	JTAG

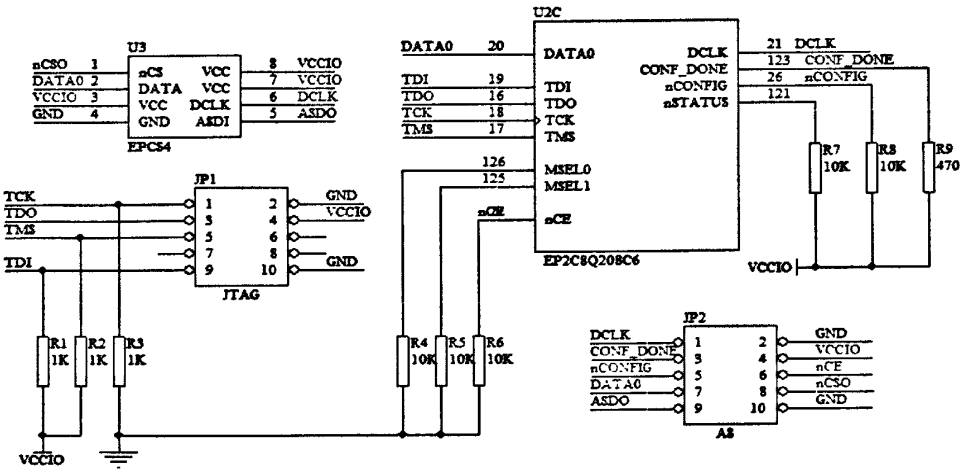


图 3.20 FPGA 配置电路

Fig.3.20 The configuration circuit of FPGA

3.4.4 电源模块设计

为防止电源的干扰引起主芯片工作的不稳定，本设计中的单片机 LM3S8962、FPGA EP2C8、有源恒温晶振分别有单独的供电模块。

(1) LM3S8962 供电电路设计

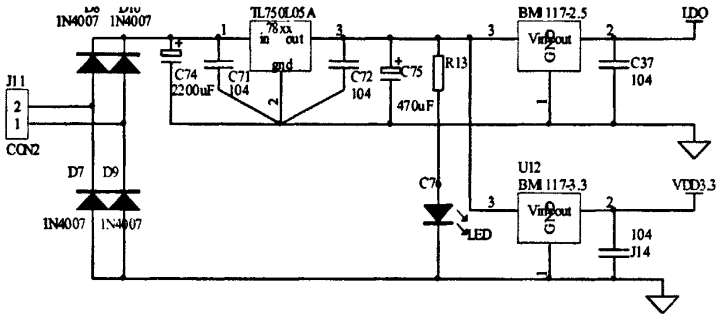


图 3.21 LM3S8962 供电电路

Fig.3.21 The power supply circuit for LM3S8962

如图 3.21 所示，220V 交流电源经变压、整流、滤波后，由一片 78L05 三端稳压器向系统提供＋5V 电源电压信号。在＋5V 电源电压的基础上通过一片 BM1117－2.5 三端稳压器和一片 BM1117－3.3 三端稳压器向单片机分别提供 2.5V 和 3.3V 的电源电压信号。电容 C74、C75 滤去低频干扰，电容 C71、C72 滤去高频干扰，共同起到去除电源纹波的作用。发光二极管 C76 作为电源指示灯，给系统上电后，电源指示灯就会被点亮。

(2) FPGA 供电电路设计

因为 FPGA 对电源要求更敏感，稍有干扰，程序就会跑飞，而只有在重新上电时，才能由配置芯片重新配置，不易复位，所以采用开关稳压集成电路 LM2576 作为电压调节和稳压器件，它具有可靠的工作性能、较高的工作效率和较强的输出电流驱动能力，从而为 FPGA 的稳定、可靠工作提供了强有力的保证。在 LM2576+5V 的输出基础上通过一片 BM1117－1.2 三端稳压器和一片 BM1117－3.3 三端稳压器向 EP2C8 分别提供 1.2V 和 3.3V 的电源电压信号。

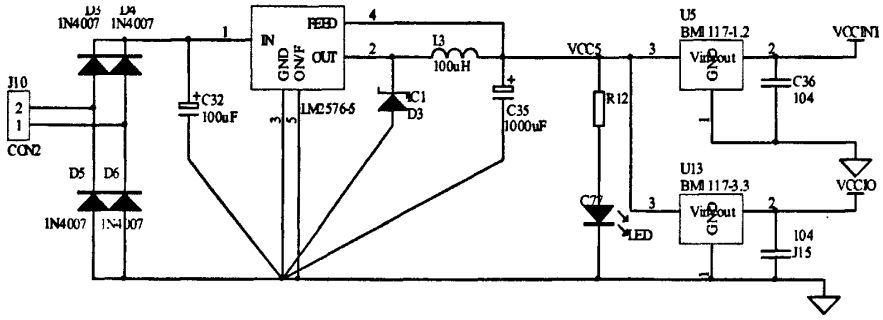


图 3.22 FPGA 供电电路

Fig.3.22 The power supply circuit for FPGA

(3) 恒温晶振供电电路设计

为使晶振输出更稳定，采用双电源+15V 供电。其中 LM317 系列稳压器输出连续可调的正电压，稳压器内含有过流、过热保护电路。R_{52} 与 W1 组成电压输出调节电路，输出电压 $V_O \approx 1.25(1+W1/R_{52})$，$R_{52}$ 为精密可调电位器。电容 C_{82} 与 W1 并联组成滤波电路，以减小输出电压的纹波电压。二极管 D18 的作用是防止输出端与地短路时，损坏稳压器。

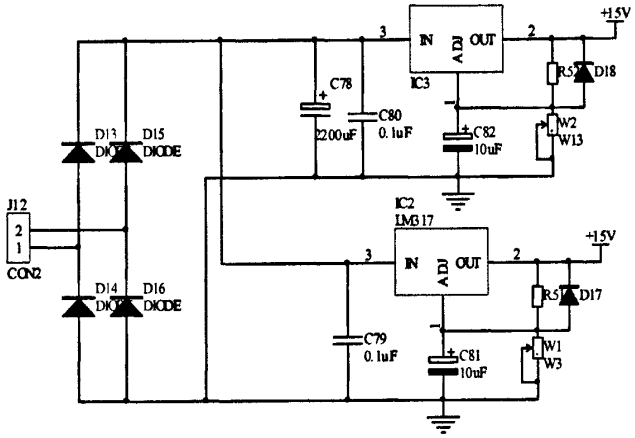


图 3.23 恒温晶振供电电路

Fig.3.23 The power supply circuit for OCXO

4 系统的软件设计

在设计中把控制能力强的单片机与现场可编程芯片结合起来，实现了对时间继电器和频率信号的测量。单片机采用 C 语言编程，可以精确地控制闸门的开启和关闭，同时完成对本地/远地键盘、液晶显示和 FPGA 的管理和控制。现场可编程芯片(EP2C8)具有编程方便、速度快、集成度高、价格低、可靠性好的优点，从而使系统研制周期大大缩短，提高了产品的性能价格比。采用混合描述方式在 FPGA 内实现了门控电路、时基产生电路、计数器电路、本地/键盘接口和 LCD 显示器驱动电路的编程设计。流程如图 4.1。

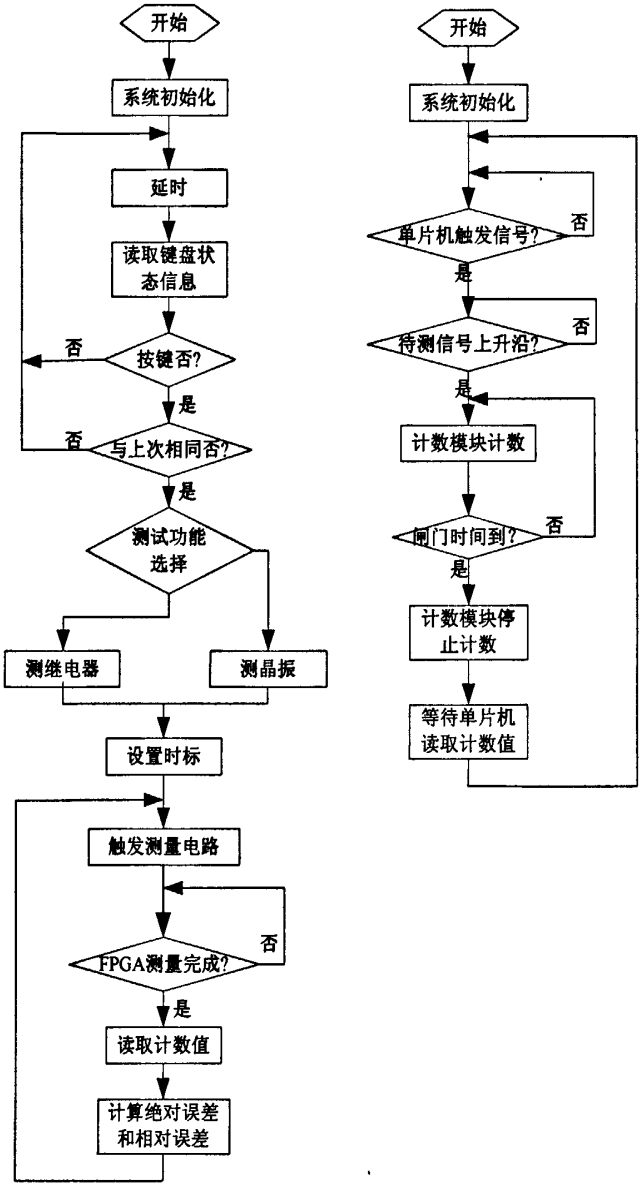


图 4.1 主程序流程图

Fig.4.1 Flowchart of main program

以 LM3S8962 和 FPGA 为核心，单片机主要完成键盘扫描,LCD 显示,控制、回读并处理 FPGA 传来的数据。FPGA 则完成分频，计数，控制计数时机，计数完成后输出标志信号等待单片机读取数。

4.1 基于 LM3S8962 的程序开发

4.1.1 LM3S8962 开发环境介绍

本项目的下位机软件开发环境使用 IAR EWARM 4.42 版，IAR Embedded Workbench for ARM（下面简称 IAR EWARM）是一个针对 ARM 处理器的集成开发环境[33]，它包含项目管理器、编辑器、C/C++编译器和 ARM 汇编器、连接器 XLINK 和支持 RTOS 的调试工具 C-SPY。在 EWARM 环境下可以使用 C/C++[34]和汇编语言方便地开发嵌入式应用程序。比较其他的 ARM 开发环境，IAR EWARM 具有入门容易、使用方便和代码紧凑等特点。

目前 IAR EWARM 支持 ARM Cortex-M3 内核的最新版本是 4.42a，该版本支持 Luminary 全系列的 MCU。为了方便用户学习评估，IAR 提供一个限制 32K 代码的免费试用版本。用户可以到 IAR 公司的网站 www.iar.com/ewarm 下载。

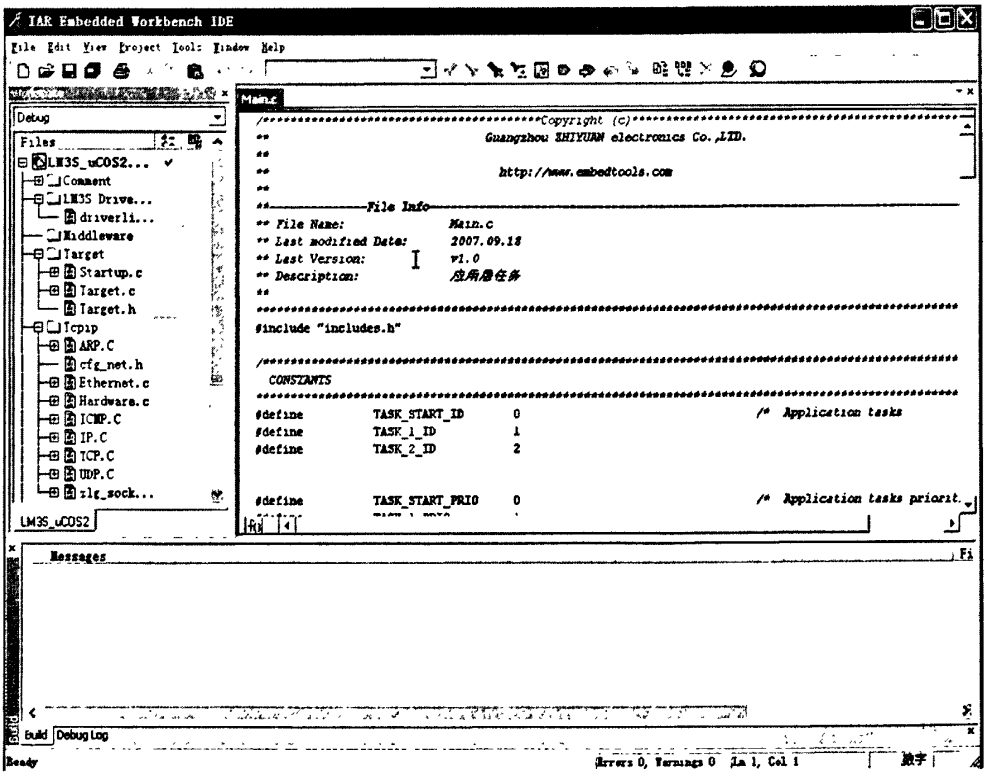


图 4.2 IAR 集成开发环境运行界面

Fig.4.2 IAR integrated development environment

IAR EWARM 的运行界面如图 4.2 所示，该软件同大多数的 ARM 开发工具使用方法基本一样，在此不再具体描述，并且它还支持 LM LINK 调试器调试，使我们调试程序方便。

