

前言

卷积码是由伊莱亚斯(Elias)于1954年首先提出来的。它充分利用了各组之间的相关性,本组的信息元不但决定本组的监督元,而且也参与决定以后若干组的监督元。同时在译码过程中,不但从该时刻所收到的码组中提取译码信息,而且还利用以后若干时刻内所收到的码组来提取有关信息。无论从理论上还是实际上均已证明其性能优于线性分组码。近年来众多有关卷积码研究结果表明,卷积码最有希望实现香农信道编码定理。但卷积码在译码理论及实际应用较为复杂,这些缺点限制了其进一步发展和应用。

维特比译码算法由维特比(Viterbi)1964年提出,算法实质是最大似然译码,但它利用了编码网格图的特殊结构,在网格图中选择一条路径,使相应的译码序列与接收到的序列之间的汉明距(即量度)最小的一种最大似然译码方法,从而大大降低了计算的复杂性。

目前,第三代移动通信系统(3G)在我国已经开始紧锣密鼓地实施,它带来的高速度、高品质的无线通信服务,将使我们领略到信息技术的无穷魅力。信道纠错编码技术作为保证信息可靠传输的技术,在3G各系统中广泛采用,并且我们现在所使用的第二代移动通信系统,如GSM、CDMA通信系统,还有卫星与空间通信系统广泛采用了卷积码信道编、译码技术。

本次设计将以(2, 1, 2)卷积码为例,通过单片机,实现卷积码的译码,借助RS232完成单片机与单片机、单片机与计算机的串口通信,借助7279完成键盘扫描和数码管显示。

第一章 系统组成及工作原理

本次设计的 (2, 1, 2) 卷积码的译码由 89C52 单片机为工具, 接收来自另一单片机或计算机的编码信息, 通过 7279 的键盘扫描和数码管显示功能可以方便地观察到单片机接收的信息和译码结果。其系统框图如下图 1-1 所示。

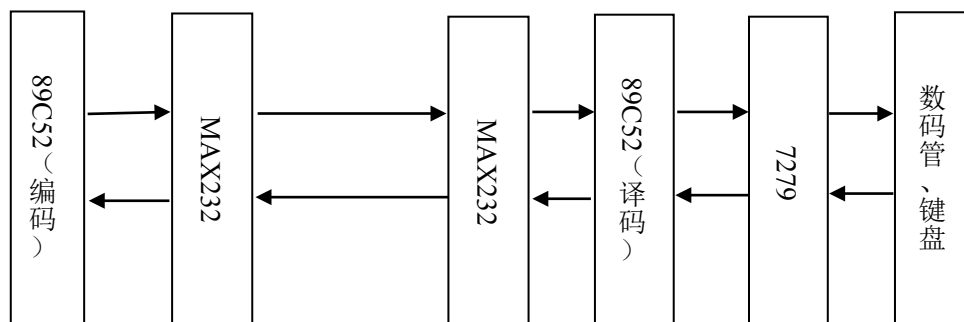


图 1-1 系统框图

MAX232 芯片能实现 EIA-RS-232C 的正负电压与 TTL 的高低电平之间的转换, 通过它能够方便实现单片机与单片机、单片机与计算机间的串口通信。

HD7279 是一片具有串行接口的, 可同时驱动 8 位共阴极数码管的智能显示驱动芯片, 该芯片同时还可连接多达 64 键的键盘矩阵, 单片即可独立完成显示、键盘接口的全部功能。采用 7279 芯片能够节省单片机资源, 降低编程复杂度。

1.1 总体系统分析

本系统的具体设计要求为: 系统框图

1. 用汇编语言进行 (2, 1, 2) 卷积码的译码, 并要求在数码管上显示输入的信息;
2. 用 RS232 串行芯片实现单片机与计算机的信息传输;
3. 用 RS232 串行芯片实现单片机与单片机的信息传输;
4. 在上位机上显示所传输的译码结果。

1.2 系统单元功能模块

本系统具有以下单元功能模块:

1. 单片机译码部分;
2. RS232 串口通信部分;
3. 数据显示部分。

第二章 软件设计

2.1 相关理论说明

2.1.1 卷积码的描述

卷积码是把 k 个信息比特的序列编成 n 个比特的码组，每个码组的 $n-k$ 个校验位与本码组的 k 个信息位有关，而与其他码组无关。为了达到一定的纠错能力和编码效率，分组码的长度一般都比较长。编译码时必须把整个信息码组存储起来，由此产生的译码延时随 n 的增加而增加。

