# 摘 要

随着嵌入式系统和数据通信技术的发展，数据通信系统的实时性和可靠性要求也越来越高，因此在嵌入式数据通信系统中差错控制技术的性能显得至关重要。本文以基于 VxWorks 的串口通信系统为背景，对嵌入式串行数据通信系统中的差错控制技术进行了研究，使得系统的可靠性和实时性进一步提高。

本文首先分析了差错控制技术的研究现状，重点阐述了基于实时嵌入式系统的差错控制技术的研究背景和意义。然后，对返回-n 式自动请求重传(GBN-ARQ)差错控制技术中使用到的循环冗余校验(CRC-16)进行了分析和改进，提高了系统的可靠性。同时，基于 Markov 链对 GBN-ARQ 进行了性能分析，用 Markov 链描述了 GBN-ARQ 的工作过程，把 GBN-ARQ 的工作过程分为七个状态，画出了状态转移图，推导出了 GBN-ARQ 系统中有效传输效率和传输时延的表达式，并在 matlab 上对其进行了仿真分析，证明存在一个最佳帧长度使得系统的有效传输效率最大的同时传输时延可以降到最小。最后，实现了基于 VxWorks 的串口数据通信系统，依据系统功能对系统应用软件进行了任务的划分，对各个任务模块的设计进行了详细的阐述，重点在工作日志上传任务中的实现了对差错控制技术的应用，记录实验中数据传输的时延，验证实验中接受数据的正确性，进一步证明了在 GBN-ARQ 差错控制技术中，存在最佳帧长使得数据有效传输效率最大，时延最短。采用本文的 GBN-ARQ 差错控制技术提高了系统数据传输的实时性，保证了系统数据传输的可靠性。

**关键词：回退 N 帧-ARQ　马尔科夫　VxWorks　循环冗余校验**

# Abstract

As the development of embedded systems and data communication technology, timeliness and reliability in data communication system is also become more and more important .So, the error control technology's performance appears very important in the embedded data communication system. This paper is based on the VxWorks serial port communications system, the error control technology in embedded serial data communication system has been researched, system's reliability and timeliness are further enhanced.

In this paper, the present situation of the error control technology is introduced at first, focusing on the background and significance of the study about the error control technology based on the real-time embedded system. Then the CRC-16 used in the GBN-ARQ error control technology is analyzed to improve the reliability of system. The performance analysis is made about GBN-ARQ based on the Markov chain, the work process is described with Markov and state transition diagram is drawn, to drived the expression of the effective efficiency and delay of transmission in GBN-ARQ system, and the influence of the length of the data frame transmitted on efficiency and delay of the transmission is simulated with matlab, to conclude that there exists a best length of the date frame sent. At last, the serial data communication system based on VxWorks is implemented; the system application software is divided into several tasks according to the function of the system, and each task in the system is described in detail, focusing on the application of the error control technology in the task of uploading of the work log. The delay of data transmission in experiment is recorded and the accuracy of data is verified to prove that there exists a best length to data frame to make the efficiency of the data transmission biggest and delay shortest. So, the GBN-ARQ error control technology used in this paper could be satisfied with timeliness and reliability of the system.



**Keywords: GBN-ARQ Markov VxWorks CRC**

# 目 录


第一章 绪论 1
1.1 本文研究背景和意义 1
1.2 嵌入式系统的发展 2
1.3 差错控制技术的分类 3
1.3.1 差错控制技术简介 3
1.3.2 差错控制方式的分类 4
1.3.3 几种常用的检错码 5
1.4 串行数据通信系统中的差错控制方案设计 8
1.5 文组织结构安排 9
第二章 嵌入式实时操作系统 11
2.1 操作系统介绍 11
2.1.1 嵌入式系统介绍 11
2.1.2 实时系统介绍 12
2.1.3 常见的几种嵌入式操作系统 13
2.2 嵌入式实时操作系统 VxWorks 14
2.2.1 VxWorks 的特点 14
2.2.2 VxWorks 的组成 15
2.2.3 VxWorks 的编程 16
2.2.4 VxWorks 交叉开发环境 Tornado 19
2.3 本章小结 22
第三章 改进型 CRC-16 校验码的实现及性能分析 23
3.1 经典 CRC 校验的原理与实现 23
3.1.1 CRC 校验的一般原理 23
3.1.2 CRC 校验的性能分析 23
3.2 改进型 CRC-16 校验码在串行通信系统中的实现 25
3.2.1 用余式表计算 CRC 校验码的原理 25
3.2.2 改进型 CRC-16 在 VxWorks 下的实现 26
3.3 改进型 CRC 在 labview 下的仿真 27
3.4 本章小结 29
第四章 基于 Markov 链的 GBN-ARQ 性能优化分析 31
4.1 三种基本的 ARQ 31
4.2 GBN-ARQ 的 Markov 描述 33
4.3 GBN-ARQ 的性能指标 35
4.3.1 GBN-ARQ 的有效传输效率 35
4.3.2 GBN-ARQ 的传输时延 36
4.4 数值分析和仿真 37