4.1.2 单片机主程序设计

单片机以总线的方式连接到EP2C8。其主程序主要是进行FPGA 配置、仪器的初始化、键盘扫描、闸门开/关控制、计数器控制、管理LCD显示、中断响应、数据处理等工作。具体流程如图3—2所示。系统初始化后，主程序循环执行键盘扫描子程序，当某键按下时，程序自动跳转到相应的子程序执行其功能，然后返回继续执行键盘扫描程序。

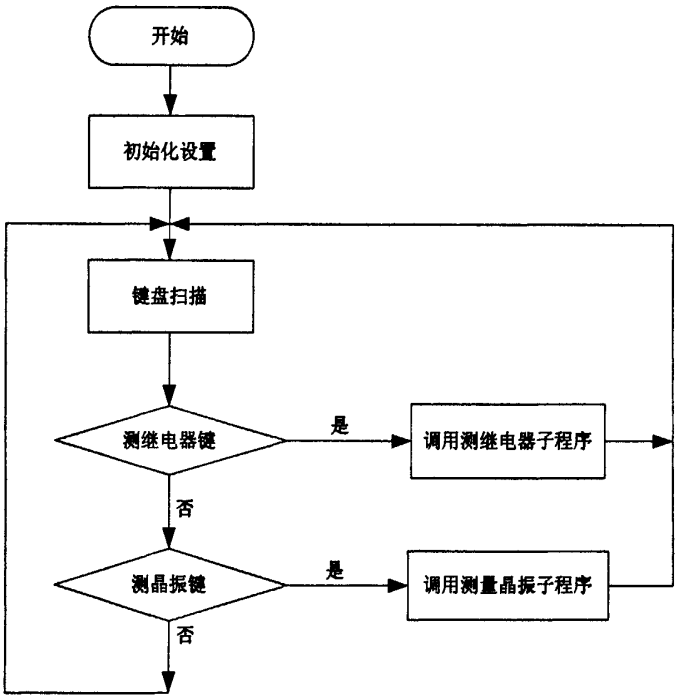


图4.3 主程序流程图

Fig. Flow chart of the main program

主程序的详细代码见附录 B，其逻辑流程设计既能实现本地控制，又能响应上位机的远程控制。实现了友好的人机界面交互功能。

4.1.3 键盘扫描子程序设计

```
UCHAR keyRead (void)
{
   UCHAR i,a=0;
    for(i=0;i<8;i++)
    {
      PinSetBit(SER);
```

```
        PinClrBit(SCL);
        PinSetBit(SCL);

    }
    PinClrBit(SER);
    PinClrBit(SCL);
    PinSetBit(SCL);

    for(i=0;i<9;i++)
    {
        if(0==GPIOPinRead(KEY_PORT,FLAG1))
        {
            if (i==1)       a=16;
            else if (i==2)  a=14;
            else if (i==3)  a=4;
            else if (i==4)  a=8;
            else if (i==5)   a=12;
            else if (i==6)  a=6;
            else if (i==7)  a=10;
            else if (i==8)  a=2;
            else {;}
        }
        if(0==GPIOPinRead(KEY_PORT,FLAG2))
        {
            if (i==1)        a=15;
            else if (i==2)  a=13;
            else if (i==3)  a=3;
            else if (i==4)  a=7;
            else if (i==5)   a=11;
            else if (i==6)  a=5;
            else if (i==7)  a=9;
            else if (i==8)  a=1;
            else {;}
        }
        PinSetBit(SER);
        PinClrBit(SCL);
        PinSetBit(SCL);

    }
    return a;
}
```

变量 a 被赋予不同的键值，当检测到相应的键被按下后，此键值将被返回给主函数，主函数据此进入不同的分支处理。

4.1.4 使用 LM3S8962 的 IO 口模拟总线

```
/*****************************************************************************
** Function name:          GPIOOutPut_Init
** Descriptions:           GPIO 作为输出的初始设置,针对数据口 GPIOE,GPIOF,地址和控制口已设
**                         置为输出
** input parameters:       无
** output parameters:      无
** Returned value:         无
*****************************************************************************/
void GPIOOutPut_Init()
{
   /*  设置 GPIOC,GPIOF 为输出              */
   GPIODirModeSet(GPIO_PORTC_BASE, GPIOCH, GPIO_DIR_MODE_OUT);
   GPIODirModeSet(GPIO_PORTF_BASE, GPIOFL, GPIO_DIR_MODE_OUT);
    /*  设置 GPIOD 的驱动强度和类型    */
   GPIOPadConfigSet(GPIO_PORTC_BASE, GPIOCH,
   GPIO_STRENGTH_4MA, GPIO_PIN_TYPE_STD);
   GPIOPadConfigSet(GPIO_PORTF_BASE, GPIOFL,
   GPIO_STRENGTH_4MA,GPIO_PIN_TYPE_STD);
}
/*****************************************************************************
** Function name:          GPIOInPut_Init
** Descriptions:           GPIO 作为输入的初始设置
** input parameters:       无
** output parameters:      无
** Returned value:         无
*****************************************************************************/
void GPIOInPut_Init()
{
    /*  设置 GPIOD 口为输出              */
    GPIODirModeSet(GPIO_PORTC_BASE,GPIOCH,GPIO_DIR_MODE_IN);
    GPIODirModeSet(GPIO_PORTF_BASE, GPIOFL, GPIO_DIR_MODE_IN);
    GPIOPadConfigSet(GPIO_PORTC_BASE, GPIOCH,
    GPIO_STRENGTH_4MA, GPIO_PIN_TYPE_STD);

    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIOFL,
    GPIO_STRENGTH_4MA,   GPIO_PIN_TYPE_STD);
 }
/*****************************************************************************
** Function name:          WriteByte
** Descriptions:           由译码地址选中相应器件 送入控制字
** input parameters:       Address:  数据写入的地址
```

```
** Data:                  要写入的字节数据
** output parameters:     无
** Returned value:        无
**addr+小写 L+_+数字
**与 GPIOOutPut_Init()配合使用
*********************************************************************************/
void WriteByte (UINT   Address, UCHAR   Data)
{
    UCHAR addrh_1,addrh_2,addrl_1,addrl_2;
    UCHAR datah,datal;

    GPIOOutPut_Init();

    addrh_1=(Address>>8)&0xFC;
    addrh_2=((Address>>8)&0x03)<<2;
    addrl_1=Address&0xFC;
    addrl_2=Address&0x03;

    datah=(Data&0xf0);
    datal=Data&0x0f;

    GPIOPinWrite(GPIO_PORTD_BASE, AddrH_1, addrh_1);
    GPIOPinWrite(GPIO_PORTE_BASE, AddrH_2, addrh_2);
    GPIOPinWrite(GPIO_PORTA_BASE, AddrL_1, addrl_1); /*向 GPIOA 口地址线写入地址 */
    GPIOPinWrite(GPIO_PORTE_BASE, AddrL_2, addrl_2);

    GPIOPinWrite(GPIO_PORTF_BASE, GPIOFL,datal);        //写数据
    GPIOPinWrite(GPIO_PORTC_BASE, GPIOCH,datah);

    GPIOPinWrite(GPIO_PORTG_BASE, nRD,nRD);          /*  将读信号 nRD 置高     */
    GPIOPinWrite(GPIO_PORTG_BASE, nWR, ~nWR); /*使能写信号 ,初始配置时已经设为高 */
    GPIOPinWrite(GPIO_PORTG_BASE, nWR, nWR);
  }
/*********************************************************************************
** Function name:          ReadByte
** Descriptions:           从相应地址读回数据
** input parameters:       Address:  数据读出的地址
** Returned value:         Data
**addr+小写 L+_+数字
**与 GPIOOutPut_Init()配合使用
*********************************************************************************/

UCHAR ReadByte (UINT   Address)
```

```
{
    UCHAR   Data;
    UCHAR addrh_1,addrh_2,addrl_1,addrl_2;
    UCHAR datah,datal;

    GPIOInPut_Init();
    addrh_1=(Address>>8)&0xFC;
    addrh_2=((Address>>8)&0x03)<<2;
    addrl_1=Address&0xFC;
    addrl_2=Address&0x03;

    GPIOPinWrite(GPIO_PORTD_BASE, AddrH_1, addrh_1);
    GPIOPinWrite(GPIO_PORTE_BASE, AddrH_2, addrh_2);
    GPIOPinWrite(GPIO_PORTA_BASE, AddrL_1, addrl_1);/*向 GPIOA 口地址线写入地址    */
    GPIOPinWrite(GPIO_PORTE_BASE, AddrL_2, addrl_2);

    GPIOPinWrite(GPIO_PORTG_BASE, nWR, nWR);
    GPIOPinWrite(GPIO_PORTG_BASE, nRD,~nRD);

    datal=GPIOPinRead(GPIO_PORTF_BASE, GPIOFL);
    datah=GPIOPinRead(GPIO_PORTC_BASE, GPIOCH);
    Data=datah|datal;

    GPIOPinWrite(GPIO_PORTG_BASE, nRD,nRD);
    return(Data);
}
```

将用到的地址总线与数据总线预先做好初始化工作，调用时，仅向形参传送地址，就可以将相应位置处的数据通过模拟的数据总线读到单片机里。

4.2 基于 FPGA 的程序

4.2.1 FPGA 开发环境

(1) Quartus II 概述

Quartus II 是 Alera 提供的 FPGA/CPLD 集成开发环境[35]，Alera 是世界最大可编程逻辑器件供应商之一。Quartus II 在 21 世纪初推出，是 Alera 前一代 FPGA/CPLD 集成开发环境 MAX+PLUS II 的更新换代产品，其界面友好，使用便捷。在 FPGA 上可以完成开发的整个流程，它提供了一种与结构无关的设计环境，使设计者能方便地进行设计输入、快速处理和器件编程。

Alera 的 Quartus II 提供了完整的多平台设计环境，能满足各种特定设计[36]的需要，也是单芯片可编程系统（ SOPC ）设计的综合性环境和 SOPC 开发的基本设计工具，并

为 Alera DSP 开发包进行系统模型设计提供了集成综合环境。Quartus II 设计工具完全支持 VHDL、Verilog 的设计流程，其内部嵌有 VHDL、Verilog 逻辑综合器。Quartus II 也可以利用第三方的综合工具，如 Leonardo Spectrum，并能直接调用这些工具。同样 Quartus II 具备仿真功能，同时也支持第三方的仿真工具，如 Model Sim 等。此外 Quartus II 与 MATLAB 和 DSP Builder 结合，可以进行基于 FPGA 的 DSP 系统开发，是 DSP 硬件系统实现的关键 EDA 工具。

Quartus II 包括模块化的编译器。编译器包括的功能模块有分析/综合器(Analysis&Synthesis)、适配器(Fitter)、装配器(Assembler)、时序分析器(Timing Analyzer)、设计辅助模块(Dsign Assistant)、EDA 网表文件生成器(EDA Netlist Writer)和编辑数据接口(Compiler Database Interface) 等。可以通过选择 Start Compilation 来运行所有的编译器模块，也可以通过选 Start 择单独运行各个模块。还可以通过选择 Compiler Tool (Tools)菜单，在 Compiler Tool 窗口中运行该模块来启动编译器模块。在 Tool 窗口中，可以打开该模块的设置文件或报告文件，或打开其他相关窗口。