卷积码是一个有限记忆系统，它也将信息序列分割成长度 k 的一个个分组，然后将 k 个信息比特编成 n 个比特，但 k 和 n 通常很小，特别适合以串行形式进行传输，时延小。与分组码不同的是在某一分组编码时，不仅参看本时刻的分组而且参看以前的 $N-1$ 个分组，编码过程中互相关联的码元个数为 nN 。 N 称为约束长度。常把卷积码写成 $(n, k, N-1)$ 卷积码。正因为卷积码在编码过程中，充分利用了各级之间的相关性，无论是从理论上还是实际上均已证明其性能要优于分组码。

2.1.2 卷积码的编码

本次设计以 $(2, 1, 2)$ 卷积码为例。图 2-1 为这种卷积编码器的结构，它的编码方法是：序列依次移入一个两级移位寄存器，编码器每输入一位信息 b_i ，输出端的开关就在 c_1 和 c_2 之间来回切换一次，输出为 $c_{1,i}$ 和 $c_{2,i}$ ，其中

$$c_{1,i} = b_i + b_{i-1} + b_{i-2}$$

$$c_{2,i} = b_i + b_{i-2}$$

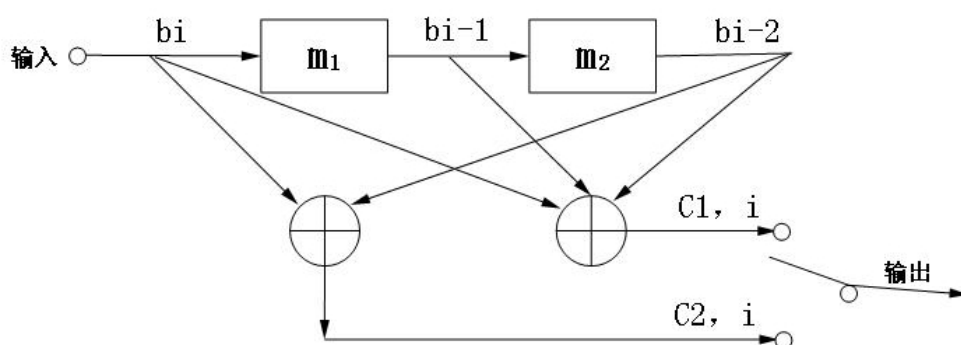


图 2-1 $(2, 1, 2)$ 卷积码编码器

设寄存器 m_1 、 m_2 的起始状态为零，则编码器的输入输出时序关系可用图 2-2 表示。

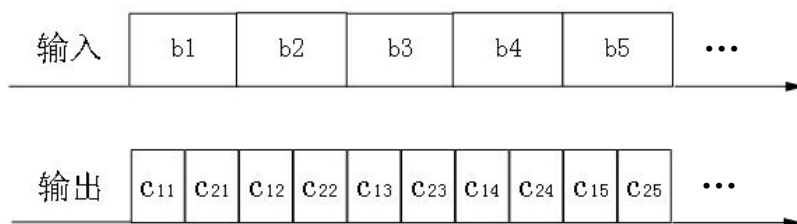


图 2-2 (2, 1, 2) 卷积码编码器的输入输出时序

要使最后 1 位输入同样影响 3 对输出，并且使编码器回到全零状态，还需使编码器多输出 2 对信息，为了做到这一点，需要增加 2 个时钟循环，并且在此期间保持输入为 0，这一过程叫做“点亮”编码器。如果不执行“点亮”操作，最后 2 位输入信息的纠错能力就会下降。

2.1.3 卷积码的图解表示

根据卷积码的特点，常采用图解表示法对其进行研究。主要的图解表示法有两种，即状态转移图、网格图。

因为卷积码的编码器的记忆性是有限的，所以可以使用状态转移图来表示其转移过程。在状态转移图中，卷积编码器的每一个状态对应于一个椭圆，状态的转移用两个椭圆间的有向线段表示，在线上标出状态转移的输入和对应的输出。图 3-1 中 (2, 1, 2) 卷积码的状态转移图如图 2-3 所示。