4.4.1 有效传输效率分析.....37
4.4.2 传输时延分析.....38
4.4.3 最佳帧长分析.....40
4.5 本章小结.....42
第五章 基于 VxWorks 的串行通信系统的设计与实现.....43
5.1 串行通信系统的总体设计.....43
5.1.1 串行通信系统的功能分析.....43
5.1.2 串行通信系统中任务的划分.....43
5.1.3 软件流程.....45
5.2 系统各个任务模块设计.....46
5.2.1 主程序模块（Mainapp）.....46
5.2.2 通道检测任务（tComQuery）.....47
5.2.3 与上位机通信的任务（tPCDeal）.....47
5.2.4 50ms 周期任务（tSendTCCB）.....48
5.2.5 SSPC 数据处理任务（tDealCCB）.....49
5.3 GBN-ARQ 差错控制方法在系统中的应用.....50
5.3.1 GBN-ARQ 协议的定制.....50
5.3.2 GBN-ARQ 的实现流程.....51
5.3.3 工作日志上传时 GBN-ARQ 机制的验证.....52
5.4 本章小结.....53
第六章 总结和展望.....55
6.1 总结.....55
6.2 展望.....56
致 谢.....57
参考文献.....59
作者在读期间研究成果.....61

# 第一章 绪论

## 1.1 本文研究背景和意义

目前，嵌入式技术的应用越来越广泛，实时系统是嵌入式系统的一个非常重要的环节。在嵌入式实时系统的设计中，不可避免的会有大量的串行数据通信。串行通信实现过程简单，在嵌入式实时系统中使用串行通信，不仅可扩展嵌入式设备通信能力，还可扩大其应用范围。在嵌入式串行数据通信过程中，如何同时满足系统对数据实时性和可靠性的要求，是嵌入式实时系统设计的一个关键。

数据在信道中传输时，由于受到信道内噪声和各种干扰的影响，不可避免地会造成数字信号码元串扰，同时会产生差错。为了满足系统对差错率的要求，就必须采用差错控制技术。差错控制技术就是指在数据传输的过程中，发送端在发送出的数据中加入一定的冗余信息，在接收端则根据某种相关规则，通过信道译码来校验识别，进而纠正信道传输所造成的差错或重传出错数据。

实现差错控制的方式大致有自动请求重发（ARQ）、前向纠错（FEC）和混合纠错（HEC）等几类。与 FEC 和 HEC 系统相比，ARQ 系统中冗余码本身只有检错能力，而无纠错能力，但 ARQ 方式的优点是：编译码设备比较简单；在一定的多余度码元下，检错码的检错能力比纠错码的纠错能力要高得多，因而整个系统的可靠性极强，能获得极低的误码率；由于检错码的检错能力与信道干扰的变化基本无关，因此这种系统的适应性很强。ARQ 的这些优点满足了基于嵌入式系统的串行数据通信系统对高可靠性的要求。在 ARQ 的实现过程中，为了减少数据发送的延时，进一步提高系统的实时性，必须选择最佳的帧长度、滑动窗口长度以及重传次数。因此，在数据传输之前，需要选择合适的检错码，并对 ARQ 的性能进行深入的研究和分析，得出最佳的 ARQ 实现方案。

在三种基本的 ARQ 中，返回 n-ARQ（GBN-ARQ）由于具有比停-等式 ARQ（SW-ARQ）较高的系统效率以及比选择式 ARQ（SR-ARQ）较低的系统实现复杂度而备受广泛的研究[1]。Chu[2]和 Morris[3]讨论了基本 ARQ 协议及其扩展的最佳帧长确定，Martins[4]根据信道质量动态选择帧长，显著提高了 SR-ARQ 协议性能，但在这些讨论的过程中都假设了正向信道帧出错总能被收放检出并且反向信道无错因而应答无错，但实际上反向信道与正向信道一样是非理想的，接收方也不可能完全检测出所有的错误。Hlaing[5]等人基于 SW-ARQ 建立了自适应三模式多状态的 ARQ 系统，即根据信道噪声状况的不同发送帧长度不同的数据帧，在此基础上推导得出了系统延时和吞吐量的解析式，并对系统参数做了优化分析，获得更高的系统性能；Ranko[6]建立了基于等效传输周期的自适应三模式多状态的 ARQ 传