此外，Quartus II 还包含许多十分有用的 LPM(Library of Parameterized Modules)模块，它们是复杂或高级系统构建的重要组成部分，在 SOPC 设计中被大量使用，也可在普通设计文件一起使用。Quartus II 提供的函数均基于器件的结构做了优化设计。在许多实用情况中，必须使用宏功能模块才可以使用一些特定器件的硬件功能。例如各类片上存储器、DSP 模块、PLL 等。Quartus II 编译器支持的硬件描述语言有 VHDL(支持 VHDL'87 及 VHDL'97 标准)、Verilog HDL、AHDL(Alera HDL)， AHDL 是 Alera 公司自己设计、制定的硬件描述语言，是一种以结构描述方式为主的硬件描述语言，只有企业标准。

Quartus II 允许来自第三方的文件输入，并提供了很多 EDA 软件的接口。Quartus II 支持层次化设计，可以在一个新的编辑输入环境中对使用不同输入设计方式完成的模块(元件)进行调用，从而解决了原理图与 HDL 混合输入设计的问题。在设计输入之后，Quartus II 的编译器将给出设计输入的错误报告。Quartus II 拥有性能良好的设计错误定位器，用于确定文本或图形设计中的错误。对于使用 HDL 的设计，可以使用 Quartus II 带有 RTL Viewer 的观察综合后的 RTL 图。在进行编译后，可对设计进行时序仿真。在作仿真前，需要利用波形编辑器编辑一个波形激励文件，用于仿真验证时的激励。编译和仿真经检测无误后，便可以将下载信息通过 Quartus II 提供的编程器下载入目标器件中了。

(2) 硬件描述语言 HDL (Hardware Describe Language) 概述

随着 EDA 技术的发展，使用硬件语言设计 PLD/FPGA 成为一种趋势。目前最主要的硬件描述语言是 VHDL 和 Verilog HDL。 VHDL 发展的较早，语法严格，而 Verilog HDL 是在 C 语言的基础上发展起来的一种硬件描述语言,语法较自由。两种语言的差别并不

大，他们的描述能力也是类似的。

①VHDL 语言简介[37]：

VHDL 的英文全称 VHSIC（Very High Speed Integrated Circuit）Hardware Description Language。1983 年由美国国防部（DOD）发起创建，由 IEEE（The institute of Electrical and Electronics Engineers）进一步发展并在 1987 年作为“IEEE 标准 1076”发布。1993 年被更新为 IEEE 标准 1164 。HDL 的出现是为了适应电子系统设计的日益复杂。若以计算机软件的设计与电路设计做个类比，机器码好比晶体管/MOS 管；汇编语言好比网表；则 HDL 语言就如同高级语言[38]，VHDL 在语法和风格上类似于现代高级编程语言，如 C 语言。但要注意，VHDL 毕竟描述的是硬件，它包含许多硬件特有的结构。

现在 VHDL 被广泛用于[39]：电路设计的文档记录、设计描述的逻辑综合、电路仿真等。VHDL 及自顶向下方在大型数字系统设计中被广泛采用。先用较抽象的语言(行为/算法)来描述系统结构，然后细化成各模块，最后可借助编译器将 VHDL 描述综合为门级。设计过程一般如下：

- 代码编写；
- 由综合器(如 Synplify,Synopsys 等)综合成门级网表；
- 前仿真/功能仿真；
- 布局/布线至某一类 CPLD/FPGA 中；
- 后仿真/时序仿真。

② Verilog HDL 语言简介：

Verilog HDL(Verilog Hardware Description Language)是一种硬件描述语言，可以从算法级、门级到开关级的多种抽象层次对电子电路和系统的行为进行描述。基于这种描述，结合相关的软件工具，可以得到所期望的实际的电路与系统[40]。由于 Verilog HDL 既是机器可读的语言也是人类可读的语言，因此它支持硬件设计的开发、验证、综合和测试；硬件数据之间的通信；硬件的设计、维护和修改。现在，Verilog HDL 已经成为数字系统设计的首选语言，并成为综合、验证和布局布线技术的基础。

Verilog HDLw 从 20 世纪 80 年代由 GDA（Gateway Design Automation）公司最早推出，于 1995 年被接纳为 IEEE 标准，到现被全球范围内的众多设计者所接受，已经经历了 20 多年的时间。它使各种设计工具（包括验证仿真、时序分析、测试分析以及综合）能够在多个抽象层次上以标准文本格式描述数字系统，简单、直观并富有效率。由于其丰富的功能，Verilog HDL 已经成为数字系统设计的首选语言。

Verilog 包含了丰富的内建原语，包括逻辑门、用户定义的原语、开关以及线逻辑。它还具有器件管脚间的时延和时序检查功能[41]。从本质上讲，Verilog 所具有的混合抽象层次由两种数据类型所提供，这两种数据类型是线网（net）和变量（variable）。对

于连续赋值，变量和线网的表达式能够连续地将值驱动到线网，它提供了基本的结构级建模方法。对于过程赋值，变量和网络值的计算结果可以存储于变量当中，它提供了基本的行为级建模方法。一个用 Verilog HDL 描述的设计包含一组模块，每一个模块都包含一个 I/O 接口和一个功能描述。模块的功能描述可以是结构级的、行为级的、也可以是结构级和行为级的混合。这些模块组成一个层次化结构并使用线网进行互连。

使用 Verilog HDL 语言的主要原因之一是，通过代码综合可以采用可编程逻辑器件（CPLD 或 FPGA）或 ASIC 来实现所需的电路[42]。图 4.4 给出了采用 Verilog 进行电路设计的流程。

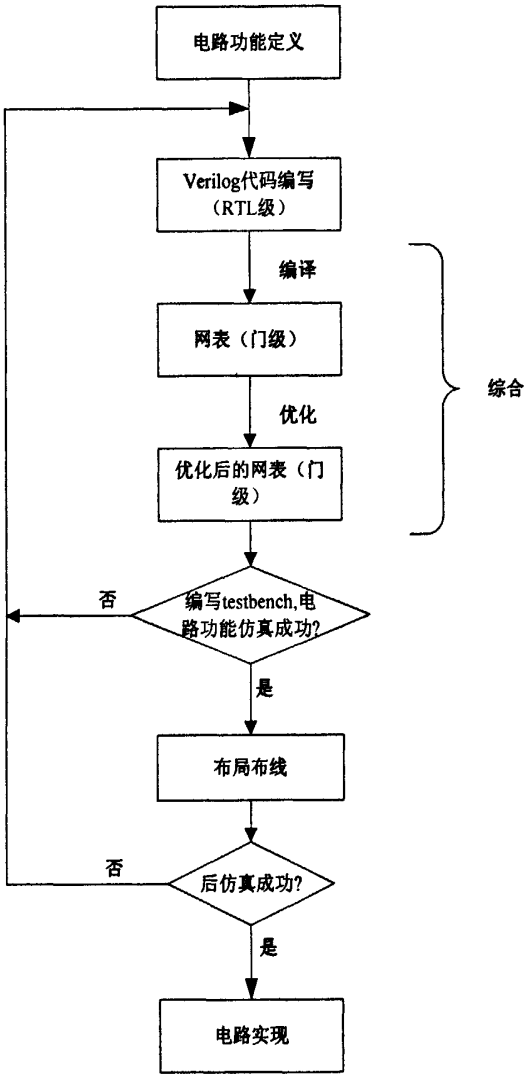


图 4.4　Verilog 设计流程图

Fig.4.4　Design Flow used Verilog

如图 4.4 所示，设计的第一个阶段是编写 Verilog 代码，编写后的代码保存为一个后缀为.v 的文件。代码编与完毕后进入综合价段。综合价段的第一步是进行代码编译，

代码编译过程把寄存器传输级（RTL：Register Transfer Level）的Verilog代码转换成门级网表；综合价段的第二步是优化，主要是根据对电路工作速度和占用硬件资源大小的要求，对门级网表进行优化。在综合之后，可以对设计进行功能仿真。功能仿真通过后，布局布线工具可以在具体的PLD/FPGA器件是对各种电路单元进行布局布线工作，或者调用专用的电路单元库生成ASIC电路。完成布局布线后，可以进行电路的后仿真，仿真结果可以最大限度地接近真实值的结果。完成电路的后仿真后，就可以采用PLD/FPGA专用集成电路实现所需的设计功能了。

4.2.2 FPGA单元模块的实现

(1)分频电路的实现

偶数倍（2N）的分频比较简单，使用一模N计数器模块即可实现，即每当模N计数器上升沿从0开始计数至N-1时，输出时钟进行翻转，同时给计数器一复位信号使之从0开始重新计数，以此循环即可。本设计的十分频电路描述如下：

```
module 10fenpin(clki,clko);
input clki;
output clko=0;
reg clko;
reg [3:0]m;
always @(posedge clki)
begin
    m=m+1;
    if(m>=4'D5)
    begin
    clko=~clko;
    m=0;
    end
end
endmodule
```

奇数倍分频，要使占空比为50%，以如下思路实现：首先进行上升沿触发进行模N计数，计数到某一个值时进行输出时钟翻转，然后经过（N-1）/2再次进行翻转得到一个占空比非50%的奇数n分频时钟。再者同时进行下降沿触发的模N计数，到和上升沿触发输出时钟翻转选定值相同值时，进行输出时钟时钟翻转，同样经过（N-1）/2时，输出时钟再次翻转生成占空比非50%的奇数n分频时钟。两个占空比非50%的n分频时钟相或运算，得到占空比为50%的奇数n分频时钟。本设计的五分频电路描述如下：

```
module fenpin5(clk, clk1x, rst);
input clk;                          //  时钟输入
input rst;
```

```
output clk1x;                                    //5 分频输出
reg clk1xpose;
reg clk1xnege;
reg[2:0] coutpose;
reg[2:0] coutnege;
parameter div1 = 2 , div2 = 4;   // div1  =  5 / 2, div2 = 5 - 1
assign clk1x = clk1xpose | clk1xnege;
always@(posedge clk or negedge rst)
 begin
  if(!rst)
     clk1xpose = 0;
  else if(coutpose == div1)
     clk1xpose = ~clk1xpose;
  else if(coutpose == div2)
     clk1xpose = ~clk1xpose;
  else
     clk1xpose = clk1xpose;
 end
always@(negedge clk or negedge rst)
 begin
  if(!rst)
     clk1xnege = 0;
  else if(coutnege == div1)
     clk1xnege = ~clk1xnege;
  else if(coutnege == div2)
     clk1xnege = ~clk1xnege;
  else
     clk1xnege = clk1xnege;
 end
always@(posedge clk or negedge rst)
 begin
  if(!rst)
    coutpose = 0;
  else if(coutpose == div2)
    coutpose = 0;
  else
    coutpose = coutpose + 1;
 end
always@(negedge clk or negedge rst)
 begin
  if(!rst)
    coutnege = 0;
  else if(coutnege == div2)
    coutnege = 0;
```

```
    else
      coutnege = coutnege + 1;
  end
endmodule
```

div1 为奇数分频除 2 的商，采用上升延和下降延分别触发不同波形，最后叠加的方式产生奇数分频。五分频的仿真波形图 4.5：

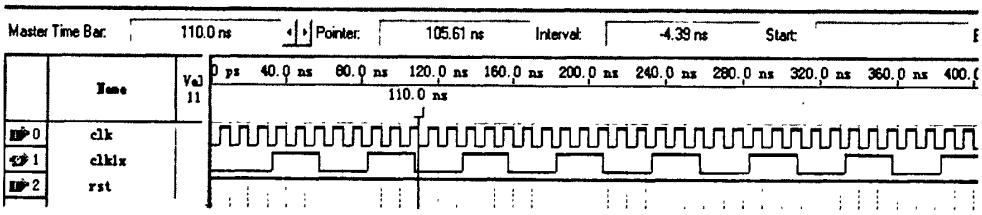


图 4.5 五分频仿真波形图

Fig.4.5 The fifth-frequency simulation waveforms

(2)门控模块逻辑

门控逻辑功能的实现是整个系统的关键，当启动信号有效时等待时间继电器的开门信号有效，当开门信号有效时计数器开始计数，时间继电器的下降沿到来时停止计数，计数结束后标志信号 kk 变为高电平，以此来通知单片机读取，读信号产生一个负脉冲把计数值读走，读操作完毕后，kk 变为低电平，本次测量结束。如果启动信号无效即使时间继电器闸门信号有效计数器也不计数；如果启动信号与门控信号同时有效，则本次不计数，等到下一次被测信号有效时才开始数。用 VHDL 语言描述如下：

```
library ieee;
use ieee.std_logic_1164.all;
entity aa is port(cont,gate,fenq,DATERD:in std_logic;
            count,kk:out std_logic);
end entity aa;
architecture bb of aa is
    signal flag1,flag2,flag3,sedflag1:std_logic;
begin
    P1:process (cont,flag2,flag3)
        begin
            if flag2='1' then
                flag1<='0';
            elsif flag3='1' then
                sedflag1<='0';
            elsif rising_edge(cont) then
                if gate='0' then
                    flag1<='1';
```

```
                else
                    sedflag1<='1';
                end if;
            end if;
        end process P1;

    P2:count<=fenq and gate and (flag1 or flag3);

    P3:process (gate,DATERD)
        begin
            if DATERD='0' then
                flag2<='0';kk<='0';
            elsif falling_edge(gate)then
                if flag1='1'  or flag3='1' then
                    flag2<='1';
                    kk<='1';
                elsif flag3='1' then
                    kk<='1';
                end if;
            end if;
        end process P3;
    P4:process(gate,flag2,sedflag1)
        begin
            if flag2='1' then
                flag3<='0';
            elsif rising_edge(gate) then
                if sedflag1='1' then
                    flag3<='1';
                end if;
            end if;
        end process;
    end architecture bb ;
```