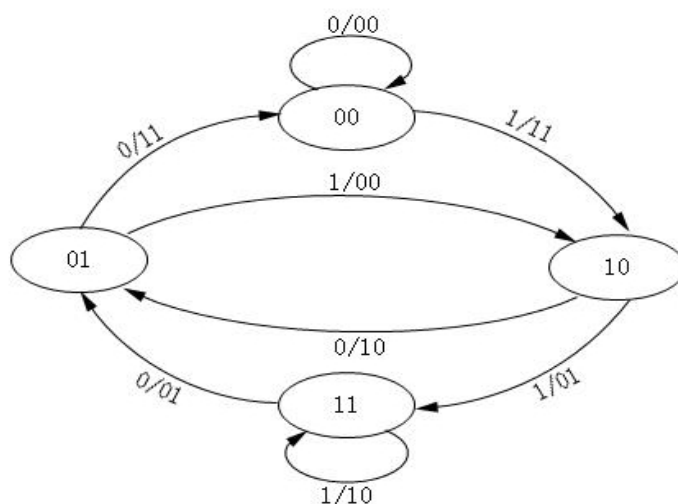


图 2-3 状态转移图

另一种更为常用的描述卷积码的方法是网格图。该图根据时间的推进来反映状态的转移。网格图上纵坐标表示状态，横坐标表示时间，每一次状态转移利用连接相邻时间点上两个状态的有向线段来表示。图 2-4 是图 2

—3 中编码器所对应的网格图，其中实线表示“0”，虚线表示“1”。

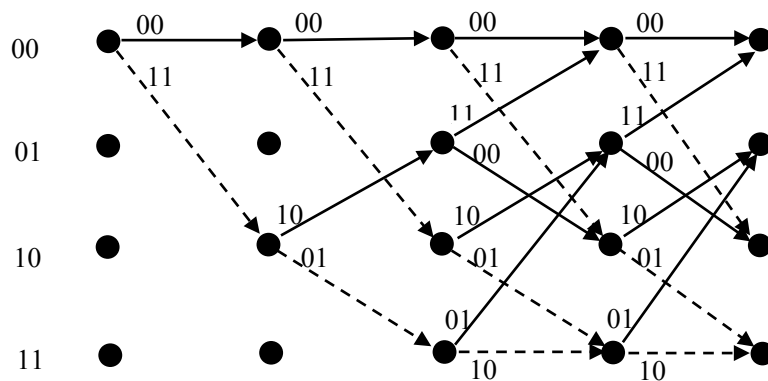


图 2-4 网格图

2.1.4 卷积码的译码

卷积码的译码方法可分为两大类。一类是代数译码，利用编码本身的代数结构进行译码，不考虑信道的统计特性；另一类是概率译码，这种译码建立在最大准则的基础上，由于计算时用到了信道的统计特性，因而提高了译码性能，但这是以增加硬件复杂度为代价的，常用的概率译码方法是维特比译码。

维特比译码是基于最大似然准则的概率译码，它的基本思想是比较接收序列与所有可能的发送序列，从中选择与接收序列汉明距离最小的发送序列作为译码输出。可能的发送序列与接收序列的汉明距离称为量度。

维特比译码使用网格图描述卷积码，每个可能的发送序列都与网格图中的一条路径相对应，如果在某个节点上发现某条路径已不可能与接收序列具有最小距离，那么就放弃这条路径，这样一直进行到倒数第二级。由于这种方法较早地丢弃了那些不可能的路径，因而减轻了译码的工作量。

2.2 系统总流程

本系统的总流程图如图 2-5 所示。其主要功能模块有：系统初始化、数据接收、读数据、数据判断、显示、维特比译码。下面对各个部分进行说明。

1. 系统初始化

堆栈指针 SP 设为 67H，清 10H—7FH 内存单元，7279 初始化，设置定时器 1 工作于方式 2，波特率为 1200，启动定时器，串行口工作于方式 2 并允许串口接收数据。

2. 数据接收

当译码器接收到有效数据时，将接收到的 12 字节的数据存于存储器 16H—21H 中，并将其转换为适合维特比译码的 6 字节数据，存于存储器 10H—15H 单元中。

