学 位 论 文 独 创 性 声 明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签　名：陈美娟　日　期：2004.9

关于学位论文使用授权的说明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学研究生院办理。

签　名：陈美娟　导师签名：[signature]　日　期：2004.9

摘　要

一个具有时代特色、形式新颖的实验系统，不仅有利于学生更好地掌握专业知识，而且有利于提高学生的综合素质。东南大学移动通信国家重点实验室开发了一系列通信原理实验系统，这些系统对传统的实验进行了革新，它们将计算机技术与通信教学实验有机结合，涉及硬件电路设计、高级语言程序设计和计算机通信等，实验具有较强的参与性。本人负责开发循环码部分实验系统，编码部分采用硬件实现，译码部分采用软件实现。这两部分可以相互配合，也可以独立应用。学生可以通过一些简单的设置，实现不同循环码的编码；通过 C、C++、Delphi 等高级语言的编程，实现不同循环码的译码。

全文共分五章。

第一章为绪论，介绍了课题背景，给出了论文的内容结构及章节安排。

第二章为编码器硬件平台的实现，是本论文的一个重点，详细介绍了各部分电路的设计和工作原理。

第三章为编码器与译码器的接口设计，介绍使用 51 系列单片机实现串行口通信，完成编码器和译码器的会话。

第四章为译码器软件平台的实现，是本论文的另一个重点，详细描述了如何用 Delphi 和 C/C++高级语言实现循环码的译码。

第五章对系统改进方案作了分析设计。

关键词：循环码　编码器　译码器

Abstract

A better experiment not only is helpful to master of professional knowledge for students, but also is useful to enhance their abilities in various fields. The serial experiments in communication theory have designed and developed by the National Mobile Communications Research Lab of Southeast University. These experiments improve the disadvantage of traditional ones. They combine computer technique with teaching experiment of communication. The students should study the hardware circuit design, high-level language program and computer communication technology in every experiment so they can get the great progresses in the theory combining to practice. In this thesis, the cyclic code experiment system has been developed and implemented. The coder part is completed by hardware and the decoder part is implemented by software. It is proved by experiment that the coder and decoder work rightly whether they do together or separately. Different coders are obtained by changing the set simply and different decoders are also gotten by programming the high-level language such as C, C++, Delphi.

There are five chapters in the thesis.

In chapter 1, we introduce the project background and the architecture of this thesis.

Chapter 2 is one of main contents of this thesis. The implement of coder hardware platform is given. The main circuits of cyclic coders are designed and developed in detail and their principles are discussed.

In chapter 3, the design of the interfaces of coder and decoder is studied, the serial interface port of MCS-51 is introduced and the session between the coder and decoder is completed.

Chapter 4 is another important one of this thesis. The implement of decoder software platform is given and how to design decoder using Delphi and C/C++ to program is analyzed.

In chapter 5, the improved system design scheme is given and analyzed.

Keywords: Cyclic code Coder Decoder

第一章 绪 论

通信系统的基本问题是保证信息的可靠传送，然而由于传输过程中不可避免地要受到干扰的影响，使信号码元波形变坏，所以信号传输到接收端后可能发生错误判决。所以必须采用一定的差错控制技术降低差错率，提高通信的可靠性。

通信中差错控制主要涉及三个方面的问题，即：纠检错编码、信道的数学模型、差错控制方式。本课题主要是针对纠检错编码技术开发相应的实验系统。

本课题的目的是为配合高校通信类学生的“现代通信原理实验”而设计，通过使用本系统，有助于学生掌握专业知识，对于增强学生创新意识、提高学校教学质量将会起到积极的作用。

本人负责实现循环码的编码和译码。传统的通信原理实验存在着诸多问题，例如实验设备落后，没有较好地使用计算机技术；实验的参与性差，多数为演示性实验，不利于激发学生的创新精神和提高实践能力等。本次开发的系统，提出了一种基于“硬件编码和软件解码”的新型循环码实验系统，它将计算机技术与通信教学实验有机结合，涉及硬件电路设计、高级语言程序设计和计算机通信等。实验可参与性强，参与点涉及循环码编译码的每一个关键知识点。编码器部分主要通过一些拨码开关实现编码器的通用性，译码器部分通过完善高级语言程序实现译码器的通用性。

译码器与编码器可配合使用，也可独立使用。本系统也可以作为一个研发平台，在此基础之上学生可实现其他类型循环码的设计。

全文共分五章。

第一章为绪论，介绍了课题背景，给出了论文的内容结构及章节安排。

第二章为编码器硬件平台的实现，是本论文的一个重点，在对编码器部分的功能和结构作了简述后，详细描述了数据源产生及数据显示、信息流产生、信息位形成，控制电路设计、除法电路设计、编码结果显示等电路的结构和工作原理。

第三章为编码器与译码器的接口设计，介绍使用 51 系列单片机实现计算机串行口通信，完成软/硬件之间的数据传递。这部分设计包括 51 芯

片硬件的连接和软件编程。

第四章为译码器软件平台的实现，是本论文的另一个重点，在对译码器部分的功能和结构做了简述之后，详细描述了如何使用 Delphi 和 C/C++语言实现循环码的译码。

第五章对系统改进方案作了分析设计。

第二章 循环码编码器的设计

循环码编码器采用硬件实现，在选用逻辑器件时，出发点有两个：一是器件逻辑功能简单，便于学生理解电路工作原理；一是学生能够方便地控制器件的工作情况。所以我选用的逻辑器件以 74 系列芯片为主，这样，我们可以在电路的任何地方设置参与点与观察点，从而对整个编码过程有一个很清楚的了解。

考虑到编码器的通用性，实现时在电路中多处采用了开关。通过开关的不同设置，就可得到不同类型的循环码编码器。另一方面，只有当学生理解了循环码的编码原理之后，才能正确地设置开关值。所以，通过开关的设置，促使了学生对循环码编码器理论的灵活掌握。

循环码可用 (n,k) [1]来表示，n 为码长，k 为信息位长，$r = n - k$ 是监督位长。本系统可实现 6 种循环码的编码，如表 1 所示。

表 1 循环码类型表

循环码	生成多项式
(7,4)_1	$x^3 + x + 1$
(7,4)_2	$x^3 + x^2 + 1$
(7,3)_1	$x^4 + x^3 + x^2 + 1$
(7,3)_2	$x^4 + x^2 + x + 1$
(15,7)	$x^8 + x^7 + x^6 + x^4 + 1$
(15,5)	$x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$

系统设置了 4 个参与点，即 4 组拨码开关，用于实现通用循环码编码器，即得到不同信息位的编码。

系统设置了一些观察点，主要用于观察输入信息位、反馈信号和编码结果等。

编码器可用一个带反馈的最长线性移位寄存器来实现[2]，对应生成多项式 $g(x) = x^3 + x + 1$，(7,4)循环码的编码器结构如图 1 示。

循环码编码器的设计，主要是围绕如何实现上面的结构图，并使之具有通用性。实现的主要部分如下所述：

1. 输入信息位

应做到系统能够很方便地输入信息位，而且可以任意改变信息位，系统可对任意信息位进行编码。信息位的长度也可灵活改变，以满足 6

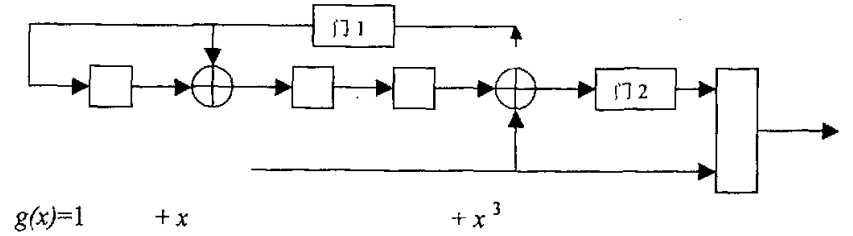


图 1（7，4）循环码编码器结构图

种循环码的编码需要。

这里用到了两组拨码开关，一组用于输入信息位，一组用于控制信息位的长度。

2. 除法电路实现

除法电路用反馈移位寄存器实现。移位寄存器的级数由循环码的监督位 r 决定，由于是通用编码器，所以级数长度是可变的。

通过一组拨码开关设置移位寄存器的级数。

需要在输入端引入反馈的那些移位寄存器，是对应于循环码生成多项式中系数不为零的那些项，因为要通用，所以，这个编码器的反馈可引入至不同的移位寄存器。

通过一组拨码开关来设置给哪些级引入反馈。

3. 门 1、门 2 实现

门 1 是在输入信息位时打开，在信息位输入结束时关闭，作用是让反馈信号通过。我们的信息位是按照时钟 clk 的节拍来输入的，所以，门 1 应在信息位输入时打开，持续 k 个 clk 时钟周期后关闭。

门 2 是在信息位输入结束后打开，作用是让监督位输出。所以，在设计时，控制在信息位有效后 k 各 clk 时钟周期后门 2 打开。

4. 显示

为了增加试验结果的可观察性，系统中设计了编码结果显示电路。为了与输出相对应，输入的信息位也有显示电路。同时，系统还设计了示波器观察点，以方便学生用示波器观察波形。

编码器的具体实现下面将做详细介绍。

2．1 数据源产生及数据显示

为了方便编码器的数据输入，数据源采用并行输入方式，由一组拨码开关 S3 控制。数据值可取‘0’或‘1’。数据源产生及显示电路如图 2 所示。

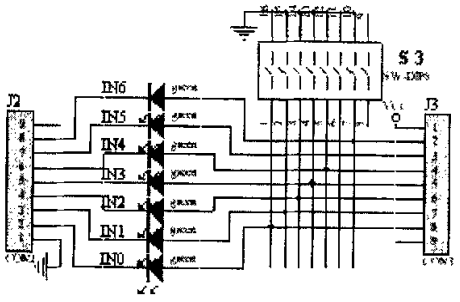


图 2 数据源产生实现图

电路工作原理说明如下：

J3 为一 680Ω排阻，S3 为一拨码开关组，IN0～IN6 为发光二极管，J2 为一 680Ω排阻。

J3 的公共端接电源，排阻另一端接 S3 及发光二极管；S3 一端接，另一端接 J3 及发光二极管。J2 公共端接地，另一端接发光二极管。这样，当闭合 S3 某开关个时，就是设置相应输入数据位为‘0’，发光二极管灭；打开时，就是设置相应输入数据位为‘1’，发光二极管亮。

本系统可实现循环码有（7，3）、（7，4）、（15，5）和（15，7），数据位最多为 7 位，故拨码开关 S3 用到了 7 位。

2．2 信息流产生

数据源输入的是并行数据，但后续电路是按串行的方式处理数据的，所以对于前面输入的并行数据，需要进行一个并/串变换。电路如图 3 所示。

电路工作原理说明：

S2：按键式开关。

U25：MAX708,一种微处理器电源监控芯片，可同时输出高电平有效和低电平有效的复位信号。

U1:74LS161,4 位同步二进制计数器。

U2：74HC165，8 位移位寄存器。

U3：74LS74，双上升沿 D 触发器。

✧ 按键式开关 S2 与 MAX708 共同作用，为编码器提供负脉冲。

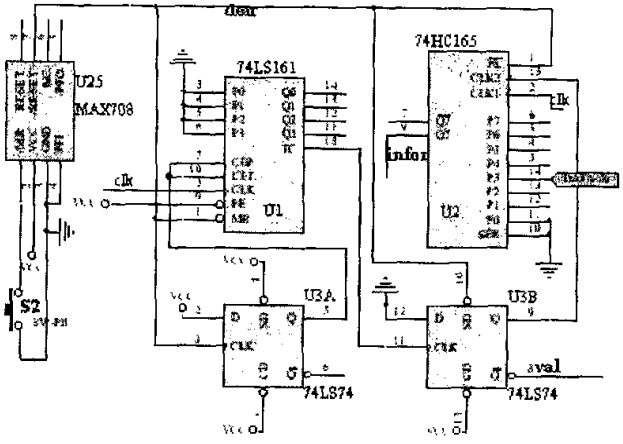


图 3 信息流产生电路实现图

开关 S2 按下时，MAX708 输出负脉冲，在图 3 中，这个负脉冲作用是：

1. 置 D 触发器 U3B 的初始状态为‘1’；
2. 给 U3A 的 CLK 一个时钟上升沿。
3. 二进制同步计数器 74LS161 清 0；
4. 使并/串变换器 74LS165 进行采样。

负脉冲到来时，74LS161 清零，当 S2 打开时，74LS161 的 CEP/T 端为高电平，开始计数。

✧ 工作时钟的安排：

1. 74LS165 用系统时钟 CLK；
2. 74LS161 用系统时钟 CLK；
3. U3A 的时钟是复位信号；
4. 74LS74 的时钟信号来自 74LS161-TC 的输出。

✧ 74LS74 的作用：

U3B 的 CD 端接电源，SD 端接负脉冲，输入端 D 接地，所以它的初始状态为高电平。当 74LS161 的 TC 端输出正脉冲后，它的时钟信号有效，输出低电平。

✧ 74LS161 的作用：

MR 端接负脉冲，用于清 0。当控制开关送来高电平，CEP/T 端有效，74LS161 开始计数。计 15 个 CLK 之后，TC 端输出一个正脉冲。

✧ 并/串变换器设计：

并/串变换器采用 74LS165，并行输入的数据连接到 P7、P6、P5、P4、P3、P2、P1。因为串行信号按照 P7-P1 的顺序输出，所以，P7 是数据的高位。P0 没有用到，所以接地。

/PL 是数据采样控制信号，低电平期间采样数据。用复位开关提供一个负脉冲，这样当我们提供一个负脉冲时就可采样一次数据，实现了数据改变的随时性。

74LS165 有两个与时钟有关的信号端 CLK1 和 CLK2，其中 CLK1 是正常时钟，而 CLK2 是一个时钟禁止端，只有当 CLK2 低电平有效时，CLK1 才起作用。工作波形[10]如图 4 示。

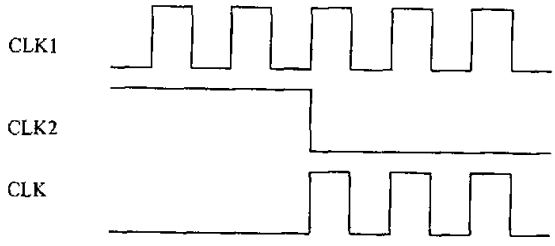


图 4 时序图

✧ 并/串变换电路工作原理：

编码器启动由开关 S2 触发，闭合 S2，产生一个负脉冲，这时 74LS165 采样并行数据、74LS161 清 0、U3B 置 1；打开开关 S2，74LS161 开始计数，15 个 CLK 周期后 TC 端输出一正脉冲；U3B 的时钟有效，输出低电平；74LS165 的 CLK2 有效；74LS165 的 9 端开始输出串行数据。