将门控模块的程序仿真运行，结果如图 4.6，可以看出，在各种输入信号的制约下，得到了希望的输出。

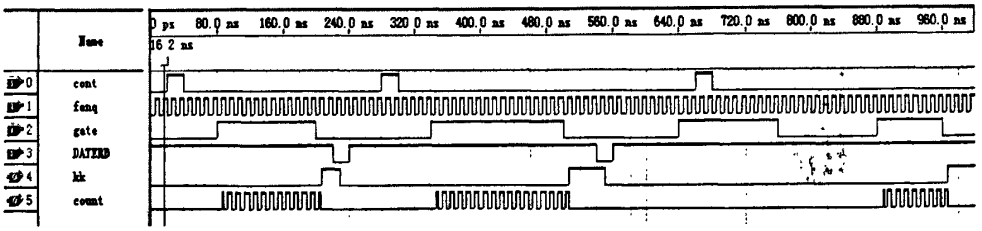


图 4.6　门控模块仿真运行结果

Fig. 4.6　The simulation result of the gate control module

4.3　上位机开发环境 Labwindows/CVI

要通过上位机对仪器进行通信，必然要有控制界面。开发仪器控制界面的应用软件环境的选择可因开发人员的喜好而不同，但最终必须提供给用户一个界面友好、功能强大的应用程序。目前，虚拟仪器系统应用软件开发环境主要包括两种：一种是基于传统的文本语言式的平台，主要是NI公司的LabWindows/CVI[43]，VC++，Delphi等；另一种是基于图形化工程环境的平台，如HP公司的HPVEE，NI公司的LabVIEW等。图形化软件开发虽然比较直观，但功能受到太多的屏蔽。LabWindows/CVI是以ANSI C为核心的交互式虚拟仪器开发环境，它将功能强大、应用广泛的C语言与测控技术有机结合，实现了数据的采集、分析和显示。

另外，它的集成化开发平台、交互式编程方法、丰富的面板功能和库函数等特点，为熟悉C语言的开发人员建立检测系统，自动测量环境，数据采集系统，过程监控系统等提供了一个理想的软件开发环境，是实现虚拟仪器及网络化的快速途径。

4.3.1 虚拟仪器界面模块

图4.7是本测试系统的虚拟仪器界面：

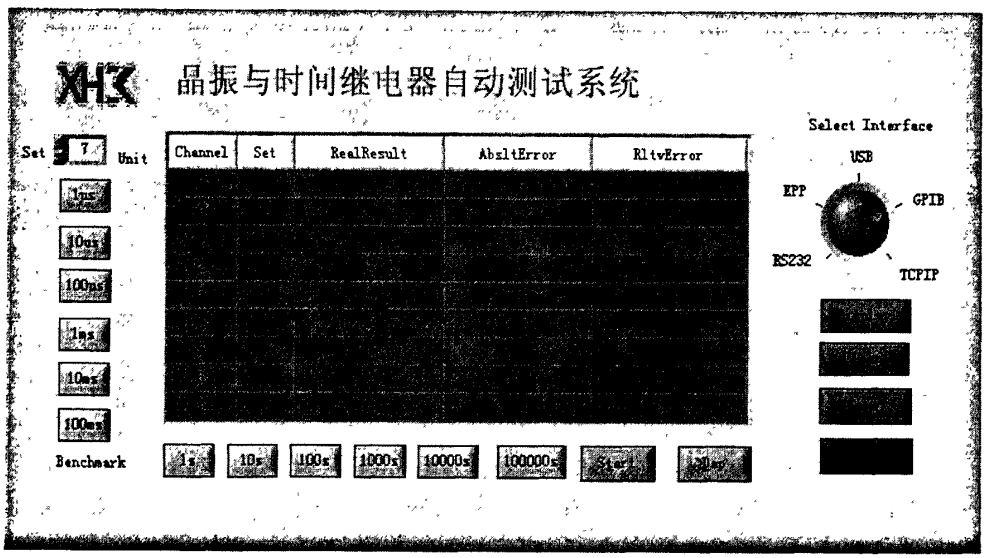


图4.7 上位机虚拟仪器控制界面

Fig.4.7 The virtual instrument control interface Of host computer

可以看出，虚拟仪器反映了传统仪器的真实界面和数据显示的真实状态[44]，使人有一个更加直观的状态。用户可以像操作真实的仪器那样操作虚拟仪器。为了达到设计的界面和传统仪器的前面板相似，Labwindows/CVI本身并没有提供这么多的外观控件，但是我们可以使用它的Picture Command控件，把自己做的按钮图形加载到这个控件中，就可以做出和传统仪器相似的按钮出来。

4.3.2 应用程序设计

(1) 源代码生成与编辑[45]

设计完前面板后，从用户图形界面窗口中选择：Code>>Generate>>All Code…在弹出的对话框中选择最初装入并显示的 Panel 以及哪一个函数作为退出程序的终止函数被调用。点击 OK 按扭，便会弹出如图 4.8 所示的源代码(Code)窗：

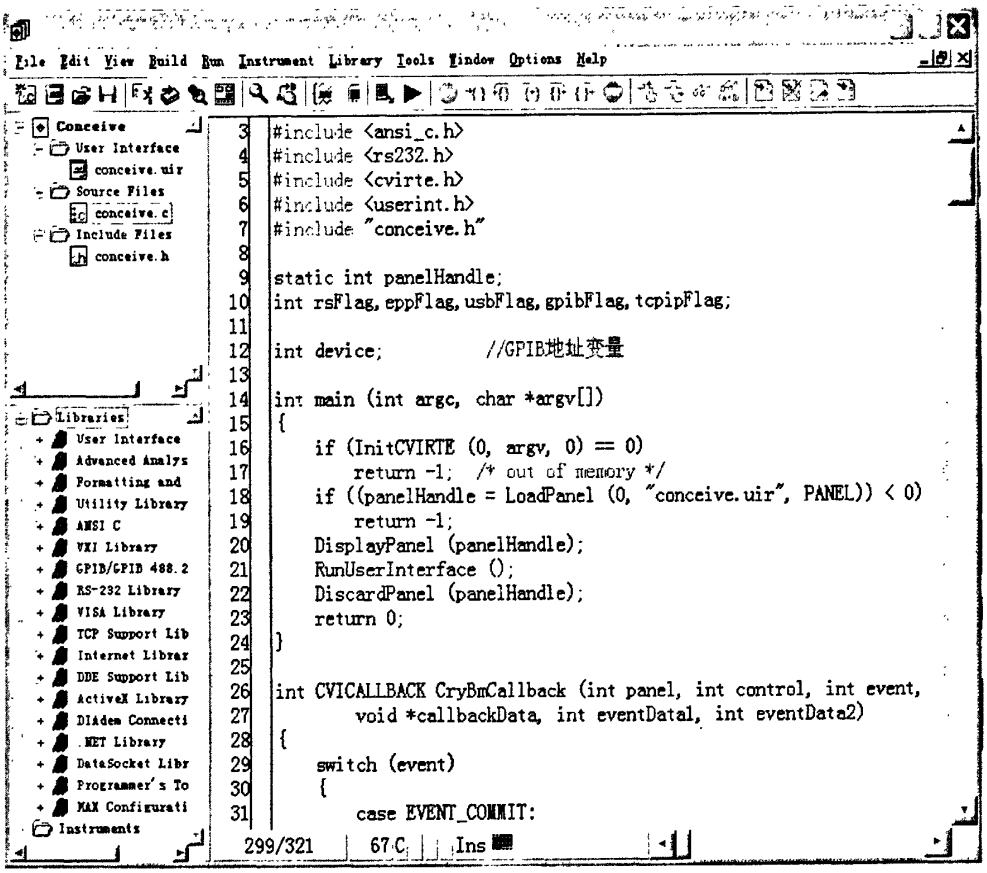


图 4.8 源代码编辑窗口

Fig.4.8 The source code editor window

Main()和终止程序时所选的回调函数中的代码是计算机自动生成的，main()函数是程序的入口，它的功能是初始化程序，装载用户界面面板，并显示。计算机只生成其它回调函数的框架，其内部的用来响应并处理消息的代码，需要设计者根据功能控制自己添加，这也正是 lab Windows/CVI 的应用灵活之处。

(2) 程序框架与文件间的关系

程序框架与文件间的关系如图 4.9 所示

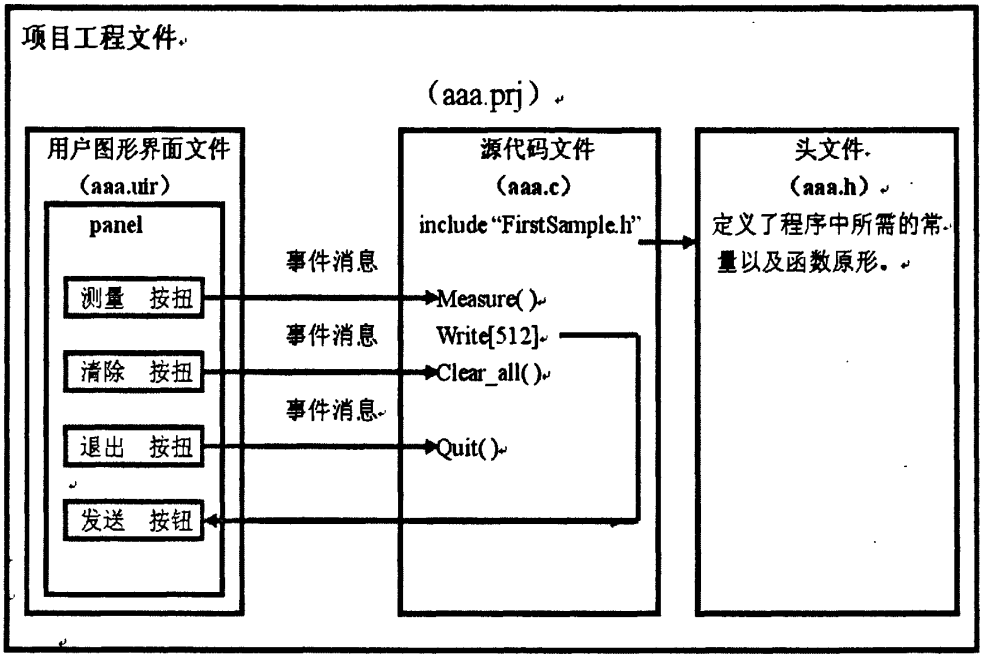


图 4.9 程序框架与文件间的关系

Fig.4.9 The relationship between the application Framework and the document

Labwindows/CVI 的编程是基于事件[46]编写的。所谓的用户事件也就是 Windows 操作系统的消息，Labwindows/CVI 通过回调函数的方式处理用户事件，这种基于事件的编程方式简化了对应用程序流程的控制。在 Labwindows/CVI 编程中用到最多的事件就是 EVENT_COMMIT 事件（鼠标释放事件），当用户点击一个命令按钮时，被点击的按钮产生一个 EVENT_COMMIT 事件，通过 Labwindows/CVI 传递给用户 C 程序，然后由程序执行被点击控件定义的事情。

Labwindows/CVI 的强大功能很大程度体现在它提供丰富的函数和简单快捷的函数面板工具。点击源代码编辑窗工具栏的 Library，就可以打开函数库的下拉菜单。LabWindows/CVI 所提供的库函数从用户图形界面，数据采集，数据分析，仪器控制 … 到现在 Internet 时代的 TCP。所以说 LabWindows/CVI 在测量领域成为先锋的同时又与当前时代的新科技保持了同步。