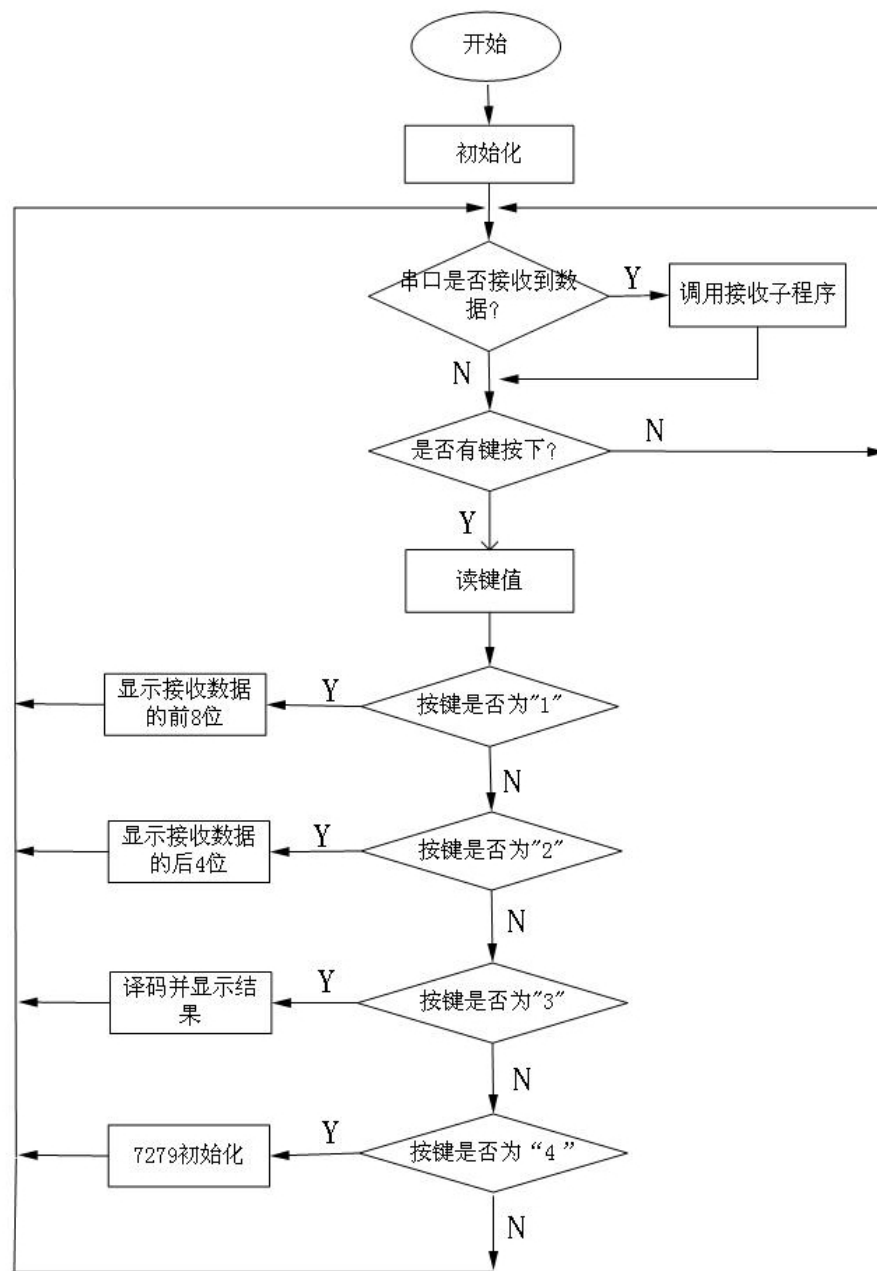


图 2—5 系统总流程图

3. 读数据

当有键按下时，读取键值，并存于 Acc 寄存器中。

4. 数据判断

判断按键号，并根据键号转向相应的处理程序。

5. 显示

将接收到的数据或译码结果直观的在数码管上显示出来。

6. 维特比译码

对接收到的数据进行维特比译码，并将译码结果存于存储器 58H—5DH 单元中。

2.3 初始化部分

系统初始化是为以后串口接收数据，7279 显示作做准备，其主要工作是将堆栈指针 SP 设为 67H，清 10H—7FH 内存单元，7279 初始化，设置定时器 1 工作于方式 2（自动恢复初始的 8 位计数器），波特率为 1200（不一定要 1200，但波特率过高，数据传输过程中可能出错），启动定时器，串行口工作于方式 2（9 位异步串行通信，一帧信息为 11 位：1 位起始位，8 位数据位，1 位附加数据位，1 位停止位）并允许串口接收数据。