U3B 的/Q 端是在编码器工作 15 个 CLK 周期后，输出高电平。此时，74LS165 也正好开始输出串行数据，这样，/Q 端可作为数据有效指示端。在后续电路中要用到这个信号。

2．3 信息位形成

74LS165 输出信号的初始状态取决于最高输入端 P7，P7 的取值来自于数据源产生电路，由拨码开关 S2 控制其状态。编码器的输入是灵活的，可以任意设置，所以，74LS165 输出的串行数据其初始状态就不定。但为了后续电路能够比较方便地处理数据，希望数据位的初始状态为低电平。我们把后续电路处理的数据称为信息位，所以在这里应设计电路来使信息位的初始状态为低电平。

另外，串行数据的输出最长是 7 位的，但我们实际使用的时候只取前面 k 位作为信息位。所以应设计电路从串行数据中中截取前 k 位。

实现电路如图 5 所示。

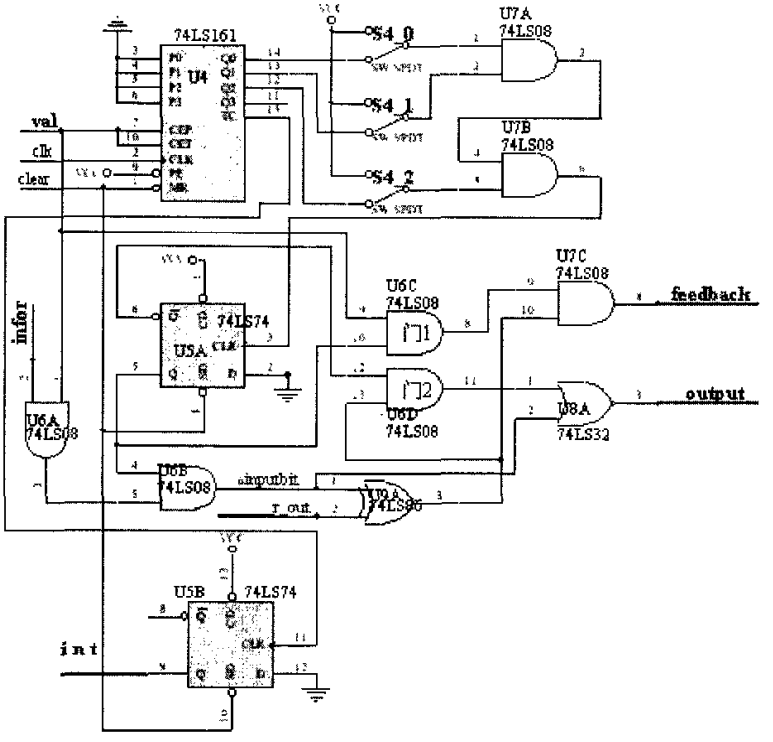


图 5 信息位形成及控制电路实现图

和信息位形成有关的器件有：U4、S4、U7、U5、U6。

电路工作原理如下：

✧ 与门 U6A：使数据位初始状态为低电平

U6A-1：接并/串变换电路中 74LS74 的/Q 端，即前所述的数据位有效端。系统启动时，数据位有效端为低电平，当数据位有效后，该端为高电平。

U6A-2：接 74LS165 的串行输出数据，其初始状态受输入数据影响，状态不定。

所以 U6A-3 端就输出初始状态为低电平的数据位。

这个数据位输入至 U6B-5。

✧ 与门 U6B：截取信息位

具体实现过程是：

1. 4 位二进制同步计数器 74LS161 在数据位有效后开始计数，我们要从这里取得信息位长 k 的值。当 74LS161 计 k 个系统时钟后，U6B-42 应从高电平变为低电平，使 U6B 关闭，这样就可截取信息位。

2. 开关组 S4 的作用就是设置 k 值，它们的一端全部短连，接至高电平，另一端分别连接至 74LS161 的输出端 Q2、Q1、Q0 端。

开关设置与 k 值的对应关系如表 2 所示。

表 2 信息位长度控制表

开关 S4			K
S4_2	S4_1	S4_0	
V_{CC}	Q_1	Q_0	3(011B)
Q_2	V_{CC}	V_{CC}	4(100B)
Q_2	V_{CC}	Q_0	5(101B)
Q_2	Q_1	Q_0	7(111B)

3. 与门 U7A、U7B 串联，输出作为 U5A(74LS74-D 触发器)的时钟信号。

4. U5A 触发器的 CD 端接地 GND、SD 端接负脉冲、输入端 D 接地。所以 U5A 触发器的初始状态为高电平，当时钟信号有效后，输出低电平。即当信息位长到时 U5A 输出低电平。

假设数据位为 1010110，k=5 时此电路工作波形如图 6 所示。

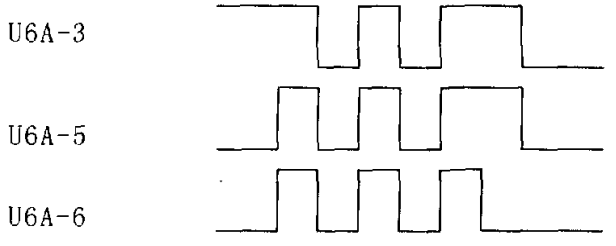


图 6 信息位形成波形图

2.4 控制电路设计

这里所说的控制电路指的是前面编码器原理图中的门 1 和门 2。当输入信息位时，门 1 打开，反馈信号通过；信息位结束时，门 1 关闭，切断反馈信号。门 2 是在信息位结束时打开，用于输出除法电路的余数，即循环码的监督位。

电路如上面图 5 所示，和控制电路有关的器件是：U5、U6、U7、U8、U9。

电路工作原理如下：

U9A：两个输入端分别连接信息位和除法电路的输出，所以 U8B 的输出是反馈信号。

U6C： -9 接数据位有效信号；

-10 接前面信息流形成电路（图 3）中 D 触发器输出端 Q，即信息位结束时变为低电平。

所以 U6C 输出高电平，时间是从信息位有效开始，持续 k 个 clk 周期。

U7C：在信息位有效期间输出反馈信号，在信息位结束后输出低电平，即切断反馈信号。

U6D： -12 接信息流形成电路（图 3）中 D 触发器的/Q 端，即信息位结束时变为高电平有效；

-13 接反馈信号。

所以 U6D 在信息位结束后输出反馈信号，这时的反馈信号就是循环码的监督位。

U8A：输出编码器的编码结果，它是系统码的形式，即前面是 k 位信息位，后面是 r 位监督位。

2．5 除法电路设计

除法电路由带反馈的最长线性移位寄存器构成，主要用到的逻辑器件是 D 触发器 74LS74 和异或门 74LS86。在这里要解决的问题主要有两个：反馈信号的引入和除法电路的输出。

✧ 反馈信号的引入

在循环码编码器电路中，引入反馈信号的地方就是对应生成多项式系数不为零的地方。所以对于不同循环码编码器，需要引入反馈信号的数量和位置不同。本系统设计成一个通用形式的，即，反馈信号的引入位置可灵活设置，满足不同循环码的需要。

✧ 除法电路的输出

不同的循环码，最长线性移位寄存器的长度不同。它由监督位长决定。我们设计的编码器，应该能够适应这个可变的长度。这就意味着除法电路从哪一级输出应该是灵活的。

除法电路原理图如图 7 所示。

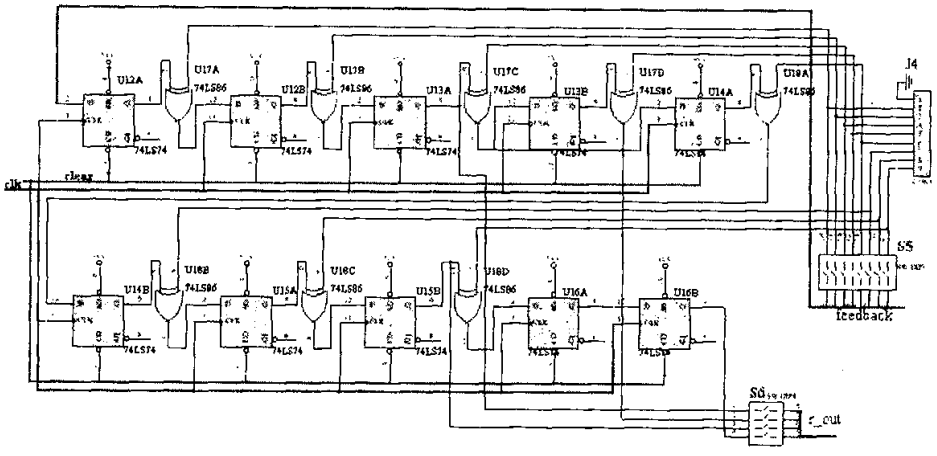


图 7　除法电路实现图

电路中所用的逻辑器件：

74LS74：双上升沿 D 触发器；74LS86：4 二输入异或门；J4：3.3kΩ的排阻；S5：控制反馈信号的拨码开关；S6：控制除法电路输出的拨码开关。

电路工作原理如下所述：

✧ **移位寄存器 D 触发器：**

所有的 D 触发器其时钟端 CLK 接系统时钟，CD 端接负脉冲，SD 端接电源，所以移位寄存器的初始状态为低电平。输入端 D 或接反馈信号或接异或门的输出，输出端或接异或门或作为除法电路的输出，它们都是要设计成可以灵活连接的方式。

✧ **开关组 S5：**

开关组 S5 一端全部短连，用于连接输入的反馈信号；另一端一方面经电阻接地，一方面作为异或门的输入。电阻的取值是保证：当开关打开时异或门输入为低电平，闭合时异或门输入为反馈信号，电路中阻值为 3.3kΩ。

所以开关组 S1 的作用是决定哪些级连接反馈信号。

✧ **开关组 S6：**

开关组 S6 一端全部短连，用于作为除法电路的输出，另一端连接一些 D 触发器的输出端，这些 D 触发器就是那些可以作为除法电路输出的触发器。

例如：对应生成多项式为 $g(x) = x^3 + x^2 + 1$ 的 (7，4) 循环码，第 2 级需引入反馈信号，所以开关组 S5 中与第 2 级异或门连接的开关闭合；移位寄存器应有 3 级，即除法电路从第 3 级输出，所以开关组 S6 中与第 3 级 D 触发器输出端连接的开关闭合。

设计的编码电路可实现 6 种循环码的编码，它们的生成多项式如前表 1 所示。

在这 6 种循环码中，我们看到，监督位最长的是（15，5）码，r = 10，所以在设计除法电路时，用了 10 级 D 触发器，即 5 个 74LS74。

这 6 种循环码，它们的监督位长分别为 3、4、8、10，这就是说，第 3、4、8、10 级 D 触发器的输出端可以作为除法电路的输出。所以电路图 7 中 S6 为 4 个一组的开关组，开关与输出级的对应关系如表 3 所示。这 6 种循环码，对应生成多项式系数不为零级数如表 4 所示。

表 3 输出级开关功能表

输出级	S6			
	1	2	3	4
第 3 级	闭合			
第 4 级		闭合		
第 8 级			闭合	
第 10 级				闭合

表 4 生成多项式系数表

码型	1	2	3	4	5	6	7	8	9	10
(7,4)_1	√		√							
(7,4)_2		√	√							
(7,3)_1		√	√	√						
(7,3)_2	√	√		√						
(15,7)				√		√	√	√		
(15,5)	√	√		√	√			√		√

通过表 4 我们可以看到，需要引入反馈的级数可能是第 1、2、3、4、5、6、7、8、10 级，去掉要作为输出的那些级，实际电路设计时需要引入反馈的是第 1、2、3、4、5、6、7、8 级。所以我们用了一个 8 个一组的开关组 S5，开关与各级反馈的对应关系如表 5 所示。

表 5 反馈开关功能表

S1	1	2	3	4	5	6	7	8
(7,4)_1	闭合							
(7,4)_2		闭合						
(7,3)_1		闭合	闭合					
(7,3)_2	闭合	闭合						
(15,7)				闭合		闭合	闭合	
(15,5)	闭合	闭合		闭合	闭合			闭合

2．6 编码结果显示

为了编码结果的直观性，同显示数据位一样，我们也采用发光二极管来显示编码的结果。从前面的叙述中我们看到，编码结果是串行输出的，为了显示方便，我们先进行了串/并变换，然后送显示电路及后续电路。

编码结果显示电路如下图 8 所示。

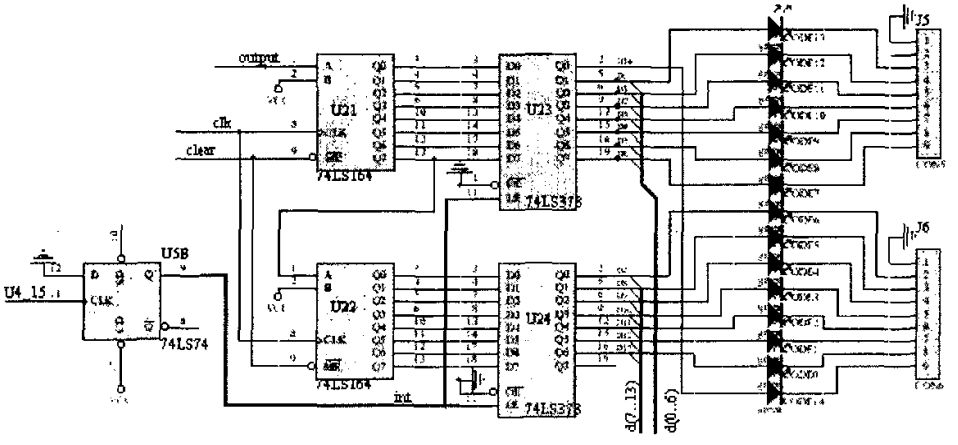


图 8 编码结果显示电路

此电路功能分为两部分：

一、串/并变换

编码器的编码结果，我们一方面是要显示，另一方面要送至计算机进行译码，我们需要一组稳定的数据，所以让编码结果经过一个串/并变换器，变为并行信号后再送后续电路。

图 8 中完成并/串变换功能的器件有：U21 和 U22

U21/U22:74LS164 是 8 位移位寄存器，编码结果最多是 15 位，所以用了两片 74LS164，采用串连的方式。74LS164 有两个输入端 A、B，我们只用到了一个，两个 74LS164 都用 A，所以 B 接高电平。时钟端 CLK 接系统时钟，/MR 端接系统负脉冲，U21 输出端 Q7 连接至 U22 的输入端 A，用于两个芯片的级连。

电路工作原理如下：

系统启动后，74LS164 清 0，稍后，编码结果送至 U16 输入端 A，在时钟的作用下，编码结果开始做移位操作，15 个 CLK 之后，全部编码出现在 74LS164 的输出端。