(3) 通过 GPIB 接口发送 SCPI 消息

LabWindows/CVI 中的 GPIB/GPIB-488.2 函数库可以实现打开/关闭 GPIB 设备、总线配置、I/O 读写、GPIB 设备控制、总线控制和板控制功能[47]。在进行 GPIB 其它操作之前，一定要先打开 GPIB 设备，初始化设置，然后通过 GPIB 接口向外部发送命令。

在这一节中主要介绍发送命令的编程方法[1]，利用“Start”按钮的回调函数

1

Measure()实现。通过上面的章节我们知道，Labwindows/CVI 的编程是基于事件编写的，最常用的事件就是 EVENT_COMMIT（鼠标释放），当点击一个命令按钮时，就发生了 EVENT_COMMIT 事件。当点击“Start”按钮时，Labwindows/CVI 将调用 Measure()函数，Measure()函数将测试语句以 SCPI 格式依次自动下发。下面给出程序控制晶继与继电器自动测量的子程序。

```
int CVICALLBACK Measure (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
  static char write_buffer[100];
  int len;
  int i;
  char *Command[]={"*RST",":INST:SEL1",":PROG:SEL:NUMB 1",":PROG:SEL:DEF 1",
                   ":TRIG:SEQ:TTL",":CAL:STAT"}; //整个的测量指令发给仪器.
  switch (event)
  {
    case EVENT_COMMIT:
        for(i=0;i<5;i++)
        {
          ibwrt (device, Command[i], strlen(Command[i]));     //向 GPIB 设备发送数据
          ibwrt (device, "\n", 1);                            //发送结束符
          Delay(1);
        }
        if(ibsta & 0x8000)                                    //判断数据发送是否出错
        {
          SetCtrlVal (panelHandle, PANEL_ERRSTATUS, 1); //设置出错状态灯亮
          SetCtrlVal (panelHandle, PANEL_ERR, iberr);         //在控件 ERR 中显示出错代码
        }
       else
        {
         SetCtrlVal (panelHandle, PANEL_ERRSTATUS, 0);    //设置出错状态灯灭
         SetCtrlVal (panelHandle, PANEL_ERR, 0);
        }
    break;
  }
  return 0;
}
```

ibwrt()是 GPIB 函数库时里的写字符串命令函数，函数原型：int ibwrt(int ud, const void *data, long num_bytes);输入参数有：ud，设备描述符；data，被写入的常量字符串数组指针，num_bytes：从用户队列写入的字符个数。返回值：ibsta，通过判断全局变

量 ibsta 在发送过程中是否出错，如果出错将错误代码显示到 ERR 控件中。

SetCtrlVal：设置控件的值。函数原型：int SetCtrlVal(int panelHandle, int controlID,void value)。输入参数：value，控件的新值（value 的数据类型必须与控件的数据类型相匹配）。

程序说明：为了实现系统的通用型，上位机的命令是被组合称为 SCPI 命令格式发布的，经由 GPIB 接口转换卡接收命令并根据不同仪器进行相应的命令解析并发布给相应仪器，实现远地控制。

5 系统测试及结果

5.1 调试过程

5.1.1 硬件调试

硬件电路的焊接可以先焊接电源模块，电源模块焊接完毕后给系统上电，电源指示灯亮，用万用表测试电源是否工作正常，经测试电路完全正常。然后再焊接单片机模块的电路，焊接完毕后可以用万用表测试一下是否有虚焊的现象，经测试焊接良好。下面可以给单片机烧进一个测试程序，以判断 CPU 工作是否正常，经测试系统正常。

然后再焊接 FPGA 模块，焊接完毕后，对 208 个引脚依次用万用表检测，确保没有虚焊现象；给芯片上电，测量所有的电源引脚，要保证每一种电压每一个电源引脚都正确加载；检查 JTAG 引脚，上拉、下拉电阻是否都正确连接；时钟是否起振，频率是否吻合；编写程序，测试 FPGA 工作是否正常。至此，硬件电路完全工作正常，可以进行软件的调试了。

5.1.2 软件调试

软件调试分上位机软件调试和下位机软件调试两个部分。

(1) 下位机软件调试

下位机软件调试包括单片机和 FPGA 的程序调试，单片机的程序用 C 语言编写，完成LCD、键盘的驱动，对FPGA的总线操作以及内部的数据处理，使用在线调试工具LM LINK调试器，该调试工具在开发环境 IAR 下可以在线单步调试程序、设置断点调试以及下载程序，从而为调试程序，查找错误提供了一个非常简便的方法。

FPGA 部分采用混合描述方式建立电路模型，用结构描述方式将现有电路单无或模块作为“元件”来调用和连接，使原理框图清晰分层；用行为描述方式实现电路单元的功能描述，使代码紧凑。主要在 QuartusII 建立波形文件进行时序仿真进行功能验证。

(2)上位机软件调试

本测试仪利用 GPIB 总线进行数据传输，所以必须使 PC 机和仪器能够正常通讯的基础上才能对仪器进行控制和数据采集。上位机软件的调试第一步是要将 GPIB 通信调通，调试 GPIB 通信在软件的 GPIB 通信检测区进行，调试方法是，我们通过软件向仪器发送 GPIB 标准命令，从返回的字符串来判断 PC 机是否和仪器通信正常。

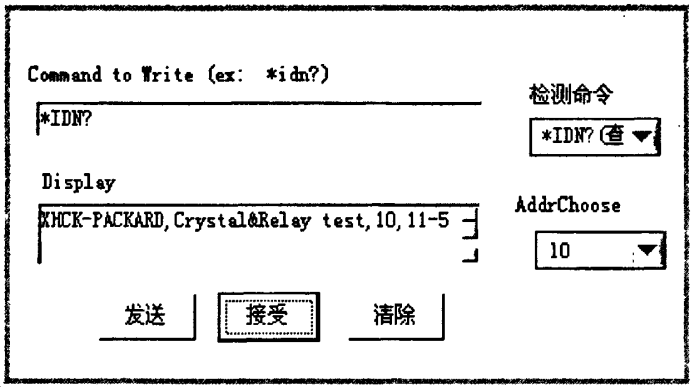


图 5.1 GPIB 通信检测

Fig.5.1 GPIB Communication Detection

如图 5.1 所示，先设置检测命令为：*IDN?，在设计 AddrChoose（仪器 GPIB 地址）为 10，然后点击发送按钮，再点击接受按钮，如果 PC 机和数字多用表通信正常的话，就会在 Display 的文本框中显示从仪器回来的一串字符串，这段字符串的含义是描述被控仪器的状态。若通信不正常的话就不会得到这段字符串，这个时候就要检查仪器的 GPIB 是否设置正确，GPIB 线缆有没有插紧和有没有问题，若还是不能正常通信，就要检查程序中调用 GPIB-PCI 卡的驱动函数有没有问题等等。本系统的 GPIB 通信经过调试是全部都通的，这样才可以保证数据正常采集。GPIB 通信调试通过后，就可以对仪器进行调试了。以下是依次发送的 SCPI 语名，注释里的数字是用 16 进制表示的由 SCPI 解析模块译成的二进制代码。

*RST	//复位	*05	
:INST:SEL 1	//功能选择，(1 选择晶振)	:0E05	1
:PROG:SEL:NUMB 1	//时标设置（1S）	:130205	1
:PROG:SEL:DEF 1	//标称值 (1M)	:130200	1
:TRIG:SEQ:TTL	//送启动信号	:1C0012	
:CAL:STAT	//校准，	:0206	
:FETC			

5.2 系统测试

根据国标要求，对数显时间继电器校准点的选择，s 档为 1s、9s、99s、999s。min 档为 1min、9min。为提高测量效率，8 路继电器同时进行测试时，统一设置为相同的档位下相同的值。

具体操作如下：

(1)接通电源。

(2)开始测量前，先进行必要的配置。详细步骤如下：

1	2	3	4
5	6	7	8
9	0	Quit	Set
Realy	Crystal	Select	Start

图 5.2 本地键盘布局图

Fig.5.2 The layout of Local keyboard

①选择本地远地

在本地键盘上按键 2，将在 LCD 上看到“本地”提示，表示下面的操作将进入利用本地键盘进行控制的操作。按键 3，将在 LCD 上看到“远地”提示，表示进入远程控制模式。

②在本地控制下的测试流程如下：

- 选择测试菜单

  根据所要测的类型，按下相应的测试按钮，如果测继电器，按 Realy 键；测晶振，按 Crystal 键，在 LCD 上会弹出相应的测试菜单。
- 输入继电器的设置值或晶振的标称值
- 按 Set 键，如果第 1 步选的是 Relay 键，此时将会看到 LCD 上“设置值”菜单下出现一个闪烁的光标，按数字键 0～9，输入与被测继电器相同的设置值；如果第 1 步选的是 Crystal 键，此时将会看到 LCD 上“标称值”菜单下出现一个闪烁的光标，按数字键 0~9，输入与被测晶振相同的标称值。每输入一位数，光标后移一位，继续闪烁，设置完成后，按 Quit 键结束操作，此时光标消失。
- 按 Select 键，将会在 LCD 的“继电器时标”菜单下看到 1us 这个时标单位，按一下扩大 10 倍，时标单位 1us——10s 依次出现。按 8 次之后，出现一个提示语句：“Next”将进入下一轮的分频设置。最终的分频值以最后的选择为准。
- 完成上面两步的设置后，按下 Start 键，测试工作正式开始。
- 一次测量结束后，进行下一次测量之前，如果需要改变时标，依次按照步骤②中的项目编号三、四操作，如果不需要改变时标，直接操作步骤②中的第四步，可进行反复测量。

(3)在远程控制下的测试流程：

本仪器可通过多种接口与上位机通信，在此仅以 GPIB 接口通信为例进行介绍。图 4.7 是 lab Winsdows/CVI 控制的上位机界面，还可以在 PC 机上运行安捷伦连接专家软件，依次发送图 5.3 中 Instrument Session History 中的 SCPI 指令即可实现对仪器的标

准化控制。

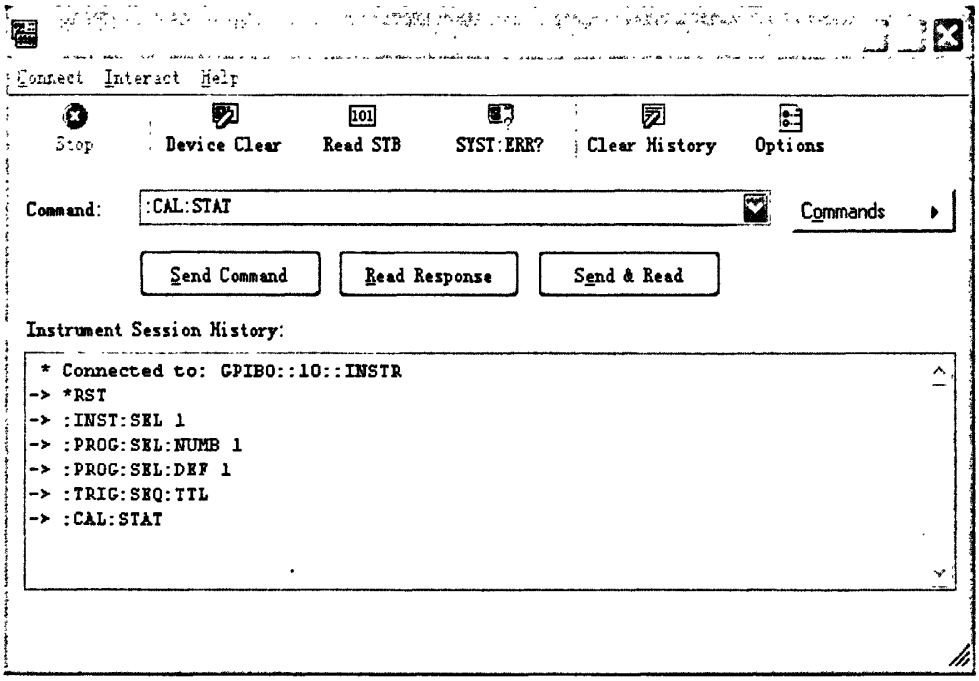


图 5.3 GPIB 接口操作界面

Fig.5.3 The Operation interface of GPIB

5.3 作品实物及测量结果显示

晶振与时间继电器测测试仪实物图如图 5.4 所示，从图中我们可以看到当设置时间继电器的门控时间是 7s，基准频率选取 100us 时，该系统测得的第二通道继电器的实际值为 7.35555s， 第四通道继电器的实际值为 7.3534。并别对应给出实测值与设置值之间的绝对误差和相对误差。