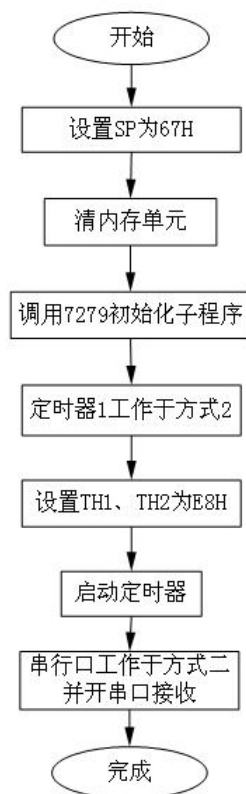


图 2—6 系统初始化流程图

2.4 数据接收部分

在主程序的循环中，每次都要判断是否从串中接收到有效数据（RI 是否为 1），当串中接收到有效数据时，将接收到的 12 字节的数据存于 16H—21H 中，并将其转换为适合维特比译码的 6 字节数据，存于 10H—15H 单元中。串口数据接收流程如下图 2—7 所示。

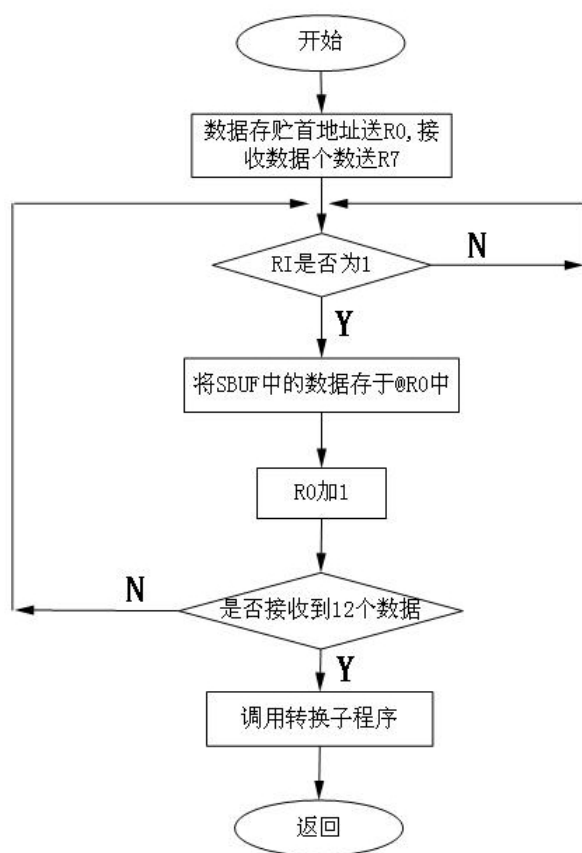


图 2-7 接收串口数据流程图

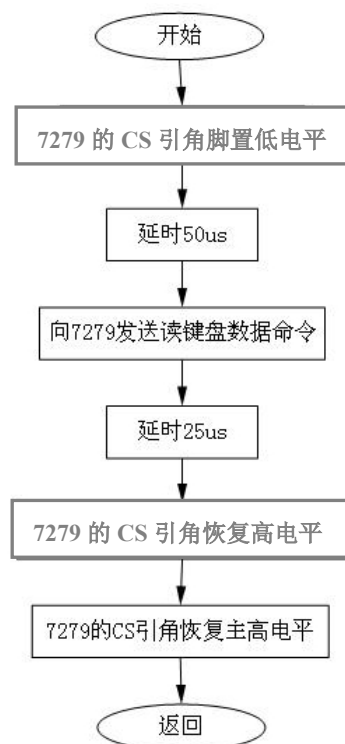


图 2-8 读键值流程图

2.5 读数据部分

主程序循环中每次都要判断是否有键按下，当有键按下时，调用读键值子程序，读出此按键的键号，并将键号存于 Acc 寄存器中，为后面的判断键号作准备。读键值流程图如图 2-8 所示。

2.6 数据判断部分

当有键按下时，判断按键号，并根据键号转向相应的处理：如果键号为“1”，数码管显示接收数据的前 8 位；如果键号为“2”，数码管显示接收数据的后 4 位；如果键号为“3”，则对接收到的数据进行译码并显示译码结果；如果键号为“4”，对 7279 进行初始化；如果是其它键号，则不做任何处理。其程序流程在主程序流程图 2-5 中已经体现出来。