注意电路中最下面是数据的最高位。

二、 编码结果显示

上述串/并变换电路，在时钟的作用下会一直进行移位操作，我们需要的是从信息位有效后的 15 位数据，这是编码结果。如果不加控制，74LS164 会一直进行移位操作，我们就取不到编码结果，所以需要一个功能电路，能够锁存住需要的 15 位编码结果，并把它显示出来。

图 8 中完成编码结果显示功能的器件有：U5B、U23、U24、J5、J6。

U5B 是 74LS74，它的时钟是 U4_15，即系统启动后 15 个时钟周期，U4_15 输出正脉冲信号，这时 U5 的输出端就由初始的高电平变为低电平。

U23/U24 是锁存器 74LS373。U23、U24 的数据输入端接串/并变换的输出，U24 接高位，U23 接低位。输出使能端/OE 接地，使输出有效；锁存信号 LE 接 U5B 的输出，在系统 15 个时钟时 15 个编码位全部输出至串/并变换器的输出端，同时送到锁存器 74LS373 的输入端，所以在第 16 个系统时钟到来时我们让锁存控制信号端 LE 有效，对编码输出做了锁存。

J5/J6 是阻值为 220Ω的排阻。

2．7 编码器总结

编码电路的设计可以满足 6 种循环码，实现的循环码编码器小结如下：

一、(7,4)_1- $g(x)=x^3+x+1$

开关 S4_2 接 Q2，S4_1 接电源，S4_0 接电源，设置信息位长为 4；

开关组 S5 的 1 闭合，设置第一级移位寄存器接反馈信号；

开关组 S6 的 1 闭合，设置除法电路从第三级输出。

编码器电路如图 9 所示。

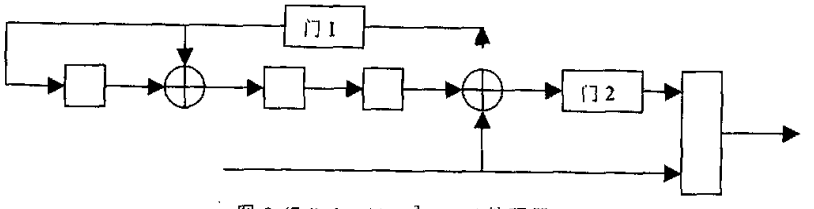


图 9 (7,4)_1- $g(x)=x^3+x+1$ 编码器

编码结果如下：

序号	信息位	监督位
0	0000	000
1	0001	011
2	0010	110
3	0011	101
4	0100	111
5	0101	100
6	0110	001
7	0111	010
8	1000	101
9	1001	110
10	1010	011
11	1011	000
12	1100	010
13	1101	001
14	1110	100
15	1111	111

二、(7,4)_2 - $g(x)=x^3+x^2+1$

开关 S4_2 接 Q2，S4_1 接电源，S4_0 接电源，设置信息位长为 4；

开关组 S5 的 2 闭合，设置第二级移位寄存器接反馈信号；

开关组 S6 的 1 闭合，设置除法电路从第三级输出。

编码器电路如图 10 所示。

编码结果如下：

序号	信息位	监督位

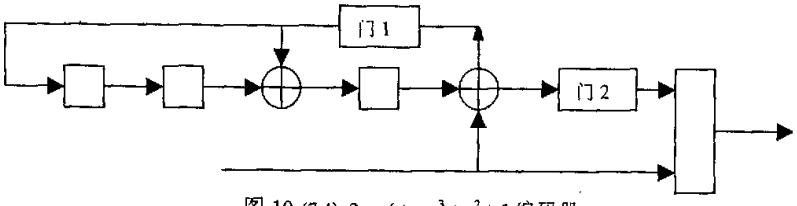


图 10 (7,4)_2 - $g(x) = x^3 + x^2 + 1$ 编码器

0	0000	000
1	0001	101
2	0010	111
3	0011	010
4	0100	011
5	0101	110
6	0110	100
7	0111	001
8	1000	110
9	1001	011
10	1010	001
11	1011	100
12	1100	101
13	1101	000
14	1110	010
15	1111	111

三、(7,3)_1 - $g(x) = x^4 + x^3 + x^2 + 1$

开关 S4_2 接电源，S4_1 接 Q1，S4_0 接 Q0，设置信息位长为 3；
开关组 S5 的 2、3 闭合，设置第二、三级移位寄存器接反馈信号；
开关组 S6 的 2 闭合，设置除法电路从第四级输出。
编码器电路如图 11 所示。

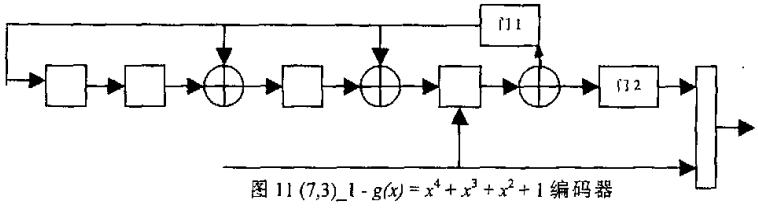


图 11 (7,3)_1 - $g(x) = x^4 + x^3 + x^2 + 1$ 编码器

编码结果如下：

序号	信息位	监督位
0	000	0000
1	001	1101
2	010	0111
3	011	1010

4	100	1110
5	101	0011
6	110	1001
7	111	0100

四、(7,3)_2 - $g(x) = x^4 + x^2 + x + 1$

开关 S4_2 接电源，S4_1 接 Q1，S4_0 接 Q0，设置信息位长为 3；

开关组 S5 的 1、2 闭合，设置第一、二级移位寄存器接反馈信号；

开关组 S6 的 2 闭合，设置除法电路从第四级输出。

编码器电路如图 12 所示。

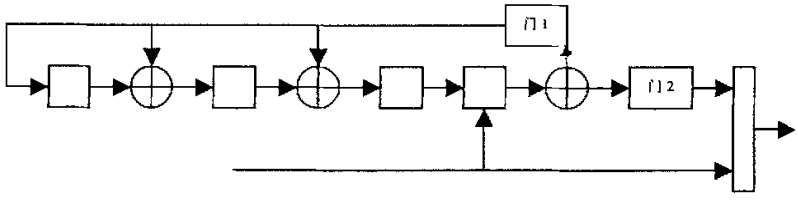


图 12 (7,3)_2 - $g(x) = x^4 + x^2 + x + 1$ 编码器

编码结果如下：

序号	信息位	监督位
0	000	0000
1	001	0111
2	010	1110
3	011	1001
4	100	1011
5	101	1100
6	110	0101
7	111	0010

五、(15,7) - $g(x) = x^8 + x^7 + x^6 + x^4 + 1$

开关 S4_2 接 Q2，S4_1 接 Q1，S4_0 接 Q0，设置信息位长为 7；

开关组 S5 的 4、6、7 闭合，设置第四、六、七级移位寄存器接反馈信号；

开关组 S6 的 3 闭合，设置除法电路从第八级输出。

编码器电路如图 13 所示。

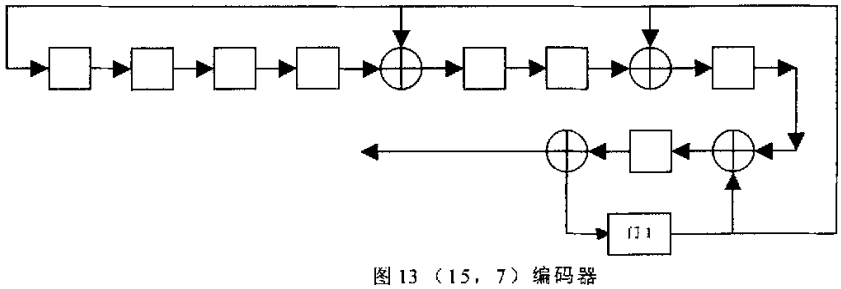


图 13 （15，7）编码器

编码结果如下：

序号	信息位	监督位
0	0000000	00000000
1	0000001	11010001
2	0000010	01110011
3	0000011	10100010
4	0000100	11100110
5	0000101	00110111
6	0000110	10010101
7	0000111	01000100
8	0001000	00011101
9	0001001	11001100
10	0001010	01101110
11	0001011	10111111
12	0001100	11111011
13	0001101	00101010
14	0001110	10001000
15	0001111	01011001
16	0010000	00111010
17	0010001	11101011
18	0010010	01001001
19	0010011	10011000
20	0010100	11011100
21	0010101	00001101
22	0010110	10101111
23	0010111	01111110
24	0011000	00100111
25	0011001	11110110
26	0011010	01010100
27	0011011	10000101
28	0011100	11000001
29	0011101	00010000
30	0011110	10110010
31	0011111	01100011
32	0100000	01110100
33	0100001	10100101
34	0100010	00000111
35	0100011	11010110
36	0100100	10010010
37	0100101	01000011
38	0100110	11100001
39	0100111	00110000
40	0101000	01101001
41	0101001	10111000
42	0101010	00011010
43	0101011	11001011
44	0101100	10001111
45	0101101	01011110
46	0101110	11111100
47	0101111	00101101
48	0110000	01001110
49	0110001	10011111
50	0110010	00111101

51	0110011	11101100
52	0110100	10101000
53	0110101	01111001
54	0110110	11011011
55	0110111	00001010
56	0111000	01010011
57	0111001	10000010
58	0111010	00100000
59	0111011	11110001
60	0111100	10110101
61	0111101	01100100
62	0111110	11000110
63	0111111	00010111
64	1000000	11101000
65	1000001	00111001
66	1000010	10011011
67	1000011	01001010
68	1000100	00001110
69	1000101	11011111
70	1000110	01111101
71	1000111	10101100
72	1001000	11110101
73	1001001	00100100
74	1001010	10000110
75	1001011	01010111
76	1001100	00010011
77	1001101	11000010
78	1001110	01100000
79	1001111	10110001
80	1010000	11010010
81	1010001	00000011
82	1010010	10100001
83	1010011	01110000
84	1010100	00110100
85	1010101	11100101
86	1010110	01000111
87	1010111	10010110
88	1011000	11001111
89	1011001	00011110
90	1011010	10111100
91	1011011	01101101
92	1011100	00101001
93	1011101	11111000
94	1011110	01011010
95	1011111	10001011
96	1100000	10011100
97	1100001	01001101
98	1100010	11101111
99	1100011	00111110
100	1100100	01111010
101	1100101	10101011
102	1100110	00001001
103	1100111	11011000
104	1101000	10000001
105	1101001	01010000

106	1101010	11110010
107	1101011	00100011
108	1101100	01100111
109	1101101	10110110
110	1101110	00010100
111	1101111	11000101
112	1110000	10100110
113	1110001	01110111
114	1110010	11010101
115	1110011	00000100
116	1110100	01000000
117	1110101	10010001
118	1110110	00110011
119	1110111	11100010
120	1111000	10111011
121	1111001	01101010
122	1111010	11001000
123	1111011	00011001
124	1111100	01011101
125	1111101	10001100
126	1111110	00101110
127	1111111	11111111

六、(15,5) - $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$

开关 S4_2 接 Q2，S4_1 接电源，S4_0 接 Q0，设置信息位长为 5；

开关组 S5 的 1、2、4、5、8 闭合，设置第一、二、四、五、八级移位寄存器接反馈信号；

开关组 S6 的 4 闭合，设置除法电路从第十级输出。

编码器电路如图 14 所示。

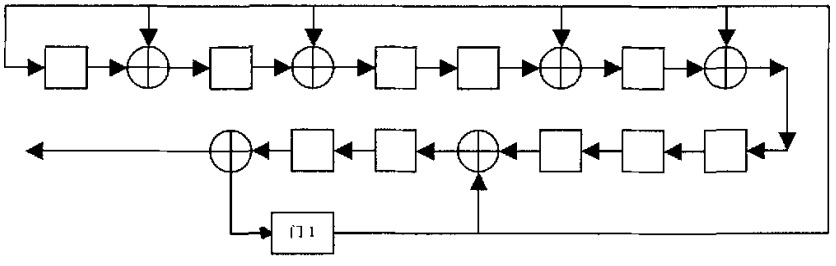


图 14 （15，5）编码器

编码结果如下：

序号	信息位	监督位
0	00000	0000000000
1	00001	0100110111
2	00010	1001101110
3	00011	1101011001
4	00100	0111100011
5	00101	0011011100
6	00110	1110000101

7	00111	1010110010
8	01000	1111010110
9	01001	1011100001
10	01010	0110111000
11	01011	0010001111
12	01100	1000111101
13	01101	1100001010
14	01110	0001010011
15	01111	0101100100
16	10000	1010011011
17	10001	1110101100
18	10010	0011110101
19	10011	0111000010.
20	10100	1101110000
21	10101	1001000111
22	10110	0100011110
23	10111	0000101001
24	11000	0101001101
25	11001	0001111010
26	11010	1100100011
27	11011	1000010100
28	11100	0010100110
29	11101	0110010001
30	11110	1011001000
31	11111	1111111111

2．8 本章小结

本章是本论文的一个重点，说明了如何实现通用循环码编码器，详细介绍了主要电路的实现方法，包括数据源的产生及数据位显示、信息流产生，信息位形成、控制电路设计、除法电路设计、编码结果显示等电路。最后总结了本系统可以实现的 6 种循环码编码器，以及它们的编码结果。

本系统的特点是：各部分电路功能清晰，实验参与性强，操作简单灵活。通过设置 4 组拨码开关就可构成不同循环码的编码器。编码器的输入、输出数据均有显示电路。

第三章 编码器与译码器的接口设计

编码器采用硬件实现，而循环码的硬件译码电路与编码器是相似的，为了避免知识点的重复，译码器采用软件来实现。这样，就存在硬件编码器和软件译码器的接口问题。这一章的主要任务，就是完成这个接口的设计。

把硬件数据读入计算机，可采用串行口进行，所以接口设计就是串行口通信设计。这里采用 MCS-51 系列可编程芯片来完成。

接口部分的设计包括：

1. 硬件电路的实现

完成串行口的连接，采用单片机 MCS-51 将编码结果输入至计算机的串行口，解决 MCS-51 的硬件连接。

2. 软件实现

即对 MCS-51 进行编程，完成读编码器的结果和把编码结果送至计算机串行口。

3．1 接口电路的实现

单片机 MCS-51 有 3 个 8 位的数据端口：P0、P1、P2；编码器的输出结果最多有 15 位，所以把编码结果送至 MCS-51 的 PORT1 和 PORT2[4]。

设计时 MCS-51 是采用中断触发的方式来读编码结果的，此中断信号是在锁存编码结果时有效，中断信号是下降沿有效。

MCS-51 芯片的输入/输出是 TTL 电平，而计算机的串行口是 RS-232 电平，所以，对 MCS-51 的信号需经过一个电平变换才可以接计算机串行口。MAX232 芯片可完成这个功能。