输机制，在这种机制下系统共有 α+β+γ+δ 个状态，更加准确地描述了系统的工作机理，因而得到的关于吞吐量的数学表达式也更为精确，但是上述讨论都是在重传次数不受限制的情况下进行推理分析的，但是，在实时系统中，对数据帧的重实时性要求很高，不可能无限次的重传。也有文献主要对 GBN-ARQ 系统进行吞吐量的算法分析[7,8]，也有在现代排队理论的基础上通过构造概率母函数的方法求得 GBN-ARQ 分组的队长和等待时延的分布函数及概率母函数[9,10]等，但过程较为复杂，过于理论化。因此，对于实时系统中的差错控制技术的研究文献较少，亟待进一步的研究。

## 1.2 嵌入式系统的发展

随着计算机技术、微电子技术、通信技术的不断创新和发展，嵌入式系统的发展已成为当今 IT 界最有发展前途的应用领域之一，已深入到电子、通信、自动控制领域等各个方面。

嵌入式系统是以应用为中心、以计算机技术为基础、软件硬件可裁剪、适用于应用系统，对功能、可靠性、成本、提及、功耗严格要求的专用计算机系统[1]。20 世纪 70 年代，以微处理器为核心的微型计算机以其小型、廉价、高可靠性特点，迅速从计算机房走了各种设备的控制单元中。这些微型机嵌入到某些应用场合中固定的设备当中，实现了对这些设备的智能化控制。它有着与通用计算机系统完全不同的技术要求与技术发展方向。嵌入式计算机系统的技术要求是对象的智能化控制能力；技术发展方向是与对象系统密切相关的嵌入性能、控制能力与控制的可靠性。

进入 20 世纪 90 年代，嵌入式技术全面展开，目前已成为通信和消费类产品的共同发展方向。在通信领域，数字技术正在全面取代模拟技术。硬件方面讲，不仅有各大公司的微处理器芯片，还有用于学习和研发的各种配套开发包。目前低层系统和硬件平台经过若干年的研究，已经相对比较成熟，实现各种功能的芯片应有尽有。而且巨大的市场需求给我们提供了学习研发的资金和技术力量。软件方面讲，也有相当部分的成熟软件系统。国外商品化的嵌入式实时操作系统，已进入我国市场的有 WindRiver、Microsoft、QNX 和 Nuclear 等产品。我国自主开发的嵌入式系统软件产品如科银(CoreTek)公司的嵌入式软件开发平台 DeltaSystem，中科院推出的 Hopen 嵌入式操作系统（虽然还不够完善）。

信息时代，数字时代使得嵌入式产品获得了巨大的发展契机，为嵌入式市场展现了美好的前景，同时也对嵌入式生产厂商提出了新的挑战，从中我们可以看出未来嵌入式系统的几大发展趋势：(1)是一项系统工程，因此要求嵌入式系统厂商不仅要提供嵌入式软硬件系统本身，同时还需要提供强大的硬件开发工具和软

件包支持；(2)网络化、信息化的要求随着因特网技术的成熟、带宽的提高日益提高，使得以往单一功能的设备如电话、手机、冰箱、微波炉等功能不再单一，结构更加复杂；(3)网络互联成为必然趋势；(4)精简系统内核、算法，降低功耗和软硬件成本；(5)提供友好的多媒体人机界面。

嵌入式操作系统作为运行在嵌入式系统上的操作系统，在实时高效性、硬件的相关依赖性、软件固态化以及应用的专用性等方面具有较为突出的特点。嵌入式操作系统不论从技术上还是从产业上都出现了一些明显的发展趋势和前景，主要表现在：可伸缩性，可移植性，可剪裁性和可配置性得到加强；出现行业标准；结构紧凑、高可用性、高可靠性；支持多处理器和分布式计算；可动态加载和升级软件。

据调查，目前国际上已有两百多种嵌入式操作系统，而各种各样的开发工具、应用于嵌入式开发的仪器设备更是不可胜数。在国内，虽然嵌入式应用、开发很广，只有三两家公司和极少数人员在从事这方面工作[11]。

## 1.3 差错控制技术的分类

### 1.3.1 差错控制技术简介

在 20 世纪 20 年代，怎样将信息传递的速度与可靠性提高，这已经成为一个非常重要的课题了.在数据传输的过程中发生错误后能在接收端发现并纠正的码称为纠错码，仅能发现错误但是不能纠正错误的码为检错码。为了使一种码具有检错或者纠错能力，必须对原来的码字增加冗余的码字，即把原来的码字按照某种规则变成具有一定冗余度的码字，而且使每个码字的码之间有一定的关系，则这种关系的建立即是编码。码字到达接收端后，接收端可以根据编码规则来判断传输过程是否有错误。有错时，纠错码可以按一定规则确定错误的位置并纠正，检错码与其他的方式结合也可以纠错。

目前，利用纠错码降低数据通信系统中数据传输的误码率，提高通信质量，延长计算机无故障运行时间等。而且，纠错码技术已开始渗透到很多领域。利用纠错码中的许多编译码原理和方法，与通信系统中其他方式相结合，得到令人惊喜的结果[12]