2.7 显示部分

显示子程序能够将接收到的数据或译码结果直观的在数码管上显示出来。显示子程序有 3 个，分别为 DISP1、DISP2、DISP3，它们分别用于显示接收数据的前 8 位、接收数据的后 4 位、维特比译码结果。现以 DISP3 为例说明显示部分，其流程图如图 2—9 所示。

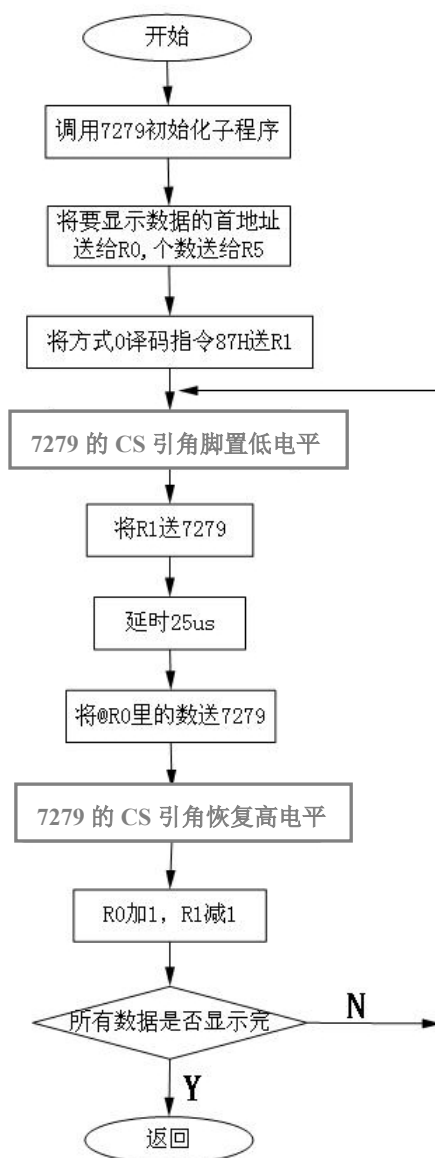


图 2—9 显示子程序流程图

2.8 维特比译码部分

维特比译码又可分为维特比译码主程序、求状态历史记录和累计误差子程序、求状态顺序子程序、求译码结果子程序。

1. 维特比译码主程序

当按键“3”按下时，对接收到的数据进行维特比译码，并将译码结果存于 58H—5DH 单元中。维特比译码程序流程图如图 2—10 所示。

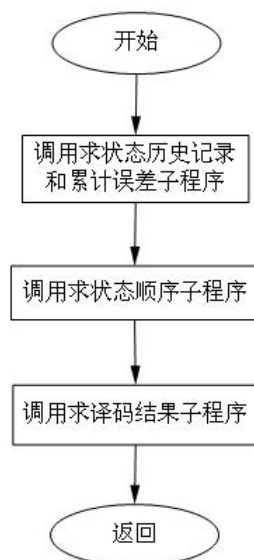


图 2—10 维特比译码流程图

2. 求状态历史记录和累计误差子程序

状态历史记录表能够记录下维特比译码过程中每一步译码后，状态的变化过程，其所占内存空间为 40H—57H。累计误差记录表能够记录上一次译码结束时的累计误差（可能的发送序列与接收序列的汉明距离）和本次译码结束后的累计误差，其中上一次的累计误差存于 30H—33H 单元中，本次结束后的累计误差存于 34H—37H 单元中。

本次设计所接收到的数据总共有 6 组，所以要将译码分为 6 步实现。已知起始状态为“0”（00），则下一个状态只可能是“0”（00）或“2”（10）（参考图 2—4），所以第一步译码只需计算两个累计误差，并将其存于 30H、32H 单元中即可，第二步译码需根据第一次译码的累计误差求出本次译码完后的累计误差，存于 34H—37H 单元中，用 34H—37H 中的内容覆盖 30H—33H 中的内容，作为下一步的起始累计误差，并将各个状态的前导状态分别存于 44H—47H 单元中，第三步的译码过程较第二步只需加一个累计误差的比较，淘汰较大的累计误差，以后的译码都是第三步译码的重复，所以从第三步开始可以用循环完成。

假如输入的序列为 010100，则接收到的序列（译码后的序列）为 001110001011，根据上述过程，可以得到状态历史记录表 2.1，其中带阴影的为幸存路径所经历的状态。