串行口连接器采用简化的 9 针连接器。

电路设计如图 15 所示。

从端口 PORT1、 PORT2 读入编码结果，中断信号连接至 INT0，中断信号来自 U5B－D 触发器的输出，即 16 个 CLK 时钟周期后，U5B 的输出端由高电平变为低电平，下降沿作为 INT0 的中断触发信号。通知 MCS-51 可以读取编码结果。

MCS-51 的数据输出 TXD、数据输入 RXD 端是 TTL 电平，经过

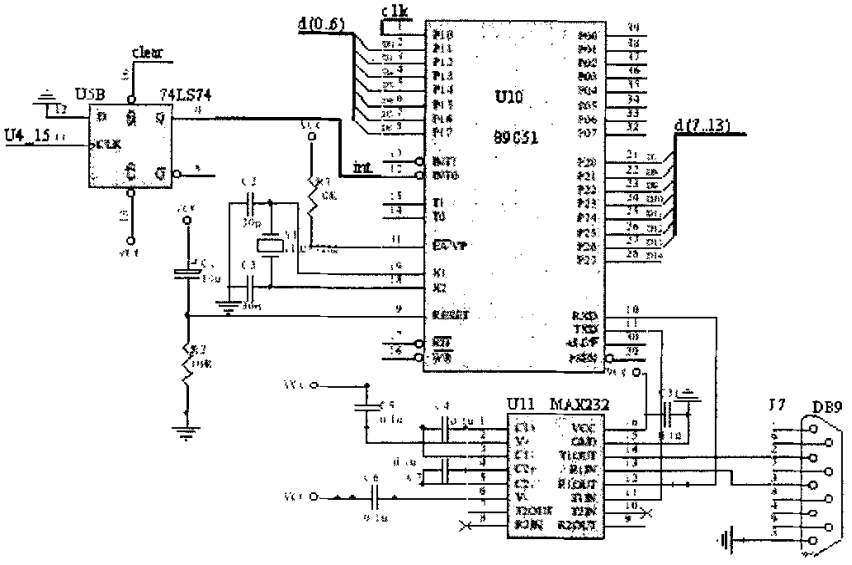


图 15 编码器与译码器的接口电路图

MAX232C 后，TTL 电平转换为 RS-232 电平，之后可连接 RS-232 接口。

接口电路的任务，就是把编码结果输入至计算机，与计算机之间进行简单的会话，所以， RS-232 接口的连接只用到了 3 根线，即：2 脚（TXD）、3 脚（RXD）、5 脚（GND）。

如图 15 的电路连接方式，MCS-51 采用内时钟方式，晶振频率为 11.0592Mhz。采用上电自动复位方式。

3．2 接口程序的实现

对单片机 MCS-51 进行编程。

MCS-51 在本系统中的作用有两个：一是产生系统所需时钟，即 clk 信号；二是完成编码器数据在串行口的可靠传递。

对 51 的编程主要完成 4 项任务：

1. 对 MCS-51 初始化[4]；

初始化时的设置：在模式控制寄存器 TMOD 中设置定时器/计数器设置为自动重装方式；

设置定时器/计数器 1 的初值；

在串口控制寄存器 SCON 设置 MCS-51 为 10 位异步收发方式；

在控制寄存器 TCON 中设置 INT0 为下降沿触发；

2. 读 POR1、PORT2 数据；

读编码器结果是在外部中断 INT0 的中断程序中实现的。

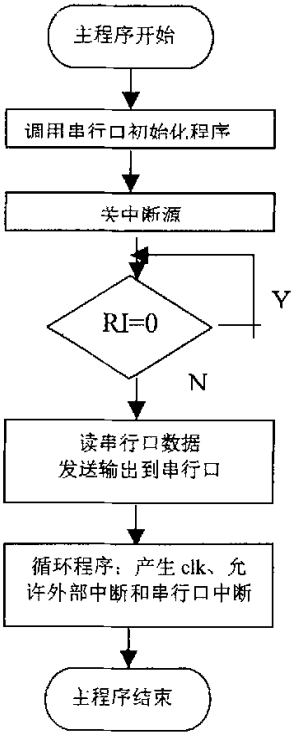
因为在硬件连接时采用的中断触发信号是与 74LS373 的锁存信号同时有效的，为了使数据有一段稳定时间，所以，在读端口数据前采用了软件延时。

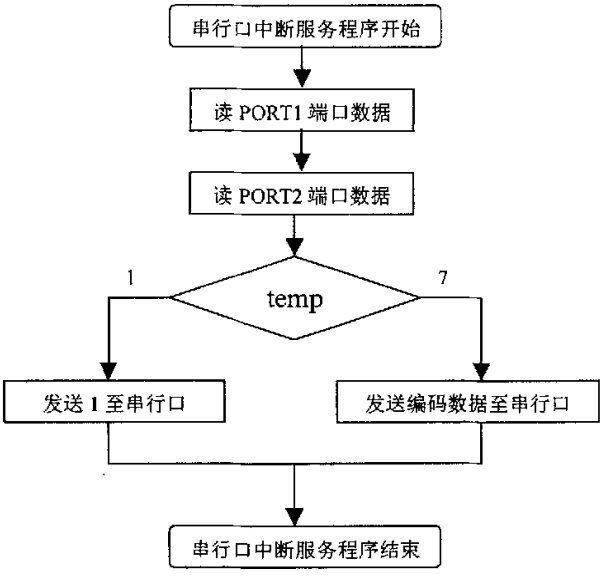
3. 51 与计算机的通信

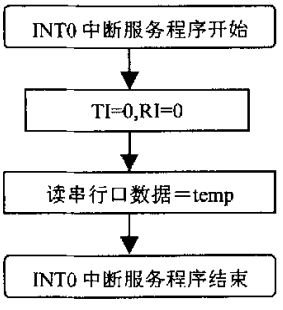
编码器系统是在计算机查询了试验板之后开始工作，所以，51 能够接收计算机的查询数据，并传送给相应的数据给计算机。

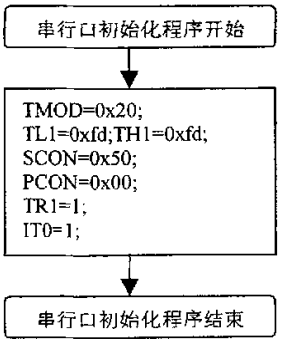
4. 产生编码器系统的时钟

51 程序流程如下：




串行口中断服务程序开始
读 PORT1 端口数据
读 PORT2 端口数据
temp
1
7
发送 1 至串行口
发送编码数据至串行口
串行口中断服务程序结束


INT0 中断服务程序开始
TI=0,RI=0
读串行口数据＝temp
INT0 中断服务程序结束


串行口初始化程序开始
TMOD=0x20;
TL1=0xfd;TH1=0xfd;
SCON=0x50;
PCON=0x00;
TR1=1;
IT0=1;
串行口初始化程序结束

3．3 本章小结

上一章说明了编码器的工作原理和实现方法，这一章主要说明了如何实现硬件编码器和软件译码器的接口。这个接口主要采用 MCS-51 单片机，通过对串行口的操作，从而完成编码器和译码器的会话。MCS-51 是在等到计算机查询时才使得编码器的时钟有效，从而控制编码器开始工作。编码结束后产生一个下降沿信号送至 51 的外部中断 0，这时 51 通过读 PORT1 和 PORT2 获得编码结果。在计算机的要求下，51 把编码结果送至串行口。编码器的时钟信号是通过 51 编程产生的。

在下一章中，我们就可以对已经出现在串行口的编码结果进行软件译码了。

第四章 循环码译码器设计

本系统的软件译码器设计为即可以与硬件编码器配合使用，也可以独立使用的形式。

在软件译码器的设计过程中主要考虑一下问题：

1. 读写串行口，实现译码器与编码器的会话；
2. 进行纠错译码；
3. 译码器界面的设计；
4. 译码器实验的可参与性与新颖性

译码器实现的思路：

为了使译码器有一个友好的界面，采用了高级语言 Delphi 来开发译码器界面。为了学生能够很好地参与纠错译码算法，采用了大多数学学生熟悉的 C 语言来编写译码算法，算法程序编写为一个动态连接文件，通过 Delphi 来调用它。在 Delphi 中完成串行口的操作，具体是通过调用一些组件来实现。在完成循环码的译码过程中，学生有很多的参与点，通过此试验可以灵活牢固地掌握循环码的纠错译码方法。

译码器的具体实现如下所述：

4．1 译码器界面的设计

1．译码器主界面

如下图 16 所示。

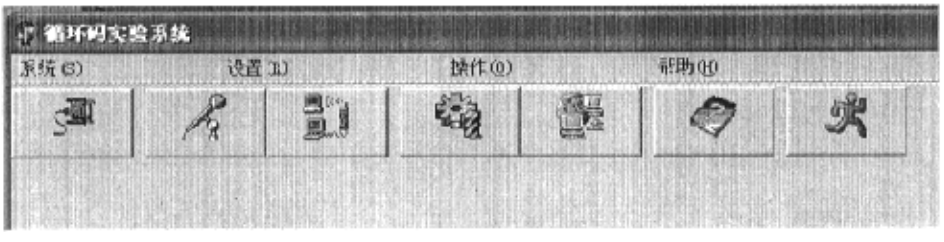


图 16 译码器主界面

主界面中各部分主要功能为：

- ✓ 系统：从这里可以推出译码器系统；
- ✓ 设置：用于设置串行口；
- ✓ 操作：从这里进入译码器，完成具体循环码的译码；

✓ 帮助：这里就使用本系统和学习循环码译码时可能遇到的问题给与了解答。

界面中的一些快捷按钮同样可以完成以上的操作。

2．串行口操作界面

串行口操作界面如下图 17 所示。

图 17 串行口操作界面

在这个界面要实现的主要功能有：

A. 初始化串行口

即对选择的串行口进行初始化操作。计算机的串行口在进行通信前，必须做串口初始化，即设置一些参数，包括：使用哪个串行口、通信速率、数据位数、奇偶校验情况、停止位数。而且这里的设置必须和 MCS-51 中的设置一致，否则不能正常通信。

在串口操作界面上，对于串口的初始化操作，只要求选择串口号，因为其他参数必须和 MCS-51 中的设置一致，所以是固定的。

B. 实验板查询

通过查询实验板，可以确定编码器是否已通过串行口连接到计算机。而编码器也是在译码器查询之后才开始进行编码。所以编码器和译码器要配合工作就必须查询实验板。

C. 接收数据

选择这项操作后，计算机就会要求译码器传送编码结果。对于译码器来说，就是完成对串口的写/读功能。这样译码器系统就有了编码的结

果，后面就可以对此数据进行译码。

3．译码器操作界面

译码时首先要选择循环码的类型，对应界面如下图 18 所示。选择了某种循环码后，系统进入译码界面，如下图 19 所示。

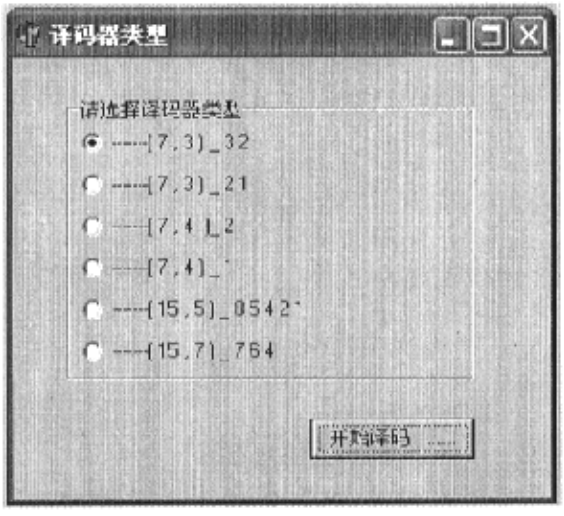


图 18 译码器类型

在译码器界面中实现循环码的译码。这里包括以下几部分：

1) 参数设置

译码器的原始数据可以是从串行口读入的，也可以是从本地输入的。这样设计带来的好处是：译码器既可与编码器配合使用，也可独立使用。

在这里可以考虑加入错码，从而形成传输线路上有干扰的现实情况。原始数据加上错码就是译码器的输入数据。

2) 计算伴随式

伴随式 $S = R \times H'$（R 为接收译码器的输入数据，H' 为监督矩阵）。在这里需输入 H'，然后系统自动计算相应的伴随式。

3) 查找错误图样

因为循环码的错误图样与伴随式式一一对应的，只要错码在循环码的纠错能力范围以内，都可以找到与伴随式对应的错误图样。这一步是系统自动完成。

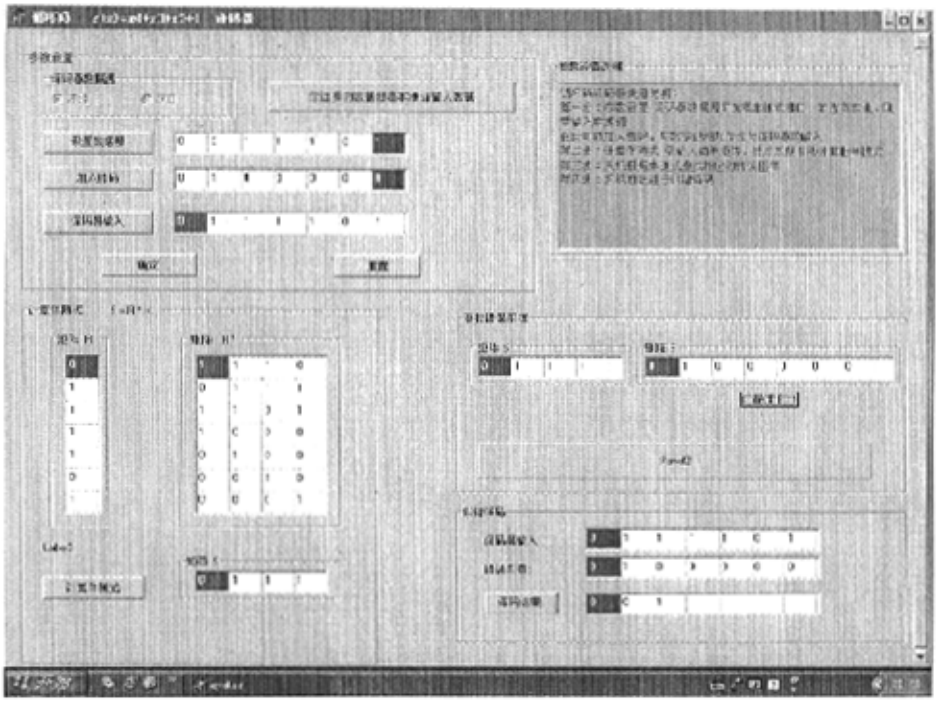


图 19 译码器界面

4) 纠错译码

译码器把数据数据和错误图样相加，得出纠错后的信息位。

4．2 串行口的操作

串行口的操作一是初始化串行口，二是完成对串行口的读写。在程序实现时，采用了 SPComm 控件实现对串行口的操作。

1．初始化串行口

初始化的参数包括：串行端口号、数据速率、数据位长度、奇偶校验情况、停止位情况。这里的设置应与编码器中的设置一致，否则串行口不能通信。

主要程序清单如下示：

```
begin
   with comm1 do
   if RadioGroup1.ItemIndex=0 then
   begin
   CommName:='COM1';
   end;
```

```
if radiogroup1.ItemIndex=1 then
begin
CommName:='COM2';
end;
BaudRate:=9600;
ByteSize:=_8;      //TByteSize
ParityCheck:=True;
Parity:=none;      //TParity = ( None, Odd, Even, Mark, Space )
StopBits:=_1;      //TStopBits = ( _1, _1_5, _2 )
StartComm;
Label2.Caption:='                    '+CommName+ '    已经初始化 !';

……

end;
```