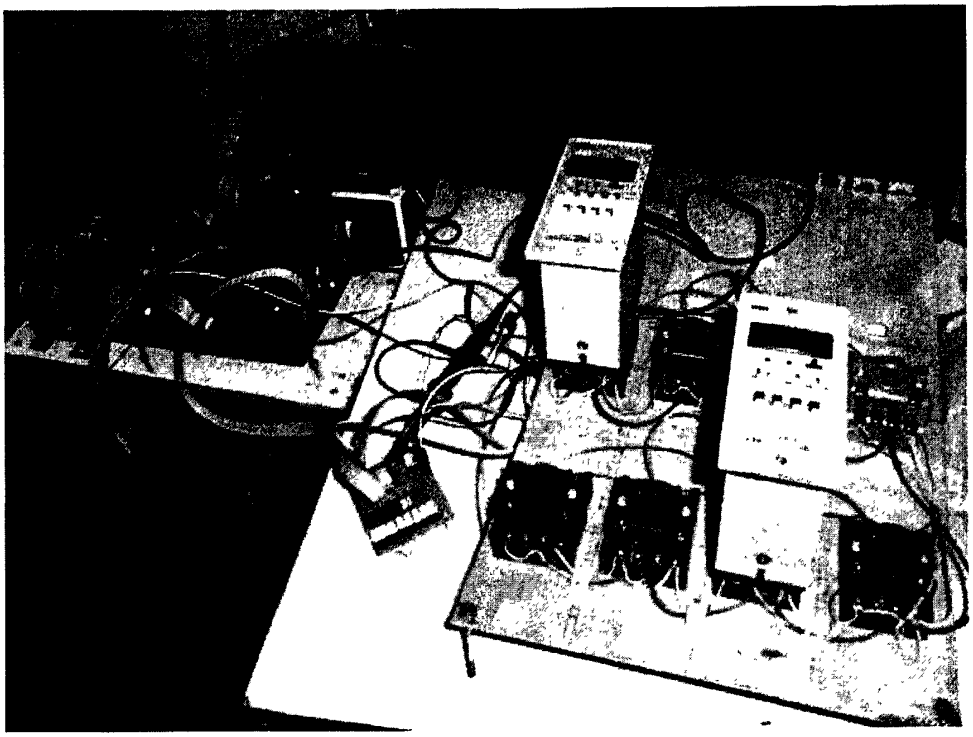


图 5.4　晶振与时间继电器测测试仪实物图

Fig.5.4　Physical map of crystal and time relay tester

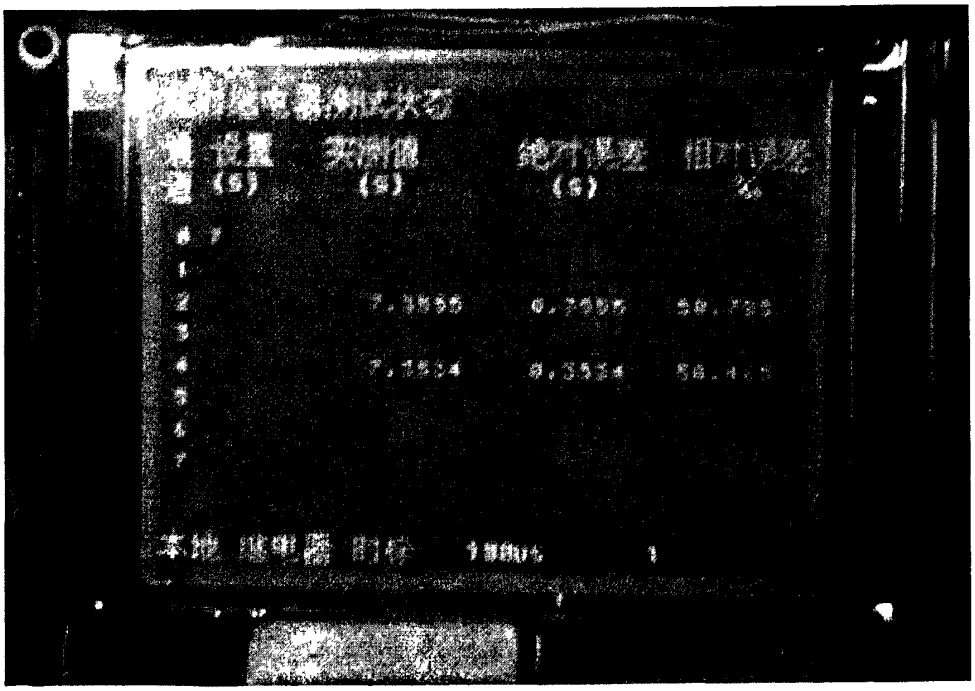


图 5.5　结果显示

Fig.5.5　The display of results

附注：二个通道上有被测继电器时的结果显示，若插座上未扫描到继电器，不对该通道进行处理。

结　论

经过实验仿真分析及验证，晶体与时间继电器自动测试系统的各项功能达到了预定的设计指标。在开发过程中，综合运用了 Quartus II、Protel 99SE、IAR 等开发工具，使用了 VHDL 及 C 语言。采用自上而下的设计方法，提高了设计效率，缩短了开发周期，降低了设计费用。

该测试仪的特点是将功能强大的 ARM 控制芯征和 FPGA 芯片的现场可编程性相结合，不但大大缩短了开发研制周期，而且使本系统具有结构紧凑、体积小、重量轻、可靠性高、测量时间/频率范围宽、精度高等优点。同时利用 ARM 具有良好的人机接口和控制运算的功能，可以较简单地实现键盘和显示控制以及数据处理运算。在显示方面，利用串转并芯片发送扫描数据，节约了 I/O 口，简化了驱动电路的设计。设计并制作了系统工作的外围电路系统如直流工作电源。系统联合调试成功后，可将单片机程序通过调试器 LM LINK 固化到单片机中，将编译后生成的 SOF 文件，下载到 CycloneII 的配置芯片 EPCS4 中，每次系统上电，由这个专用配置芯片把配置数据加载到 FPGA 中，之后，FPGA 就可以正常工作了。将整个系统的外围电路设计制作成印刷电路板。并成功的进行了调试，达到了设计要求。

本测试系统虽然实现了对晶振与继电器的自动测量、数据存储和显示等一些基本功能，由于时间比较紧以及个人能力和实验条件等方面的限制，有以下三点后续工作还有待完善：

(1)LM3S8962 支持以太网接口和 SD 卡接口，在硬件设计中也已经把这两个模块做在 PCB 板上，预计将测量数据存储在 SD 卡，通过以太网传输，实现远程网络监控。针对这两个模块的软件程序尚未进行开发。

(2)本测试仪只完成了对晶振频率准确度与继电器延时参数的测试，其它参数测试，如晶振的开机特性、日频率波动、1S 频率稳定度等，继电器的整定误差、电压波动误差、温度误差等，有待于进一步开发。

(3)未将测量结果生成报表。Labwindows/CVI 本身自带了对 WORD 和 EXCELL 操作的函数，当用户按照顺序测试完毕后，可以把测得的数据按照规定的报表形式保存在报表中，还可以对错误的数据进行统计，将统计的结果存入数据库中以便将来查询。

通过本课题的设计，掌握了 FPGA 技术和频率测量技术，掌握单片机方面的应用，受益匪浅，为我今后的工作和学习奠定了坚实的基础。相信随着微电子技术的发展，高性能芯片价格的下降和现代仪器技术、计算机技术、网络技术的进步，在此领域的研究会得到更大的发展！

参 考 文 献

[1] 徐季平,拂为,徐旻.时间频率测量仪器原理与使用[M].北京:中国计量出版社,2009.

[2] 马凤鸣,国家质量监督检验检疫总局计量司 组编.时间频率计量[M].北京:中国计量出版社,2009.

[3] 姜维正.频率的精密测量方法应用[J].计算机技术与应用，2006(4):63-65.

[4] 孙杰,潘继飞.高精度时间间隔测量方法综述[J].计算机测量与控,2007,15(2):145-148.

[5] 陈晓荣,蔡苹,周红全.基于单片机的频率测量的几种实用方法[J].工业仪表与自动化装置，2003(1):40-42.

[6] 谈学.基于单片机的等精度频率计设计[J].重庆工商大学学报，2004,21(2):133-135.

[7] 刘夫江.基于单片机和CPLD的等精度数字频率计设计[D].济南：山东大学，2007.

[8] 张兆莉，蔡永泉，王钰.基于 FPGA 的数字频率的设计与实现[J].自动化仪表，2006,27(11):10-13.

[9] 金 琳.基于EDA技术的频率计系统设计[D].吉林：吉林大学,2007.

[10] 王杰，张涛．一种铁路安全继电器高精度测试系统的实现[J]．郑州大学学报，2006,27(4):63-65.

[11] 张建平.时间继电器延迟时间的高准确度测量[J]．宇航计测技术，2003,23(1):12-16.

[12] Joseph Yiu. The Definitive Guide to the ARM Cortex-M3[M]．北京航空航天大学出版社，2009.

[13] 周立功．Cortex-M3开发指南—— 基于LM3S8000.广州致远电子有限公司.2008.

[14] 刘同法,肖志刚,彭继卫.ARM Cortex-M3 内核微控制器快速入门与应用[M]．北京:北京航空航天大学出版社，2009.

[15] 周立功.EasyARM8962实验教程.广州致远电子有限公司.2008.

[16] 熊茂华,杨震伦．ARM9嵌入式系统设计与开发应用[M]．北京：清华大学出版社,2008.

[17] 廖义奎．ARM与FPGA综合设计及应用[M]．陕西：中国电力出版社，2008.

[18] Wayne W.基于FPGA的系统设计[M].北京：机械工业出版,2006.

[19] 王诚,吴继华,范丽珍，等.Altera FPGA/CPLD设计(高级篇)[M]．北京：人民邮电出版社，2005.

[20] 刘延飞.基于Altera FPGA/CPLD的电子系统设计及工程实践[M].北京:人民邮电出版社.2009.

[21] 黄智伟.FPGA系统设计与实践[M].北京：电子工业出版社,2005.

[22] Steve K. Advanced FPGA Design[M]．北京：机械工业出版社,2009.

[23] 欧伟明.基于MCU、FPGA、RTOS的电子系统设计方法与实例[M].北京航空航天大学出版社,2007.

[24] 赵艳华.基于Quartus II的FPGA/CPLD设计与应用[M]．电子工业出版社,2009.

[25] 华清远见嵌入式培训中心．FPGA应用开发入门与典型实例[M]．北京：人民邮电出版社,2008.

[26] 刘思久,张礼勇.自动测试系统与虚拟仪器原理•开发•应用[M]．北京：电子工业出版社,2009.

[27] 王先培.测控总线与仪器通信技术[M].北京:机械工业出版社,2007.

[28] 黄玉金.GPIB的研究与设计[D].北京:北京化工大学,2000.

[29] 张毅刚.计算机自动测试[M]．哈尔滨:哈尔滨工业大学出版社,2006

[30] 吴国庆.现代测控技术及应用[M].北京:电子工业出版社,2007.