表 2.1 状态历史记录表

状态 \ t	1	2	3	4	5	6
00	00	00	00	01	00	01
01	00	02	02	03	02	00
10	00	00	00	01	00	00
11	00	02	02	03	00	00

求状态历史记录和误差流程图如图 2-11 所示，其流程图又可以分为三个部分，分别如图 2-12、2-13、2-14 所示。

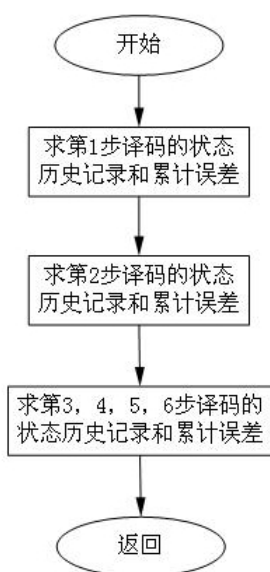


图 2-11 求状态历史记录和误差流程图

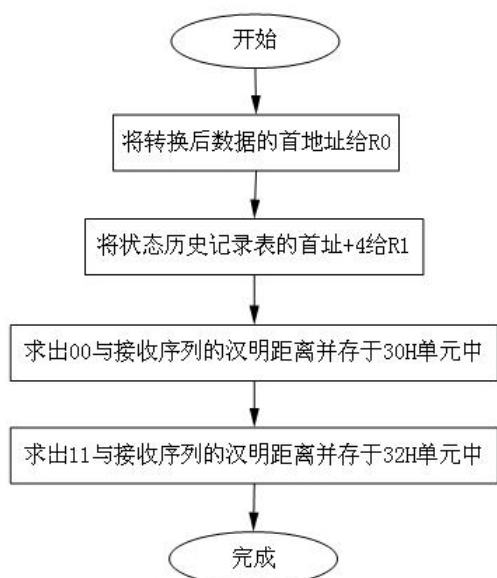


图 2-12 求第 1 步译码状态历史记录和误差流程图

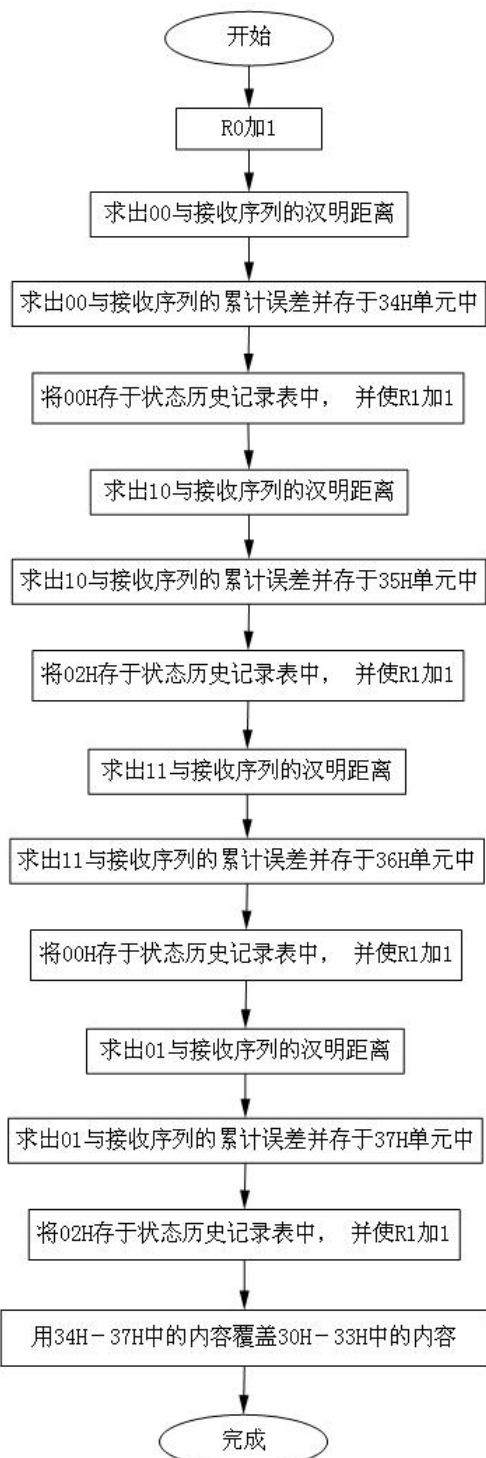


图 2-13 求第 2 步译码状态历史记录和误差流程图