2．读/写串行口

串行口读/写包括查询试验板和读编码器结果，它们都是以向串行口写数据开始，编码器接收到这个事先约定的数据后，向串行口串行相应的数据，对应的，在译码器系统中，当串行口上出现数据后，译码器就去执行一个接收串行口数据的过程，根据对接收到的数据的分析，译码器就可知道试验板是否已正常连接或者得到编码器的结果。

写串行口程序如下示：

```
if initialmode then
begin
sendCommand:=$01;
comm1.WriteCommData(@sendCommand,1);
 checkmode:=true;
end;
Label2.Caption:='          正在检查，请稍候. . . . . ';
Timer1.Interval:=5000;
Timer1.Enabled:=true;
……
end;
```

接收到编码器的编码数据后，把它进行了变换，使之成为译码器后需程序易于处理的形式。接收串行口数据的主要程序如下所示：

```
procedure Tcomini.Comm1ReceiveData(Sender: TObject; Buffer: Pointer;
  BufferLength: Word);
var    rbufb:byte;
       i,j,code1:integer;
       t,t1:integer;
       codechar1:string;
       comdata:intarray;
       PStr:PChar;
       Str:String;
begin
       for j:=1 to 2 do   comdata[j]:=0;
       for i:=1 to 2 do
       begin
            for j:=1 to 8 do
            begin
            re[i][j]:=0;
```

```
                rechar[i][j]:=' ';
                end;
            end;
           rbufb:=0;
           move(buffer^,rbufb,1);
           case rbufb of
                1:
                    begin
                    Timer1.Enabled:=false;
                    if checkmode then
                    begin
                    Label2.Caption:='      硬件连接正常 !';
                    checkbutton.Enabled:=true;
                    hardwarecheck:=true;
                    end;
                    end;
                2:
                    if receivemode then
                    begin
                    Timer2.Enabled:=false;
                    Label2.Caption:='  数据接收完毕 !';
                    receivecode:=true;
                    end;
              else
application.messagebox('硬件故障，请检查硬件连接，重新设置参数','注意
',mb_ok+mb_iconinformation);
           end;
      //接收到数据
           if rbufb=2 then
           begin
            Move(buffer^, PStr^, 3);
            Str:=PStr;
            for j:=1 to 2 do   comdata[j]:=integer(Str[j+1]);
            for i:=1 to 2 do
             begin
             re[i][1]:=comdata[i] div 128;
             t:=re[i][1]+48;
             rechar[i][1]:=chr(t);
             t1:=comdata[i]-re[i][1]*128;
             re[i][2]:=t1 div 64;
             t:=re[i][2]+48;
             rechar[i][2]:=chr(t);
             t1:=t1-re[i][2]*64;
             re[i][3]:=t1 div 32;
             t:=re[i][3]+48;
             rechar[i][3]:=chr(t);
             t1:=t1-re[i][3]*32;
             re[i][4]:=t1 div 16;
             t:=re[i][4]+48;
             rechar[i][4]:=chr(t);
             t1:=t1-re[i][4]*16;
             re[i][5]:=t1 div 8;
             t:=re[i][5]+48;
             rechar[i][5]:=chr(t);
             t1:=t1-re[i][5]*8;
             re[i][6]:=t1 div 4;
             t:=re[i][6]+48;
             rechar[i][6]:=chr(t);
             t1:=t1-re[i][6]*4;
             re[i][7]:=t1 div 2;
             t:=re[i][7]+48;
             rechar[i][7]:=chr(t);
             re[i][8]:=t1-re[i][7]*2;
             t:=re[i][8]+48;
```

```
            rechar[i][8]:=chr(t);
            end;

            for i:=1 to 8 do
            begin
            code[i]:=re[1][i];
            codechar[i]:=rechar[1][i];
            end;
            for i:=1 to 7 do
            begin
            code[i+8]:=re[2][i];
            codechar[i+8]:=rechar[2][i];
            end;
            code1:=code[1];
            codechar7[1]:=codechar[1];
            for i:=1 to 14 do
            begin
            code[i]:=code[i+1];
            code15[i]:=code[i];
            codechar[i]:=codechar[i+1];
            codechar15[i]:=codechar[i];
            end;
            code[15]:=code1;
            code15[15]:=code1;
            codechar[15]:=codechar7[1];
            codechar15[15]:=codechar7[1];

            closebutton.Enabled:=TRUE;
            quitinibutton.Enabled:=false;

        end;
    end;
```

4．3 译码的实现

在实现这部分功能的时候，一方面考虑用 C 语言程序实现循环码的译码，另一方面考虑在 Delphi 设计的译码器中如何调用这个算法。

4．3．1 动态连接文件的使用

译码算法采用 C 程序实现，对应一个动态链接文件，可供译码器调用。

译码算法在 C 语言的一个过程中实现，过程定义如下：

```
extern TESTDLL_API int nTestDll;
TESTDLL_API int fnTestDll(codetype,incode,outcode);
```

函数 fnTestDll 包含三个参数：

int codetype:输入参数，代表循环码的码型

char incode[15]：输入参数，代表接收到的循环码

char outcode[15]:输出参数，代表循环码的译码结果。

Delphi 中的调用情况：

译码器中过程定义语句：

```
function
fnTestDll(codetype:integer;rechar1:chararray;rechar2:chararray):integer;cdecl;ext
ernal 'TestDll.dll';
```

译码器中的过程调用语句：

```
returncode:=fnTestDll(codetype,rechar1,rechar2);
```

4．3．2 译码算法的实现

循环码的译码[1, 2, 3]分四步：

1. 根据错误图样 E 计算伴随式 S，得到伴随式矩阵；
2. 根据接收编码 R 再计算伴随式 S；
3. 在伴随式矩阵中查找与 R 对应的错误图样 E；
4. 纠正 R 中的错误，即 U = R + E 。

一、 循环码的纠检错能力

每一种循环码的错误图样反映了它的纠错能力，而纠错能力由以下几方面决定，设 n 为码长，k 为信息位长，e 为检错位数,t 为纠错位数。

✧ 汉明界限(Hamming bound) [1]

$$2^{n-k} >= [1 + C_n^1 + C_n^2 + \ldots\ldots + C_n^t]$$

给出了纠错能力的上限，2^{n-k} 表示码字最多可以纠正的错误图样数。

✧ 普洛特金界限（Plotkin bound）[1]

$$d_{\min} <= \frac{n * 2k - 1}{2k\text{-}1}$$

给出了纠错能力的另一上限。

✧ 码的最小距离[3]

$$d_{\min} >= e + t + 1$$

说明了纠错和检错位数的相互制约关系。

对于高码率编码，如果满足汉明界限，则一定满足普洛特金界限；对于低码率编码则不一定，所以应取其中较小者。

下面我们来分析常用的 6 种循环码的纠检错能力：

1．(7,4)_1- $g(x) = x^3 + x + 1$

(7,4)_2 - $g(x) = x^3 + x^2 + 1$

这种码的错误图样最多有 8 个，t = 1，$d_{\min}$ < = 3.7。所以它可以纠正所有小于等于 1 个的所有错误图样，对应错误图样有 8 个。纠检错能力如表 6 所示。

表 6（7，4）循环码纠检错能力表

检错(e)	纠错(t)
1	1
2	0

2. (7,3)_1 - $g(x) = x^4 + x^3 + x^2 + 1$
   (7,3)_2 - $g(x) = x^4 + x^2 + x + 1$

这种码的错误图样最多有 8 个，$t = 1$，$d_{min} <= 4$。所以它可以纠正所有小于等于 1 个的所有错误图样，对应错误图样有 8 个。纠检错能力如表 7 所示。

表 7（7，3）循环码纠检错能力表

检错(e)	纠错(t)
2	1
3	0

3. (15,7) - $g(x) = x^8 + x^7 + x^6 + x^4 + 1$

这种码的错误图样最多有 256 个，$t = 2$，$d_{min} <= 7.6$。所以它可以纠正所有小于等于 2 个的所有错误图样，对应错误图样有 121 个。纠检错能力如表 8 所示。

表 8（15，7）循环码纠检错能力表

检错(e)	纠错(t)
4	2
5	1
6	0

4. (15,5) - $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$

这种码的错误图样最多有 1024 个，$t = 3$，$d_{min} <= 7.7$。所以它可以纠正所有小于等于 3 个的所有错误图样，对应的错误图样有 576 个。纠检错能力如表 9 所示。

二、 错误图样矩阵与伴随式矩阵

循环码纠错的依据是：错误图样与伴随式是一一对应的。所以当我们计算出了接收码字的伴随式后，就可以得出相应的错误图样。错误图

样的计算方法是：

表 9 （15，5）循环码纠检错能力表

检错(*e*)	纠错(*t*)
3	3
4	2
5	1
6	0

$S=(ri+ej)\cdot H^T=ri\cdot H^T+ej\cdot H^T$

H^T：监督矩阵的转置，ri是一个码矢，所以$ri\cdot H^T=0$，因此

$S=(ri+ej)\cdot H^T=ej\cdot H^T$

三、常用循环码错误图样与伴随式小结

1. (7,4)_1- $g(x)=x^3+x+1$

序号	S	E
1	000	0000000
2	101	1000000
3	111	0100000
4	110	0010000
5	011	0001000
6	100	0000100
7	010	0000010
8	001	0000001

2. (7,4)_2 - $g(x)=x^3+x^2+1$

序号	S	E
1	000	0000000
2	110	1000000
3	011	0100000
4	111	0010000
5	101	0001000
6	100	0000100
7	010	0000010
8	001	0000001

3. (7,3)_1 - $g(x)=x^4+x^3+x^2+1$

序号	S	E
1	0000	0000000
2	1110	1000000
3	0111	0100000

4	1101	0010000
5	1000	0001000
6	0100	0000100
7	0010	0000010
8	0001	0000001

4. (7,3)_2 - $g(x) = x^4 + x^2 + x + 1$

序号	S	E
1	0000	0000000
2	1011	1000000
3	1110	0100000
4	0111	0010000
5	1000	0001000
6	0100	0000100
7	0010	0000010
8	0001	0000001

5. (15,7) - $g(x) = x^8 + x^7 + x^6 + x^4 + 1$

序号	S	E
1	00000000	000000000000000
2	00000001	000000000000001
3	00000010	000000000000010
4	00000100	000000000000100
5	00001000	000000000001000
6	00010000	000000000010000
7	00100000	000000000100000
8	01000000	000000001000000
9	10000000	000000010000000
10	11010001	000000100000000
11	01110011	000001000000000
12	11100110	000010000000000
13	00011101	000100000000000
14	00111010	001000000000000
15	01110100	010000000000000
16	11101000	100000000000000
17	00000011	000000000000011
18	00000101	000000000000101
19	00001001	000000000001001
20	00010001	000000000010001
21	00100001	000000000100001
22	01000001	000000001000001
23	10000001	000000010000001
24	11010000	000000100000001
25	01110010	000001000000001
26	11100111	000010000000001
27	00011100	000100000000001
28	00111011	001000000000001
29	01110101	010000000000001
30	11101001	100000000000001
31	00000110	000000000000110
32	00001010	000000000001010
33	00010010	000000000010010

34	00100010	000000000100010
35	01000010	000000001000010
36	10000010	000000010000010
37	11010011	000000100000010
38	01110001	000001000000010
39	11100100	000010000000010
40	00011111	000100000000010
41	00111000	001000000000010
42	01110110	010000000000010
43	11101010	100000000000010
44	00001100	000000000001100
45	00010100	000000000010100
46	00100100	000000000100100
47	01000100	000000001000100
48	10000100	000000010000100
49	11010101	000000100000100
50	01110111	000001000000100
51	11100010	000010000000100
52	00011001	000100000000100
53	00111110	001000000000100
54	01110000	010000000000100
55	11101100	100000000000100
56	00011000	000000000011000
57	00101000	000000000101000
58	01001000	000000001001000
59	10001000	000000010001000
60	11011001	000000100001000
61	01111011	000001000001000
62	11101110	000010000001000
63	00010101	000100000001000
64	00110010	001000000001000
65	01111100	010000000001000
66	11100000	100000000001000
67	00110000	000000000110000
68	01010000	000000001010000
69	10010000	000000010010000
70	11000001	000000100010000
71	01100011	000001000010000
72	11110110	000010000010000
73	00001101	000100000010000
74	00101010	001000000010000
75	01100100	010000000010000
76	11111000	100000000010000
77	01100000	000000001100000
78	10100000	000000010100000
79	11110001	000000100100000
80	01010011	000001000100000
81	11000110	000010000100000
82	00111101	000100000100000
83	00011010	001000000100000
84	01010100	010000000100000
85	11001000	100000000100000
86	11000000	000000011000000
87	10010001	000000101000000
88	00110011	000001001000000

89	10100110	000010001000000
90	01011101	000100001000000
91	01111010	001000001000000
92	00110100	010000001000000
93	10101000	100000001000000
94	01010001	000000110000000
95	11110011	000001010000000
96	01100110	000010010000000
97	10011101	000100010000000
98	10111010	001000010000000
99	11110100	010000010000000
100	01101000	100000010000000
101	10100010	000001100000000
102	00110111	000010100000000
103	11001100	000100100000000
104	11101011	001000100000000
105	10100101	010000100000000
106	00111001	100000100000000
107	10010101	000011000000000
108	01101110	000101000000000
109	01001001	001001000000000
110	00000111	010001000000000
111	10011011	100001000000000
112	11111011	000110000000000
113	11011100	001010000000000
114	10010010	010010000000000
115	00001110	100010000000000
116	00100111	001100000000000
117	01101001	010100000000000
118	11110101	100100000000000
119	01001110	011000000000000
120	11010010	101000000000000
121	10011100	110000000000000