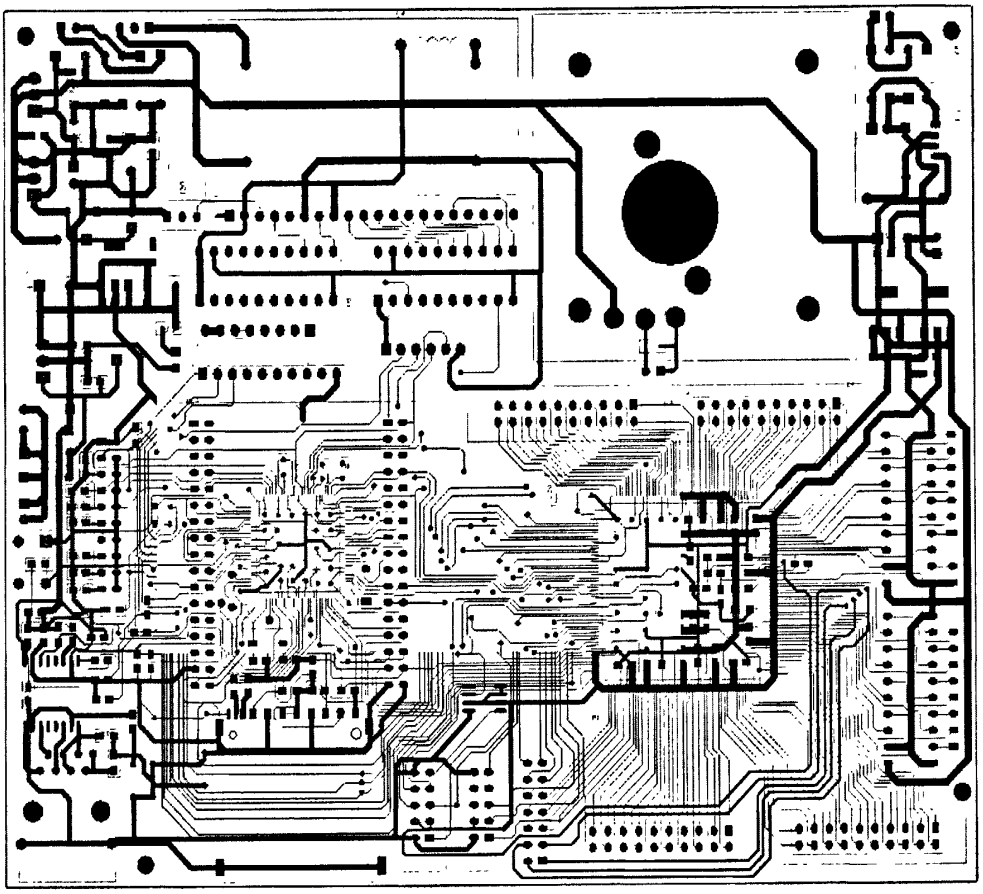
[31] 秦红磊,路辉,郎荣玲.自动测试系统：硬件及软件技术[M].北京:高等教育出版社,2007.

[32] 陈长龄.自动测试及接口技术[M].北京:机械工业出版社,2005.

[33] 李维提.液晶显示器应用技术[M].北京:北京邮电学院出版社,1997.

[34] 徐爱钧. IAR EWARM 嵌入式系统编程与实践[M]. 北京:北京航空航天大学出版社, 2006.
[35] 周霭如, 林伟键. C++程序设计基础[M]. 北京：电子工业版社，2006.
[36] 郑亚民. 可编程逻辑器件开发软件 QuartusII[M]. 北京:国防工业出版社，2006.
[37] 周润景，图雅，张丽敏. 基于 Quartus II 的 FOGA/CPLD 数字系统设计实例[M]. 北京：电子工业出版社，2007.
[38] 罗力凡. 基于 VHDL 的 FPGA 开发快速入门•技巧•实例[M]. 北京:人民邮电出版社. 2009.
[39] James R Armstrong, F Gall Gray. VHDL Design Representation and Synthesis Second Edition [M]. 北京：机械工业出版社，2003.
[40] 张永艳. 基于复杂可编程逻辑器件及用 VHDL 语言编程的数字频率计的设计[D]. 呼和浩特：内蒙古大学，2004.
[41] 徐洋. 基于 Verilog HDL 的 FPGA 设计与工程[M]. 北京:人民邮电出版社, 2009:
[42] 林灶生. Verilog FPGA 芯片设计[M]. 北京：北京航天航空大学出版社, 2006.
[43] 乔庐峰. Verilog HDL 数字系统设计与验证[M]. 北京：电子工业版社，2009.
[44] 孙晓云，郭立炜，孙会琴. 基于 Lab Windows/CVI 的虚拟仪器设计与应用[M]. 北京：电子工业版社，2005.
[45] 宋宇峰. Lab Windows/CVI 逐步深入与开发实例[M]. 北京：机械工业出版社，2003.
[46] 张毅刚，乔立岩. 虚拟仪器开发环境：Lab windows/CVI6.0 编程指南[M]. 北京：机械工业出版社，2002.
[47] 雷勇. 虚拟仪器设计与事件[M]. 北京：电子工业出版社，2005.
[48] 余成波, 冯丽辉, 潘盛辉. 虚拟仪器技术与设计[M]. 重庆：重庆大学出版社，2006.

附录 A　晶振与时间继电器测试仪 PCB 板图



附录 B　单片机 main 函数

```
int main(void)
{
  UCHAR   KeyVal;
  UCHAR SetCount=0;
  UCHAR keysetval,kyout,KeyCtrl;

  GPIO_Init();                          //对用的GPIO模块、LCD、键盘进行初始化
  LcmInition();
  keyInit();

  WriteByte(0x8009,0);                  //RAM释放
```

```
ReadByte(0x8000);                    //读一次使I/INT0中断为高，最初运行时
                                     //双方不占用RAM，中断脚均为高
while(1)
{
  KeyCtrl=keyRead();                 //键盘扫描子程序
  if(KeyCtrl==2)                     /如果扫描到键值为2，进入本地控制
  {                                  //的循环
    Putstr(0,222,"本地",1,16);
    PublicMenu();                    //在LCD上显示公用菜单
    do                               //进入设置与测量阶段
    {
      KeyVal=keyRead();
      if(KeyVal==13)                 //若扫描到键值为13
      {
        RelayMenu();                 //在LCD是显示继电器测量菜单
        RelayFlag=1;                 //用变量记录这个选择
        CrystalFlag=0;
        ClearAll();                  //清除LCD上显示的前一次测量值
        ClearSet();                  //清除设置值
        Putstr(18,222,"          ",1,16);
      }
      else if(KeyVal==14)            //若扫描到键值为14
      {
        CrystalMenu();               //在LCD是显示晶振测量菜单
        CrystalFlag=1;               //用变量记录这个选择
        RelayFlag=0;
        ClearAll();
        ClearSet();
        Putstr(18,222,"          ",1,16);
      }
      else if(KeyVal==15)            //设置分频，对分频后值用变量m记录下来
      {
        Delay(1000000);
        if(KeyVal==15)
```

```
        {
          c++;
          if(c==7)c=0       //设置8个循环可调的分段段
        }
        if(RelayFlag==1)    //如果测量对象是继电器，设置时标周期并
        {          //用TimeBase记录下来，乘以后续的计数结果即为时间间隔
          switch(c)         // m是单片机的控制字，第八位Control;
          {                 //第7位cont；第4 晶/继选择；后三位代表分频值。
            case1:Putstr(18,222,"1us",1,16);m=0x80;TimeBase=1;break;
            case2:Putstr(18,222,"10us",1,16);m=0x81;TimeBase=2;break;
            case 3:Putstr(18,222," 100us",1,16);m=0x82;TimeBase=3; break;
            case 4:Putstr(18,222," 1ms",1,16);m=0x83;TimeBase=4; break;
            case 5:Putstr(18,222," 10ms",1,16);m=0x84;TimeBase=5; break;
            case6:Putstr(18,222,"100ms",1,16);m=0x85;TimeBase=6;break;
            case 0:Putstr(18,222," Next",1,16); break;
          }
        }
        if(CrystalFlag==1)        //测量对象是继电器的设置
        {
          switch(c)
          {
            case 1:Putstr(18,222,"1s",1,16); m=0x06;TimeBase=1;break;
            case 2:Putstr(18,222,"10s",1,16);m=0x07;TimeBase=2;break;
            case 3:Putstr(18,222,"100s",1,16);m=0x08;TimeBase=3; break;
            case 4:Putstr(18,222,"1000s",1,16);m=0x09;TimeBase=4; break;
            case 5:Putstr(18,222,"10000s",1,16);m=0x0A;TimeBase=5; break;
            case 6:Putstr(18,222,"100000s",1,16);m=0x0B; TimeBase=6;break;
            case 0:Putstr(18,222," Next",1,16); break;
          }
        }
}
else if (KeyVal==16)        //扫描到键值为16，送开如测量的启动信号
{                           //并同时送进去分频值、测试对象标志信号
  Delay(500000);            // 500ms的等待时间
```

```
if (KeyVal==16)          //键盘去抖动处理
{
  ClearAll();
  WriteByte(0x800D,m&0xBF);          //送启动信号
  WriteByte(0x800D,m|0x40);          //让m的第4位分别为高、低、高
  WriteByte(0x800D,m&0xBF);          //产生高电平脉冲充当启动信号
  if(RelayFlag==1)
  {
    Delay(2000000);
    SelectAddr(RelayClr);                    //继电器的清0信号
    do
    {              //对8个通道轮循其测量结束标志位，一旦检测到为高，
      kyout=keyRead();  //用Inquire()完成此通道的读取、计算和显示
      Delay(500000);
      Inquire(Ebtri0,HeadMeas0,HeadAbso0,HeadRela0,Rslt0,DATARD00);
      Inquire(Ebtri1,HeadMeas1,HeadAbso1,HeadRela1,Rslt1,DATARD10);
      Inquire(Ebtri2,HeadMeas2,HeadAbso2,HeadRela2,Rslt2,DATARD20);
      Inquire(Ebtri3,HeadMeas3,HeadAbso3,HeadRela3,Rslt3,DATARD30);
      Inquire(Ebtri4,HeadMeas4,HeadAbso4,HeadRela4,Rslt4,DATARD40);
      Inquire(Ebtri5,HeadMeas5,HeadAbso5,HeadRela5,Rslt5,DATARD50);
      Inquire(Ebtri6,HeadMeas6,HeadAbso6,HeadRela6,Rslt6,DATARD60);
      Inquire(Ebtri7,HeadMeas7,HeadAbso7,HeadRela7,Rslt7,DATARD70);
    }while(kyout!=1);          //设置一个跳出键，免设置进入下一次测试
  }
 if( CrystalFlag==1)          //测量对象为继电器时，进行类似处理
 {
  do
  {
   kyout=keyRead();
   Delay(500000);
   CryInquire(Ebtri0,HeadMeas0,HeadAbso0,HeadRela0,Rslt0,DATARD00);
   CryInquire(Ebtri1,HeadMeas1,HeadAbso1,HeadRela1,Rslt1,DATARD10);
   CryInquire(Ebtri2,HeadMeas2,HeadAbso2,HeadRela2,Rslt2,DATARD20);
   CryInquire(Ebtri3,HeadMeas3,HeadAbso3,HeadRela3,Rslt3,DATARD30);
```

```
            CryInquire(Ebtri4,HeadMeas4,HeadAbso4,HeadRela4,Rslt4,DATARD40);
            CryInquire(Ebtri5,HeadMeas5,HeadAbso5,HeadRela5,Rslt5,DATARD50);
            CryInquire(Ebtri6,HeadMeas6,HeadAbso6,HeadRela6,Rslt6,DATARD60);
          }while(kyout!=1);
        }
      }
    }
    else if (KeyVal == 12)                    //完成被测对象标称值的设置、显示及存贮
    {
      ClearSet();                             //开始设置前清上一次的设置值
      WriteCommand( DispOn );                 // 写入指令代码
      WriteData( 0x57 );
      WriteCommand( CsrDirR );                // 自动右移光标指向
      WriteCommand( CsrW );                   // 制定光标位置
      WriteData(0x6B); //设置光标地址CSR,第7列第10个字,最后一个设置值的地址   WriteData(0x01);
      do
      {
        keysetval=keyRead();                  //读取按键设置值
        Delay(500000);
        if(SetCount==0)                       //如果共输入一次，则真实值只有一位
        {
          switch(keysetval)   //根据具体数字将其对应ASCII码写入LCD显示存储器
          {                   // 0x16B位置，0x16b=40*9+3，即第40行第三个字符处
            case1:LCDMRWite(0x16B,0x31);SetCount++;break;   //setCount记录第几
            case 2:LCDMRWite(0x16B,0x32);SetCount++; break;//次输入设置值
            case 3:LCDMRWite(0x16B,0x33);SetCount++; break;
            case 4:LCDMRWite(0x16B,0x34);SetCount++; break;
            case 5:LCDMRWite(0x16B,0x35);SetCount++; break;
            case 6:LCDMRWite(0x16B,0x36);SetCount++; break;
            case 7:LCDMRWite(0x16B,0x37);SetCount++; break;
            case 8:LCDMRWite(0x16B,0x38);SetCount++; break;
            case 9:LCDMRWite(0x16B,0x39);SetCount++; break;
            case 10:LCDMRWite(0x16B,0x30);SetCount++;break;
            default:break;
```

```
        }
    }
    else if(1==SetCount)   //如果共输入2次，则个位数字随之显示在0x16C处
    {
      switch(keysetval)
      {
        case 1:LCDMRWite(0x16C,0x31);SetCount++; break;
        case 2:LCDMRWite(0x16C,0x32);SetCount++; break;
        case 3:LCDMRWite(0x16C,0x33);SetCount++; break;
        case 4:LCDMRWite(0x16C,0x34);SetCount++; break;
        case 5:LCDMRWite(0x16C,0x35);SetCount++; break;
        case 6:LCDMRWite(0x16C,0x36);SetCount++; break;
        case 7:LCDMRWite(0x16C,0x37);SetCount++; break;
        case 8:LCDMRWite(0x16C,0x38);SetCount++; break;
        case 9:LCDMRWite(0x16C,0x39);SetCount++; break;
        case 10:LCDMRWite(0x16C,0x30);SetCount++;break;
        default:break;
      }
    }
    else if(2==SetCount) //如果共输入3次，则个位数字随之显示在0x16D处
    {                    //前面显示在0x16B、0x16C处的做为百位和十位。
      switch(keysetval)
      {
        case 1:LCDMRWite(0x16D,0x31);SetCount++; break;
        case 2:LCDMRWite(0x16D,0x32);SetCount++; break;
        case 3:LCDMRWite(0x16D,0x33);SetCount++; break;
        case 4:LCDMRWite(0x16D,0x34);SetCount++; break;
        case 5:LCDMRWite(0x16D,0x35);SetCount++; break;
        case 6:LCDMRWite(0x16D,0x36);SetCount++; break;
        case 7:LCDMRWite(0x16D,0x37);SetCount++; break;
        case 8:LCDMRWite(0x16D,0x38);SetCount++; break;
        case 9:LCDMRWite(0x16D,0x39);SetCount++; break;
        case 10:LCDMRWite(0x16D,0x30);SetCount++;break;
        default:break;
```

```
    }
  }
}while(keysetval!=11);
switch(SetCount)                    //数组高位存高位
{                   //根据设置位的位数，将其转换为十进制大小，存储用于计算
  case 1: StoreSet[0]=LCDMRead(0x16B);SetVal=StoreSet[0]-0x30;break;
  case 2: StoreSet[0]=LCDMRead(0x16C);
          StoreSet[1]=LCDMRead(0x16B);
          SetVal=(StoreSet[1]-0x30)*10+(StoreSet[0]-0x30);
           break;          了          //StoreSet[]是全局数数组,高位存高位数
  case 3: StoreSet[0]=LCDMRead(0x16D);  //LCDMRead()返回ASCII码
            StoreSet[1]=LCDMRead(0x16C);
            StoreSet[2]=LCDMRead(0x16B);
            SetVal=(StoreSet[2]-0x30)*100+(StoreSet[1]-0x30)*10+
                  (StoreSet[0]-0x30); break;
     default:break;
    }
    SetCount=0;                     //归0 为重新设置作准备
    WriteCommand( DispOn );          // 写入指令代码
    WriteData( 0x54 );              //关光标
  }                                 //设置值的结束
 }while(1) ;                        //本控里循环键值的读取
}                                   //本控的结束
else if(KeyCtrl==3)                 //Key=3选择远程控制
{                   /使能外部中断,如果3键不按下,不会响应上位机的中断
   GPIOIntTypeSet(GPIO_PORTD_BASE,I_Int0,GPIO_LOW_LEVEL);//设置I_Int0中断触发方式为高
   GPIOPinIntEnable(GPIO_PORTD_BASE,I_Int0);
   IntEnable(INT_GPIOD);            //  使能KEY1中断
   Putstr(0,222,"远地",1,16);
   do                               //进入远程控制主体的循环
   {
     if(MeasFlag==1)
     {
       if(RelayFlag==1)
```

```
    {
      Delay(2000000);
      SelectAddr(RelayClr);                    //继电器的清0信号
      Delay(1000);
      do
      {
        Delay(500000);
        Inquire(Ebtri0,HeadMeas0,HeadAbso0,HeadRela0,Rslt0,DATARD00);
        Inquire(Ebtri1,HeadMeas1,HeadAbso1,HeadRela1,Rslt1,DATARD10);
        Inquire(Ebtri2,HeadMeas2,HeadAbso2,HeadRela2,Rslt2,DATARD20);
        Inquire(Ebtri3,HeadMeas3,HeadAbso3,HeadRela3,Rslt3,DATARD30);
        Inquire(Ebtri4,HeadMeas4,HeadAbso4,HeadRela4,Rslt4,DATARD40);
        Inquire(Ebtri5,HeadMeas5,HeadAbso5,HeadRela5,Rslt5,DATARD50);
        Inquire(Ebtri6,HeadMeas6,HeadAbso6,HeadRela6,Rslt6,DATARD60);
        Inquire(Ebtri7,HeadMeas7,HeadAbso7,HeadRela7,Rslt7,DATARD70);
      }while(NextMeas==0);        //设置一个跳出键
            NextMeas=0;                    //此处清0 为下一次使用作准备
  }
  if (CrystalFlag == 1)
  {
   do
   {
     Delay(500000);
     CryInquire(Ebtri0,HeadMeas0,HeadAbso0,HeadRela0,Rslt0,DATARD00);
     CryInquire(Ebtri1,HeadMeas1,HeadAbso1,HeadRela1,Rslt1,DATARD10);
     CryInquire(Ebtri2,HeadMeas2,HeadAbso2,HeadRela2,Rslt2,DATARD20);
     CryInquire(Ebtri3,HeadMeas3,HeadAbso3,HeadRela3,Rslt3,DATARD30);
     CryInquire(Ebtri4,HeadMeas4,HeadAbso4,HeadRela4,Rslt4,DATARD40);
     CryInquire(Ebtri5,HeadMeas5,HeadAbso5,HeadRela5,Rslt5,DATARD50);
     CryInquire(Ebtri6,HeadMeas6,HeadAbso6,HeadRela6,Rslt6,DATARD60);
   }while(NextMeas!=1);
   NextMeas=0;
  }                          // if( CrystalFlag==1)结束
}                            //if(MeasFlag==1) 结束
```

```
    } while(1);                    //远控里循环键值的读取
  }                                //远控的结束
 }                                 //本地远程控制的while(1)
}                                  //主程序列结束
```