6. (15,5) - $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$

行号	S	E
1	0000000000	000000000000000
2	0000000001	000000000000001
3	0000000010	000000000000010
4	0000000100	000000000000100
5	0000001000	000000000001000
6	0000010000	000000000010000
7	0000100000	000000000100000
8	*0001000000*	*000000001000000*
9	0010000000	000000010000000
10	0100000000	000000100000000
11	1000000000	000001000000000
12	0100110111	000010000000000
13	1001101110	000100000000000
14	0111101011	001000000000000
15	1111010110	010000000000000
16	1010011011	100000000000000
17	0000000011	000000000000011
18	0000000101	000000000000101

19	0000001001	000000000001001
20	0000010001	000000000010001
21	0000100001	000000000100001
22	0001000001	000000001000001
23	0010000001	000000010000001
24	0100000001	000000100000001
25	1000000001	000001000000001
26	0100110110	000010000000001
27	1001101111	000100000000001
28	0111101010	001000000000001
29	1111010111	010000000000001
30	1010011010	100000000000001
31	0000000110	000000000000110
32	0000001010	000000000001010
33	0000010010	000000000010010
34	0000100010	000000000100010
35	0001000010	000000001000010
36	0010000010	000000010000010
37	0100000010	000000100000010
38	1000000010	000001000000010
39	0100110101	000010000000010
40	1001101100	000100000000010
41	0111101001	001000000000010
42	1111010100	010000000000010
43	1010011001	100000000000010
44	0000001100	000000000001100
45	0000010100	000000000010100
46	0000100100	000000000100100
47	0001000100	000000001000100
48	0010000100	000000010000100
49	0100000100	000000100000100
50	1000000100	000001000000100
51	0100110011	000010000000100
52	1001101010	000100000000100
53	0111101111	001000000000100
54	1111010010	010000000000100
55	1010011111	100000000000100
56	0000011000	000000000011000
57	0000101000	000000000101000
58	0001001000	000000001001000
59	0010001000	000000010001000
60	0100001000	000000100001000
61	1000001000	000001000001000
62	0100111111	000010000001000
63	1001100110	000100000001000
64	0111100011	001000000001000
65	1111011110	010000000001000
66	1010010011	100000000001000
67	0000110000	000000000110000
68	0001010000	000000001010000
69	0010010000	000000010010000
70	0100010000	000000100010000
71	1000010000	000001000010000
72	0100100111	000010000010000
73	1001111110	000100000010000

74	0111111011	001000000010000
75	1111000110	010000000010000
76	1010001011	100000000010000
77	0001100000	000000001100000
78	0010100000	000000010100000
79	0100100000	000000100100000
80	1000100000	000001000100000
81	0100010111	000010000100000
82	1001001110	000100000100000
83	0111001011	001000000100000
84	1111110110	010000000100000
85	1010111011	100000000100000
86	0011000000	000000011000000
87	0101000000	000000101000000
88	1001000000	000001001000000
89	0101110111	000010001000000
90	1000101110	000100001000000
91	0110101011	001000001000000
92	1110010110	010000001000000
93	1011011011	100000001000000
94	0110000000	000000110000000
95	1010000000	000001010000000
96	0110110111	000010010000000
97	1011101110	000100010000000
98	0101101011	001000010000000
99	1101010110	010000010000000
100	1000011011	100000010000000
101	1100000000	000001100000000
102	0000110111	000010100000000
103	1101101110	000100100000000
104	0011101011	001000100000000
105	1011010110	010000100000000
106	1110011011	100000100000000
107	1100110111	000011000000000
108	0001101110	000101000000000
109	1111101011	001001000000000
110	0111010110	010001000000000
111	0010011011	100001000000000
112	1101011001	000110000000000
113	0011011100	001010000000000
114	1011100001	010010000000000
115	1110101100	100010000000000
116	1110000101	001100000000000
117	0110111000	010100000000000
118	0011110101	100100000000000
119	1000111101	011000000000000
120	1101110000	101000000000000
121	0101001101	110000000000000
122	0010100110	111000000000000
123	0001010011	011100000000000
124	1010110010	001110000000000
125	0101011001	000111000000000
126	1000110111	000011100000000
127	1110000000	000001110000000
128	0111000000	000000111000000

129	0011100000	000000011100000
130	0001110000	000000001110000
131	0000111000	000000000111000
132	0000011100	000000000011100
133	0000001110	000000000001110
134	0000000111	000000000000111
135	1100100011	110100000000000
136	1100001010	011010000000000
137	0110000101	001101000000000
138	1001011001	000110100000000
139	1110110111	000011010000000
140	1101000000	000001101000000
141	0110100000	000000110100000
142	0011010000	000000011010000
143	0001101000	000000001101000
144	0000110100	000000000110100
145	0000011010	000000000011010
146	0000001101	000000000001101
147	0001111010	110010000000000
148	0000111101	011001000000000
149	1010000101	001100100000000
150	1111011001	000110010000000
151	1101110111	000011001000000
152	1100100000	000001100100000
153	0110010000	000000110010000
154	0011001000	000000011001000
155	0001100100	000000001100100
156	0000110010	000000000110010
157	0000011001	000000000011001
158	1101001101	110001000000000
159	1100111101	011000100000000
160	1100000101	001100010000000
161	1100011001	000110001000000
162	1100010111	000011000100000
163	1100010000	000001100010000
164	0110001000	000000110001000
165	0011000100	000000011000100
166	0001100010	000000001100010
167	0000110001	000000000110001
168	0001001101	110000100000000
169	1010111101	011000010000000
170	1111000101	001100001000000
171	1101111001	000110000100000
172	1100100111	000011000010000
173	1100001000	000001100001000
174	0110000100	000000110000100
175	0011000010	000000011000010
176	0001100001	000000001100001
177	0111001101	110000010000000
178	1001111101	011000001000000
179	1110100101	001100000100000
180	1101001001	000110000010000
181	1100111111	000011000001000
182	1100000100	000001100000100
183	0110000010	000000110000010

184	0011000001	000000011000001
185	0100001101	110000001000000
186	1000011101	011000000100000
187	1110010101	001100000010000
188	1101010001	000110000001000
189	1100110011	000011000000100
190	1100000010	000001100000010
191	0110000001	000000110000001
192	0101101101	110000000100000
193	1000101101	011000000010000
194	1110001101	001100000001000
195	1101011101	000110000000100
196	1100110101	000011000000010
197	1100000001	000001100000001
198	0101011101	110000000010000
199	1000110101	011000000001000
200	1110000001	001100000000100
201	1101011011	000110000000010
202	1100110110	000011000000001
203	0101000101	110000000001000
204	1000111001	011000000000100
205	1110000111	001100000000010
206	1101011000	000110000000001
207	0101001001	110000000000100
208	1000111111	011000000000010
209	1110000100	001100000000001
210	0101001111	110000000000010
211	1000111100	011000000000001
212	0101001100	110000000000001
213	0100011110	101100000000000
214	0010001111	010110000000000
215	1011011100	001011000000000
216	0101101110	000101100000000
217	0010110111	000010110000000
218	1011000000	000001011000000
219	0101100000	000000101100000
220	0010110000	000000010110000
221	0001011000	000000001011000
222	0000101100	000000000101100
223	0000010110	000000000010110
224	0000001011	000000000001011
225	0111000010	100110000000000
226	0011100001	010011000000000
227	1011101011	001001100000000
228	1111101110	000100110000000
229	0111110111	000010011000000
230	1001100000	000001001100000
231	0100110000	000000100110000
232	0010011000	000000010011000
233	0001001100	000000001001100
234	0000100110	000000000100110
235	0000010011	000000000010011
236	0110101100	100011000000000
237	0011010110	010001100000000
238	0001101011	001000110000000

239	1010101110	000100011000000
240	0101010111	000010001100000
241	1000110000	000001000110000
242	0100011000	000000100011000
243	0010001100	000000010001100
244	0001000110	000000001000110
245	0000100011	000000000100011
246	0110011011	100001100000000
247	1001010110	010000110000000
248	0100101011	001000011000000
249	1000001110	000100001100000
250	0100000111	000010000110000
251	1000011000	000001000011000
252	0100001100	000000100001100
253	0010000110	000000010000110
254	0001000011	000000001000011
255	1100011011	100000110000000
256	1100010110	010000011000000
257	0110001011	001000001100000
258	1001011110	000100000110000
259	0100101111	000010000011000
260	1000001100	000001000001100
261	0100000110	000000100000110
262	0010000011	000000010000011
263	1001011011	100000011000000
264	1110110110	010000001100000
265	0111011011	001000000110000
266	1001110110	000100000011000
267	0100111011	000010000001100
268	1000000110	000001000000110
269	0100000011	000000100000011
270	1011111011	100000001100000
271	1111100110	010000000110000
272	0111110011	001000000011000
273	1001100010	000100000001100
274	0100110001	000010000000110
275	1000000011	000001000000011
276	1010101011	100000000110000
277	1111001110	010000000011000
278	0111100111	001000000001100
279	1001101000	000100000000110
280	0100110100	000010000000011
281	1010000011	100000000011000
282	1111011010	010000000001100
283	0111101101	001000000000110
284	1001101101	000100000000011
285	1010010111	100000000001100
286	1111010000	010000000000110
287	0111101000	001000000000011
288	1010011101	100000000000110
289	1111010101	010000000000011
290	1010011000	100000000000011
291	1001000111	101010000000000
292	1110111000	010101000000000
293	0111011100	001010100000000

294	0011101110	000101010000000
295	0001110111	000010101000000
296	1010100000	000001010100000
297	0101010000	000000101010000
298	0010101000	000000010101000
299	0001010100	000000001010100
300	0000101010	000000000101010
301	0000010101	000000000010101
302	0101110000	101001000000000
303	0010111000	010100100000000
304	0001011100	001010010000000
305	0000101110	000101001000000
306	0000010111	000010100100000
307	1010010000	000001010010000
308	0101001000	000000101001000
309	0010100100	000000010100100
310	0001010010	000000001010010
311	0000101001	000000000101001
312	1001110000	101000100000000
313	0100111000	010100010000000
314	0010011100	001010001000000
315	0001001110	000101000100000
316	0000100111	000010100010000
317	1010001000	000001010001000
318	0101000100	000000101000100
319	0010100010	000000010100010
320	0001010001	000000001010001
321	1111110000	101000010000000
322	0111111000	010100001000000
323	0011111100	001010000100000
324	0001111110	000101000010000
325	0000111111	000010100001000
326	1010000100	000001010000100
327	0101000010	000000101000010
328	0010100001	000000010100001
329	1100110000	101000001000000
330	0110011000	010100000100000
331	0011001100	001010000010000
332	0001100110	000101000001000
333	0000110011	000010100000100
334	1010000010	000001010000010
335	0101000001	000000101000001
336	1101010000	101000000100000
337	0110101000	010100000010000
338	0011010100	001010000001000
339	0001101010	000101000000100
340	0000110101	000010100000010
341	1010000001	000001010000001
342	1101100000	101000000010000
343	0110110000	010100000001000
344	0011011000	001010000000100
345	0001101100	000101000000010
346	0000110110	000010100000001
347	1101111000	101000000001000
348	0110111100	010100000000100

349	0011011110	001010000000010
350	0001101111	000101000000001
351	1101110100	101000000000100
352	0110111010	010100000000010
353	0011011101	001010000000001
354	1101110010	101000000000010
355	0110111001	010100000000001
356	1101110001	101000000000001
357	1011110101	100101000000000
358	1111100001	010010100000000
359	1101101011	001001010000000
360	1100101110	000100101000000
361	0110010111	000010010100000
362	1001010000	000001001010000
363	0100101000	000000100101000
364	0010010100	000000010010100
365	0001001010	000000001001010
366	0000100101	000000000100101
367	1010101100	100010100000000
368	0101010110	010001010000000
369	0010101011	001000101000000
370	1011001110	000100010100000
371	0101100111	000010001010000
372	1000101000	000001000101000
373	0100010100	000000100010100
374	0010001010	000000010001010
375	0001000101	000000001000101
376	0000011011	100001010000000
377	1010010110	010000101000000
378	0101001011	001000010100000
379	1000111110	000100001010000
380	0100011111	000010000101000
381	1000010100	000001000010100
382	0100001010	000000100001010
383	0010000101	000000010000101
384	1111011011	100000101000000
385	1101110110	010000010100000
386	0110111011	001000001010000
387	1001000110	000100000101000
388	0100100011	000010000010100
389	1000001010	000001000001010
390	0100000101	000000100000101
391	1000111011	100000010100000
392	1110000110	010000001010000
393	0111000011	001000000101000
394	1001111010	000100000010100
395	0100111101	000010000001010
396	1000000101	000001000000101
397	1011001011	100000001010000
398	1111111110	010000000101000
399	0111111111	001000000010100
400	1001100100	000100000001010
401	0100110010	000010000000101
402	1010110011	100000000101000
403	1111000010	010000000010100

404	0111100001	001000000001010
405	1001101011	000100000000101
406	1010001111	100000000010100
407	1111011100	010000000001010
408	0111101110	001000000000101
409	1010010001	100000000001010
410	1111010011	010000000000101
411	1010011110	100000000000101
412	0111110101	100100100000000
413	1001100001	010010010000000
414	1110101011	001001001000000
415	1101001110	000100100100000
416	0110100111	000010010010000
417	1001001000	000001001001000
418	0100100100	000000100100100
419	0010010010	000000010010010
420	0001001001	000000001001001
421	0001110101	100100010000000
422	1010100001	010010001000000
423	1111001011	001001000100000
424	1101111110	000100100010000
425	0110111111	000010010001000
426	1001000100	000001001000100
427	0100100010	000000100100010
428	0010010001	000000010010001
429	0010110101	100100001000000
430	1011000001	010010000100000
431	1111111011	001001000010000
432	1101100110	000100100001000
433	0110110011	000010010000100
434	1001000010	000001001000010
435	0100100001	000000100100001
436	0011010101	100100000100000
437	1011110001	010010000010000
438	1111100011	001001000001000
439	1101101010	000100100000100
440	0110110101	000010010000010
441	1001000001	000001001000001
442	0011100101	100100000010000
443	1011101001	010010000001000
444	1111101111	001001000000100
445	1101101100	000100100000010
446	0110110110	000010010000001
447	0011111101	100100000001000
448	1011100101	010010000000100
449	1111101001	001001000000010
450	1101101111	000100100000001
451	0011110001	100100000000100
452	1011100011	010010000000010
453	1111101010	001001000000001
454	0011110111	100100000000010
455	1011100000	010010000000001
456	0011110100	100100000000001
457	1100101100	100010010000000
458	0110010110	010001001000000

459	0011001011	001000100100000
460	1011111110	000100010010000
461	0101111111	000010001001000
462	1000100100	000001000100100
463	0100010010	000000100010010
464	0010001001	000000010001001
465	1111101100	100010001000000
466	0111110110	010001000100000
467	0011111011	001000100010000
468	1011100110	000100010001000
469	0101110011	000010001000100
470	1000100010	000001000100010
471	0100010001	000000100010001
472	1110001100	100010000100000
473	0111000110	010001000010000
474	0011100011	001000100001000
475	1011101010	000100010000100
476	0101110101	000010001000010
477	1000100001	000001000100001
478	1110111100	100010000010000
479	0111011110	010001000001000
480	0011101111	001000100000100
481	1011101100	000100010000010
482	0101110110	000010001000001
483	1110100100	100010000001000
484	0111010010	010001000000100
485	0011101001	001000100000010
486	1011101111	000100010000001
487	1110101000	100010000000100
488	0111010100	010001000000010
489	0011101010	001000100000001
490	1110101110	100010000000010
491	0111010111	010001000000001
492	1110101101	100010000000001
493	0011011011	100001001000000
494	1011110110	010000100100000
495	0101111011	001000010010000
496	1000100110	000100001001000
497	0100010011	000010000100100
498	1000010010	000001000010010
499	0100001001	000000100001001
500	0010111011	100001000100000
501	1011000110	010000100010000
502	0101100011	001000010001000
503	1000101010	000100001000100
504	0100010101	000010000100010
505	1000010001	000001000010001
506	0010001011	100001000010000
507	1011011110	010000100001000
508	0101101111	001000010000100
509	1000101100	000100001000010
510	0100010110	000010000100001
511	0010010011	100001000001000
512	1011010010	010000100000100
513	0101101001	001000010000010

514	1000101111	000100001000001
515	0010011111	100001000000100
516	1011010100	010000100000010
517	0101101010	001000010000001
518	0010011001	100001000000010
519	1011010111	010000100000001
520	0010011010	100001000000001
521	1110111011	100000100100000
522	1101000110	010000010010000
523	0110100011	001000001001000
524	1001001010	000100000100100
525	0100100101	000010000010010
526	1000001001	000001000001001
527	1110001011	100000100010000
528	1101011110	010000010001000
529	0110101111	001000001000100
530	1001001100	000100000100010
531	0100100110	000010000010001
532	1110010011	100000100001000
533	1101010010	010000010000100
534	0110101001	001000001000010
535	1001001111	000100000100001
536	1110011111	100000100000100
537	1101010100	010000010000010
538	0110101010	001000001000001
539	1110011001	100000100000010
540	1101010111	010000010000001
541	1110011010	100000100000001
542	1000001011	100000010010000
543	1110011110	010000001001000
544	0111001111	001000000100100
545	1001111100	000100000010010
546	0100111110	000010000001001
547	1000010011	100000010001000
548	1110010010	010000001000100
549	0111001001	001000000100010
550	1001111111	000100000010001
551	1000011111	100000010000100
552	1110010100	010000001000010
553	0111001010	001000000100001
554	1000011001	100000010000010
555	1110010111	010000001000001
556	1000011010	100000010000001
557	1011010011	100000001001000
558	1111110010	010000000100100
559	0111111001	001000000010010
560	1001100111	000100000001001
561	1011011111	100000001000100
562	1111110100	010000000100010
563	0111111010	001000000010001
564	1011011001	100000001000010
565	1111110111	010000000100001
566	1011011010	100000001000001
567	1010111111	100000000100100
568	1111000100	010000000010010

569	0111100010	001000000001001
570	1010111001	100000000100010
571	1111000111	010000000010001
572	1010111010	100000000100001
573	1010001001	100000000010010
574	1111011111	010000000001001
575	1010001010	100000000010001
576	1010010010	100000000001001

四、 监督矩阵

根据伴随式的计算可知，在译码时需要监督矩阵，我们可以由循环码的生成多项式计算出它的监督矩阵。

常用循环码的监督矩阵如下：

1. (7,4)_1- $g(x) = x^3 + x + 1$

$$H = \begin{matrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}$$

2. (7,4)_2 - $g(x) = x^3 + x^2 + 1$

$$H = \begin{matrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{matrix}$$

3. (7,3)_1 - $g(x) = x^4 + x^3 + x^2 + 1$

$$H = \begin{matrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{matrix}$$

4. (7,3)_2 - $g(x) = x^4 + x^2 + x + 1$

$$H = \begin{matrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{matrix}$$

5. (15,7) - $g(x) = x^8 + x^7 + x^6 + x^4 + 1$

$$H = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

6. (15,5) - $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$

$$\begin{matrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$H = \begin{matrix} 0&1&1&0&1&0&1&0&0&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&1&1&1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&1&1&1&0&0&0&0&1&0&0&0&0&0 \\ 1&1&0&0&1&0&0&0&0&0&1&0&0&0&0 \\ 1&0&1&1&0&0&0&0&0&0&0&1&0&0&0 \\ 0&1&0&1&1&0&0&0&0&0&0&0&1&0&0 \\ 1&1&1&1&1&0&0&0&0&0&0&0&0&1&0 \\ 1&0&1&0&1&0&0&0&0&0&0&0&0&0&1 \end{matrix}$$

五、 译码流程

如果只要求循环码完成检错功能，则只要计算出接收码字对应的伴随式，就可进行鉴别。如果伴随式为 0，则说明接收码无误，即对应错误图样为 0，否则有误。如果要求完成纠错功能，则当伴随式不为 0 时，搜索伴随式矩阵中有无这个伴随试，如果没有，则说明发生的错误超出了这个循环码的纠错能力范围，否则找到这个伴随式对应的错误图样 e，进行纠错译码。

译码程序流程如下图 20 所示。

译码主要程序清单：

```
#include "stdafx.h"
#include "TestDll.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
TESTDLL_API int nTestDll=0;
TESTDLL_API int fnTestDll( codetype, incode, outcode)
int codetype;
char incode[15];
char outcode[15];
{
int select;
int n,k,r,nn,i,j,t1,t2;
int index;
char s2[10];
char s[575][10];
int a=0,ctype=0;
char rr[15];
char hh[15][10],h[10][15],e[575][15],s1[10];
char e_file[30],h_file[30],s_file[30];
FILE *input_h;
FILE *input_s;
FILE *input_e;
char e_file1[30]="(7,4)_310_e.txt";
char h_file1[30]="(7,4)_310_h.txt";
char s_file1[30]="(7,4)_310_s.txt";
char e_file2[30]="(7,4)_320_e.txt";
char h_file2[30]="(7,4)_320_h.txt";
char s_file2[30]="(7,4)_320_s.txt";
char e_file3[30]="(7,3)_4320_e.txt";
char h_file3[30]="(7,3)_4320_h.txt";
char s_file3[30]="(7,3)_4320_s.txt";
char e_file4[30]="(7,3)_4210_e.txt";
```

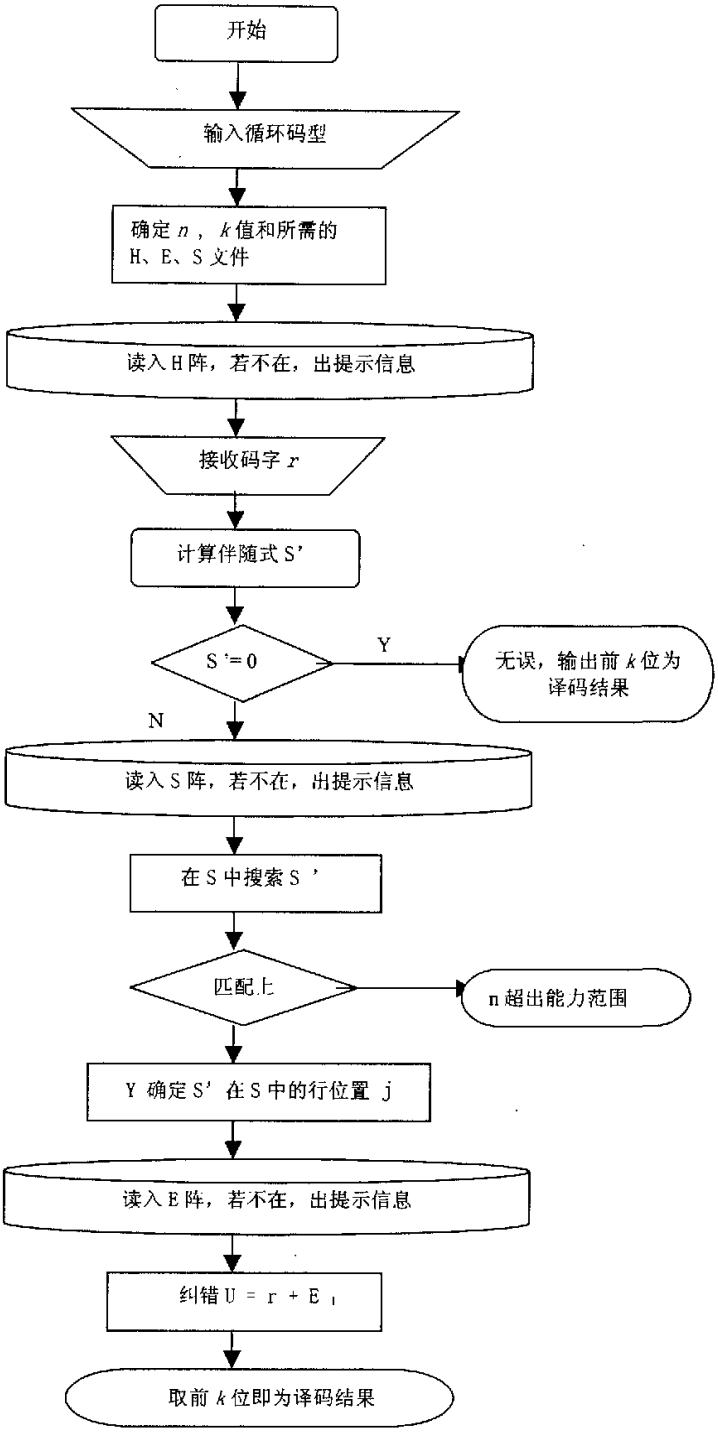

开始
输入循环码型
确定 n ， k 值和所需的 H、E、S 文件
读入 H 阵，若不在，出提示信息
接收码字 r
计算伴随式 S'
S '= 0
Y
无误，输出前 k 位为译码结果
N
读入 S 阵，若不在，出提示信息
在 S 中搜索 S '
匹配上
n 超出能力范围
Y 确定 S' 在 S 中的行位置 j
读入 E 阵，若不在，出提示信息
纠错 U = r + E j
取前 k 位即为译码结果

图 20 译码程序流程

```
char h_file4[30]="(7,3)_4210_h.txt";
char s_file4[30]="(7,3)_4210_s.txt";
char e_file5[30]="(15,7)_87640_e.txt";
char h_file5[30]="(15,7)_87640_h.txt";
char s_file5[30]="(15,7)_87640_s.txt";
char e_file6[30]="(15,5)_10854210_e.txt";
char h_file6[30]="(15,5)_10854210_h.txt";
char s_file6[30]="(15,5)_10854210_s.txt";
for (i=0;i<15;i++);
{outcode[i]=' ';}
select=codetype;
switch(select)
{
case 1:
        n=7;k=4;r=3;nn=7;
        memcpy(e_file,e_file1,30);
        memcpy(h_file,h_file1,30);
        memcpy(s_file,s_file1,30);break;
case 2:
   n=7;k=4;r=3;nn=7;
        memcpy(e_file,e_file2,30);
        memcpy(h_file,h_file2,30);
        memcpy(s_file,s_file2,30);break;
case 3:
        n=7;k=3;r=4;nn=7;
        memcpy(e_file,e_file3,30);
        memcpy(h_file,h_file3,30);
        memcpy(s_file,s_file3,30);break;
case 4:
        n=7;k=3;r=4;nn=7;
        memcpy(e_file,e_file4,30);
        memcpy(h_file,h_file4,30);
        memcpy(s_file,s_file4,30);break;
case 5:
        n=15;k=7;r=8;nn=15+105;
        memcpy(e_file,e_file5,30);
        memcpy(h_file,h_file5,30);
        memcpy(s_file,s_file5,30);break;
case 6:
        n=15;k=5;r=10;nn=15+105+455;
        memcpy(e_file,e_file6,30);
        memcpy(h_file,h_file6,30);
        memcpy(s_file,s_file6,30);break;

default:;
}
if((input_h=fopen(h_file,"r "))==NULL)          //input h
{ return 1;}
for(i=0;i<r;i++)
{
        for(j=0;j<n;j++)
        {
                h[i][j]=fgetc(input_h);
                h[i][j]=h[i][j]-48;
        }
        a=fgetc(input_h);
}
fclose(input_h);
for(i=0;i<r;i++)          //getout hh[n][r]
{
        for(j=0;j<n;j++)
                {       hh[j][i]=h[i][j];       }
}
for (i=0;i<n;i++)         //input receryed data rr[n]
{
```

```
        rr[i]=incode[i];
        rr[i]=rr[i]-48;
}
        for(i=0;i<r;i++)        //getout s[n][r] h'-列
        {
                s1[i]=0;
                for(j=0;j<n;j++)        //h'-行
                {       s1[i]=(s1[i]+(rr[j]&&hh[j][i]))%2;      }
        }
for (i=0;i<r;i++)       //right or error
{       if (s1[i]!=0) break; }
if (i==r)
{
        for (i=0;i<k;i++)
        {       outcode[i]=incode[i];}
        return 0;
}
if((input_s=fopen(s_file,"r"))==NULL) //input s[n][r]
{       return 2;       }
for(i=0;i<nn;i++)
{
        for(j=0;j<r;j++)
        {
                s[i][j]=fgetc(input_h);
                s[i][j]=s[i][j]-48;
        }
        a=fgetc(input_h);
}
fclose(input_h);
for (i=0;i<nn;i++)      //search out the match s[n][r]
{
        for (j=0;j<r;j++)
        {       s2[j]=(s1[j]==s[i][j]);  }
        t2=1;
        for (t1=0;t1<r;t1++)
        {       t2=t2&&(s2[t1]);        }
        if (t2==1)
        {       break;          }
}
index=i;
if (i>=nn)
{       return 4;       }
if((input_e=fopen(e_file,"r"))==NULL)        //       open e_file
{       return 3;       }
for(i=0;i<nn;i++)
{
        for(j=0;j<n;j++)
        {
                e[i][j]=fgetc(input_h);
                e[i][j]=e[i][j]-48;
        }
        a=fgetc(input_e);
}
fclose(input_e);
for (i=0;i<k;i++)       //corrected      output the decode
{       outcode[i]=(rr[i]+e[index][i])%2+48;  }
        return 0;
}
```