参考文献

[1]徐季平,拂为,徐旻.时间频率测量仪器原理与使用[M].北京:中国计量出版社,2009.
[2]马凤鸣,国家质量监督检验检疫总局计量司 组编.时间频率计量[M].北京:中国计量出版社,2009.
[3]姜维正.频率的精密测量方法应用[J].计算机技术与应用，2006(4):63-65.
[4]孙杰,潘继飞.高精度时间间隔测量方法综述[J].计算机测量与控,2007,15(2):145-148.
[5]陈晓荣,蔡苹,周红全.基于单片机的频率测量的几种实用方法[J].工业仪表与自动化装置，2003(1):40-42.
[6]谈学.基于单片机的等精度频率计设计[J].重庆工商大学学报，2004,21(2):133-135.
[7]刘夫江.基于单片机和CPLD的等精度数字频率计设计[D].济南：山东大学，2007.
[8]张兆莉,蔡永泉,王钰.基于FPGA的数字频率的设计与实现[J].自动化仪表，2006,27(11):10-13.
[9]金 琳.基于EDA技术的频率计系统设计[D].吉林：吉林大学,2007.
[10]王杰,张涛.一种铁路安全继电器高精度测试系统的实现[J].郑州大学学报,2006,27(4):63-65.
[11]张建平.时间继电器延迟时间的高准确度测量[J]. 宇航计测技术，2003,23(1):12-16.
[12]Joseph Yiu. The Definitive Guide to the ARM Cortex-M3[M]. 北京航空航天大学出版社，2009.
[13]周立功. Cortex-M3开发指南—— 基于LM3S8000.广州致远电子有限公司.2008.
[14]刘同法,肖志刚,彭继卫.ARM Cortex-M3内核微控制器快速入门与应用[M]. 北京:北京航空航天大学出版社，2009.
[15]周立功.EasyARM8962实验教程.广州致远电子有限公司.2008.
[16]熊茂华,杨震伦. ARM9嵌入式系统设计与开发应用[M]. 北京：清华大学出版社,2008.
[17]廖义奎. ARM与FPGA综合设计及应用[M]. 陕西：中国电力出版社，2008.
[18]Wayne W.基于FPGA的系统设计[M].北京：机械工业出版,2006.
[19]王诚,吴继华,范丽珍，等. Altera FPGA/CPLD设计(高级篇)[M]. 北京：人民邮电出版社，2005.
[20]刘延飞. 基于Altera FPGA/CPLD的电子系统设计及工程实践[M]. 北京:人民邮电出版社. 2009.
[21]黄智伟.FPGA系统设计与实践[M].北京：电子工业出版社,2005.
[22]Steve K. Advanced FPGA Design[M]. 北京：机械工业出版社,2009.
[23]欧伟明.基于MCU、FPGA、RTOS的电子系统设计方法与实例[M].北京航空航天大学出版社,2007.
[24]赵艳华.基于Quartus II的FPGA/CPLD设计与应用[M]. 电子工业出版社,2009.
[25]华清远见嵌入式培训中心. FPGA应用开发入门与典型实例[M]. 北京：人民邮电出版社,2008.
[26]刘思久,张礼勇.自动测试系统与虚拟仪器原理•开发•应用[M]. 北京：电子工业出版社,2009.
[27]王先培.测控总线与仪器通信技术[M].北京:机械工业出版社,2007.
[28]黄玉金.GPIB的研究与设计[D].北京:北京化工大学,2000.
[29]张毅刚.计算机自动测试[M]. 哈尔滨:哈尔滨工业大学出版社,2006
[30]吴国庆.现代测控技术及应用[M].北京:电子工业出版社,2007.
[31]秦红磊,路辉,郎荣玲.自动测试系统：硬件及软件技术[M].北京:高等教育出版社,2007.
[32]陈长龄.自动测试及接口技术[M].北京:机械工业出版社,2005.
[33]李维提.液晶显示器应用技术[M].北京:北京邮电学院出版社,1997.
[34]徐爱钧.IAR EWARM嵌入式系统编程与实践[M]. 北京:北京航空航天大学出版社,2006.
[35]周霭如,林伟键.C++程序设计基础[M].北京：电子工业版社，2006.

[36]郑亚民.可编程逻辑器件开发软件 QuartusII[M].北京:国防工业出版社，2006.
[37]周润景，图雅，张丽敏．基于 Quartus II 的 FOGA/CPLD 数字系统设计实例[M]．北京：电子工业出版社，2007.
[38]罗力凡．基于 VHDL 的 FPGA 开发快速入门•技巧•实例[M]．北京:人民邮电出版社．2009.
[39]James R Armstrong, F Gall Gray. VHDL Design Representation and Synthesis Second Edition [M].北京：机械工业出版社，2003.
[40]张永艳.基于复杂可编程逻辑器件及用 VHDL 语言编程的数字频率计的设计[D].呼和浩特：内蒙古大学，2004.
[41]徐洋．基于 Verilog HDL 的 FPGA 设计与工程[M]．北京:人民邮电出版社,2009.
[42]林灶生.Verilog FPGA 芯片设计[M]．北京：北京航天航空大学出版社,2006.
[43]乔庐峰.Verilog HDL 数字系统设计与验证[M]．北京：电子工业版社，2009.
[44]孙晓云，郭立炜，孙会琴．基于 Lab Windows/CVI 的虚拟仪器设计与应用[M]．北京：电子工业版社，2005.
[45]宋宇峰．Lab Windows/CVI 逐步深入与开发实例[M]．北京：机械工业出版社，2003.
[46]张毅刚，乔立岩．虚拟仪器开发环境：Lab windows/CVI6.0 编程指南[M]．北京：机械工业出版社，2002.
[47]雷勇.虚拟仪器设计与事件[M].北京：电子工业出版社，2005.
[48]余成波,冯丽辉,潘盛辉.虚拟仪器技术与设计[M].重庆：重庆大学出版社，2006.

攻读硕士学位期间发表学术论文情况

作者王春燕，杨景常，董慧，康丽奎. 自动化仪表，2010 年，第 1 期：25-28. 主办单位：工业自动化仪表研究院。（硕士学位论文第三章）。

致　谢

对于本论文的各项工作，我的导师杨景常教授给予了热情的关怀和悉心的指导，从论文的选题到论文的审阅均浸透着他辛劳的汗水。在我近三年的研究生生活期间，杨老师以他严谨的治学态度、一丝不苟的工作作风、丰富的教学实践经验，使我收益匪浅，使我在研究生阶段的科研能力得到了很大的提高。

在课题的设计过程中，还得到同实验室同学的大力帮助和支持，在此表示诚挚的感谢。