4．4 本章小结

本章介绍了循环码软件译码器的设计实现，主要说明了译码器操作界面的设计、串行口的操作和译码算法的实现。本系统可与编码系统结

合使用，也可独立使用。系统的操作界面采用 Delphi 语言开发，主要界面有译码器主界面、串行口操作界面、译码器界面、帮助界面。串行口的操作是采用了 Delphi 语言的一个控件 SPComm 来实现的，包括串行口的初始化及串行口的读/写操作。译码算法的实现是采用 C 语言编写的一个动态链接文件来实现，译码系统实现的是纠错译码，可对常用的 6 种循环码进行译码。此系统也可作为一个开发平台，在此基础之上学生可以对其他的循环码进行译码。使用本系统时，学生通过准备监督矩阵 H、错误图样矩阵 E，伴随式矩阵 S，以及填写部分 C 语言程序来完成译码。

通过本系统可使学生掌握循环码的译码方法，同时对循环码的纠检错能力有一个清晰的认识 。

第五章 系统改进方案研究

通过前面的叙述，可以看到这个循环码试验系统可以很好地实现硬件编码器和软件译码器的功能，而且两部分均有很多参与点，这些参与点涉及到循环码编码和译码的每个关键理论。编码器部分硬件电路布局合理美观，每个模块功能清晰，通过设置一些拨码开关即可得到不同类型的循环码编码器，而且输入输出数据都能够很直观地进行观察。译码器部分，操作界面美观，操作简单，只要准备相应的矩阵数据及填写部分C语言程序，就可实现不同类型循环码的译码。

这样的实验系统，可以满足我们日常实验的要求。

但是，上述系统还是有一些可以改进的地方，比如编码器原来用硬件实现，译码器用软件实现，可以改进为，编码器和译码器均可用软件或硬件来实现，做实验时学生可以自由选择，这样，实验的形式会更加新颖。

下面详细说明改进方案。

5．1 译码器改进方案

如果我们要求译码部分只具有检错功能，则编码电路的主要部分可以与译码电路共用。

达到检错目的的译码原理是：任一码组多项式都能被生成多项式整除，所以在译码电路中，我们就只对余数做判断，如果为 0，就认为无误码；如果非 0，则认为有误码。

编码电路功能流程如下图 21 所示。

对于译码电路来说，流程图中第 1、2、3 步都是需要的，只不过编码时输入数据位长为 k，译码时输入数据位长为 n。除法电路反馈信号存在的时间应该是 n 位长，所以对应计数器编码时设置为 k 值，译码时设置为 n 值。除法电路的结果在译码时我们就只关心的“监督位”--余式。译码时多一个判断电路，判断余式是否为 0。若为 0，就给发端一个证实信号，说明接收的码组无误；若为 1，就给发端一个否定信号，要求发端重发刚才的码组。

5．2 编码器改进方案

在编码部分改进方案中讲到，硬件系统编码和译码共用，那么软件

系统也可以做成编译码共用。

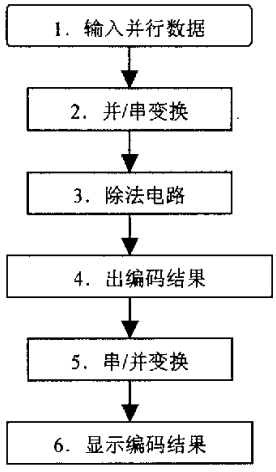


图 21 编码电路功能流程图

循环码的软件编码原理如下述：

循环码的软件编码实现原理我们可以用矩阵形式表示：

$$U = m\ G$$

U：码字；

m：k 比特的信息位序列；

G：生成矩阵。

编码流程如下图 22 所示。

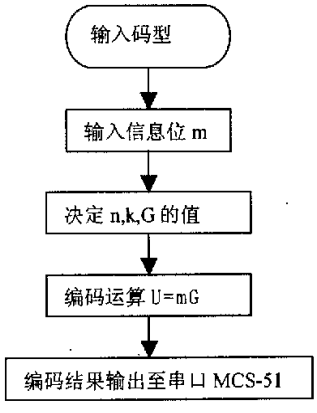


图 22 编码流程图

编码程序清单如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

```
int main(int argc, char* argv[])
{
int n,k,i,j;
int a,select=0;
char g[7][15],s1[15];
char g_file[30];
char g_file1[30]="(7,4)_310_g.txt";
char g_file2[30]="(7,4)_320_g.txt";
char g_file3[30]="(7,3)_4320_g.txt";
char g_file4[30]="(7,3)_4210_g.txt";
char g_file5[30]="(15,7)_87640_g.txt";
char g_file6[30]="(15,5)_10854210_g.txt";
printf("  crc style\r\n");
printf(" 1.(7,4)_310\r\n");
printf(" 2.(7,4)_320\r\n");
printf(" 3.(7,3)_4320\r\n");
printf(" 4.(7,3)_4210\r\n");
printf(" 5.(15,7)_87640\r\n");
printf(" 6.(15,5)_10854210\r\n");
printf(" please select :");
scanf("%d",&select);
switch(select)
{
case 1:
        n=7;k=4;
        memcpy(g_file,g_file1,30);break;
case 2:
   n=7;k=4;
        memcpy(g_file,g_file2,30);break;
case 3:
        n=7;k=3;
        memcpy(g_file,g_file3,30);break;
case 4:
        n=7;k=3;
        memcpy(g_file,g_file4,30);break;
case 5:
        n=15;k=7;
        memcpy(g_file,g_file5,30);break;
case 6:
        n=15;k=5;
        memcpy(g_file,g_file6,30);break;
default:printf("error\n");
        return 0;
}
printf("\r\nYour select is (%d,%d) crc\r\n",n,k);
printf("please input  info bit(%d bit):",k);
char rr[15];
scanf("%s",rr);
printf("\r\nr:    ");      //output rr[n]
for (i=0;i<k;i++)
{
        rr[i]=rr[i]-48;
        printf(" %d",rr[i]);
}
printf("\r\n");
FILE *input_g; //input h
if((input_g=fopen(g_file,"r"))==NULL)
{
        printf("cannot open file\n");
        return 0;
}
for(i=0;i<k;i++)
{
        for(j=0;j<n;j++)
        {
```

```
                g[i][j]=fgetc(input_g);
                g[i][j]=g[i][j]-48;
        }
        a=fgetc(input_g);
}
fclose(input_g);
for(i=0;i<k;i++)
{
        printf("\r\n");
        for(j=0;j<n;j++)
        {
                printf(" %d ",g[i][j]);
        }
}
        for(i=0;i<n;i++)
        {
                s1[i]=0;
                for(j=0;j<k;j++)
                {
                        s1[i]=(s1[i]+(rr[j]&&g[j][i]))%2;
                }
        }
return 0;
}
```

上面的程序段，依然编写为一个.dll 的文件，由 Delphi 来调用，参数有：

1. 输入参数：循环码型；
2. 输入参数：信息序列；
3. 输出参数：编码结果。

在 Delphi 编写的操作界面中增加“编码输出”选项，使编码可以输出到串口。

5．3 编码器与译码器接口的改进

如果译码用硬件实现，则在 MCS-51 程序中应增加读编码器结果的语句。因为前面在 51 的程序中以涉及到读串行口的操作，所以，这个改进是很容易的，就不再赘述。

5．4 本章小结

本章说明了循环码编译码系统的改进方案，针对于老系统编码器采用硬件实现，译码器采用软件实现，为了使系统更具有灵活性、形式更新颖，改进以后的循环码试验系统是编码器和译码器的软件/硬件形式可以自由选择。要达到这个效果，就相应的要在原系统基础之上做一些改动，包括硬件和软件。硬件的改动涉及到两个地方：一个是输入信息位长，一个是增加一个监督位为 0 检测电路。软件部分的修改是增加一个软件编码的.DLL 文件，同时在 Delphi 中增加输出编码器结果的功能。相应在硬件 MCS-51 的程序中也应稍做改进。

结 束 语

循环码是一种典型的差错控制编码，在通信类院校的《通信原理》课程中是必须掌握的一部分内容。为了更好地配合理论教学，深入理解此知识点、提高学生的操作能力和计算机水平，本人特开发了循环码实验系统，所做的主要工作如下：

1. 循环码编码器硬件平台的设计与实现

设计了一个通用的循环码编码器，系统中有许多参与点和观察点，涉及到编码器的每个关键知识点。逻辑器件主要采用 74 系列芯片，通过在电路中设置几组开关，使系统具有通用性，编码器的输入和输出数据均有良好的显示。

2. 循环码译码器软件平台的设计与实现

通过用 C/C++语言编写动态链接文件来实现译码算法，采用 Delphi 语言完成了译码器操作界面的设计并调用动态链接文件。译码器平台使用时，学生可以自己选择循环码类型，译码器的数据源可来自编码器亦可是在译码器中输入，译码涉及的几组关键数据均可观察到。译码算法程序需要学生提供相应的数据，这是理解循环码译码器的关键。

3. 编码器和译码器的接口实现

通过串行口通信来完成编码器和译码器的会话，本系统采用了 51 系列单片机来实现串行口通信，设计并实现了 51 单片机的硬件连接和软件编程。使译码器和编码器的通信得以很好的控制。

4. 对系统进行了改进研究

在系统的实现过程中已感觉到有可以改进的地方，改进后的系统可以做到：在硬件编码/软件译码和软件编码/硬件译码之间进行灵活选择，大大增加了系统的灵活性。目前已完成了系统的部分改进工作。

经编码器和译码器的良好工作情况证明，本系统可以较好地实现编译码功能。它也同时提供了一个循环码实验平台，通过循环码理论的学习，学生可在此平台上实现多种循环码的编码和译码，进一步深化对这一理论的理解和应用。

致　　谢

论文工作即将结束，研究生进修生活也很快就要画上一个句号了，在这几年中，我得到了很多来自老师、同学、家人和朋友的关心和帮助，在此，我要向他们表示我深深的谢意。

首先要特别感谢的是我的导师沈连丰教授，沈老师不但具有深厚的通信理论知识，更具有严谨求实的科学态度和优秀的人格魅力。沈老师在学业上给了我许多有益的指导，在学识、教学、为人等很多方面给我做出了很好的榜样，使我受益匪浅，这些优秀品质会一直指导我今后的工作和学习。

其次要特别感谢宋铁成老师，宋老师工作一丝不苟，在做课题期间给了我很大的帮助。

还要要特别感谢在同一个实验室的刘佳和施容两位同学，他们也给了我许多帮助。

另外还要感谢我的同窗赵新红、王丽敏、李玉良、李庞等同学，是他们的帮助和鼓励才使我能够顺利完成学业。

对于我至亲至爱的家人们，他们既是我的精神寄托也是我努力学习和工作的动力，感谢他们给了我无私的爱。

陈美娟

2004年9月于

东南大学移动通信国家重点实验室

参 考 文 献

[1]（美）Sklar,B.著，徐平平等译，数字通信-基础与应用（第二版）[M]，北京：电子工业出版社，2002.9

[2] (加) Haykin S.著，宋铁成等译，通信系统（第四版）[M]，北京：电子工业出版社，2003.10

[3] (美)Proakis,J.G.等著，叶芝慧等译，通信系统工程（第二版）[M]，北京：电子工业出版社，2002.7

[4] 樊昌信等编著，通信原理[M]，5版，北京：国防工业出版社，2001

[5] 曹志刚等，现代通信原理[M]，北京：清华大学出版社，2002.3

[6] 张培仁等编著，基于 C 语言编程 MCS-51 单片机原理与应用[M]，北京：清华大学出版社，2002

[7] 李现勇，Visual C++串口通信技术与工程实践[M]，北京：人民邮电出版社，2002.5

[8] 宋斌等编著，Visual C++6.0 教程[M]，北京：北京希望电子出版社，2000.4

[9] 谭浩强编著，C 程序设计[M]，北京：清华大学出版社，1991

[10] 陈惟斌等，Delphi4.0 应用开发指南[M]，北京：人民邮电出版社，1998.12

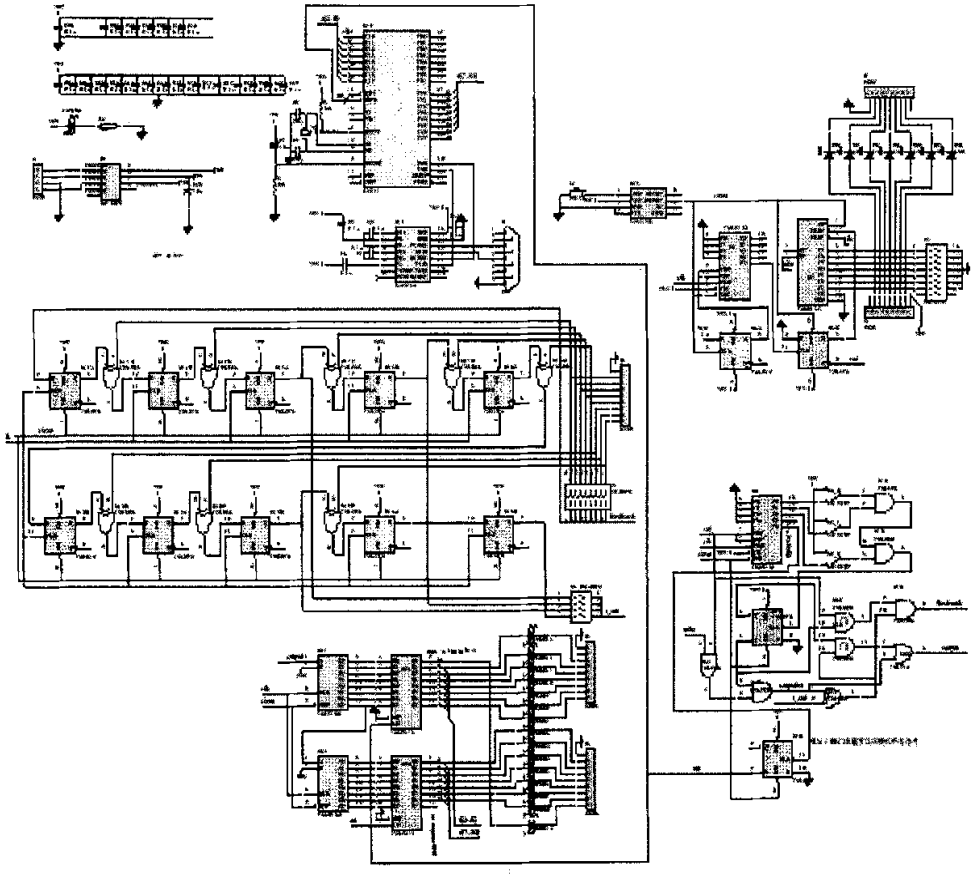
[11] 徐惠民等编著，数字电路与逻辑设计[M]，北京：人民邮电出版社，1992.1

[12] http：//www.xinqi.cn DB(DB/OL)

攻读硕士学位期间完成的论文

1. “智能网体系结构”，陈美娟 《现代通信》2002.12
2. “一种循环码实验系统的设计和研制”，陈美娟 沈连丰 《电气电子教学学报》 2004.10

附录 1：循环码编码器原理电路图



附录 2：循环码编码器印刷电路布局图