纠错码中比较重要的是分组码和卷积码。分组码是对信源序列进行分组，它的校验位仅仅同本组的信息位有关，分组码广泛应用于数据存储和数据通信系统中。卷积码不对信息序列进行分组编码，它的校验码不仅与当前的信息码有关，而且同以前的信息码有关。纠错码含有冗余信息，所以能够纠错，因此纠错码一般要比检错码长得多。通常情况下只采用检错码检错，数据的可靠性是通过重传

的方式来完成的。但是在信道环境比较恶劣的环境下才，如果才用重传则会导致重传次数很多，重传开销非常大，这个时候采用纠错码，信道质量越差，采用的纠错码汉明距离应该越大，冗余信息也就越多，编码长度响应变长[13]。

在通信系统中可靠性与实时性往往是一对矛盾。若要求实时，则必然使得每个数据码字所占的时间缩短，从而在受到干扰后产生错误地可能性增加，数据的可靠性下降。若要求可靠，则使得传送的数据的速度变慢。

因此，如何合理地解决实时性和可靠性这一对矛盾，是正确设计一个实时通信系统的关键问题之一。为保证传输过程的正确性，需要对通信过程进行差错控制。在许多数据通信系统中，广泛采用 ARQ 方式，此时的差错控制只需要检错。循环冗余校验 CRC 码是由分组码的分支而来的，其简称为循环冗余码，其编码简单而且误码率很低，在通信系统中得到了广泛的应用。

### 1.3.2 差错控制方式的分类

差错控制方式如图 1.1 所示，分为三类：自动请求重传（ARQ），前向纠错（FEC），混合纠错（HEC）。

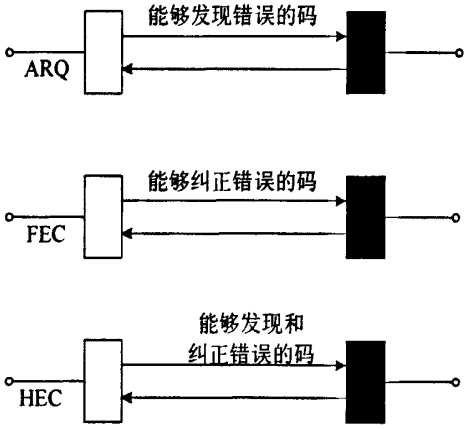


图 1.1 差错控制的基本方法

(1)自动请求重传（ARQ）

ARQ 也称检错重发，发送的码字是具有一定检错能力的冗余码，接收端通过检错译码发现有错码时，不产生输出，同时经反向信道启动发端的重发控制，将存储的源码字重新发送，然后重新检测，直到正确为止，采用该方式的通信结构如图 1.2 所示。ARQ 方式必须要有反向信道，再有它只适用于单个用户之间通信，而不能同时对多个用户进行差错控制。

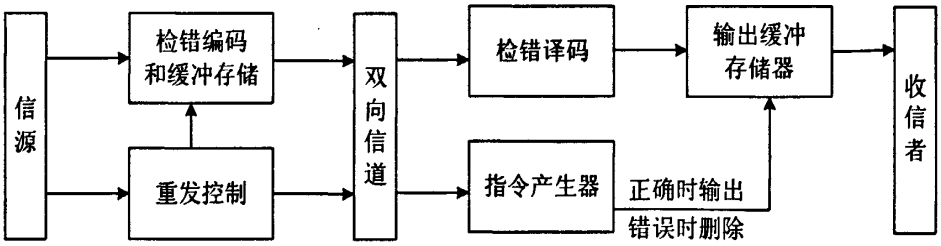


图 1.2 ARQ 方式的系统结构框图

(2)前向纠错（FEC）

前向纠错（FEC）方式，发送端发送能够纠错的码，接收端收到这些码后，通过纠错译码器不仅能自动地发现错误，而且能自动地纠正接收码字传输中的错误。它不存在因反复重发延误时间的缺点，控制电路也比 ARQ 简单，但是译码过程比较复杂，信道的适应性比较差。为了降低误码率，往往必须以最坏的信道条件来设计纠错码，故所需的多冗余码元比检错码要多得多，从而使编码效率很低。

(3)混合纠错（HEC）

混合纠错方式是发送端发送的码不仅能够被检测出错误，而且还具有一定的纠错能力。接收端收到码序列以后，首先检验错误情况，如果在纠错码的纠错能力之内，则自动进行纠错。如果错误太多，超过了码的纠错能力，但能检测出来，则接收端通过反馈信道，要求发送端重新发送的有错的信息帧。这种方式在一定程序上避免了 FEC 系统要求复杂的译码设备和 ARQ 系统信息连续性差的缺点，但它也存在着要求双相信道和不能用于多点通信的问题。

不论哪种控制差错的方式都是以降低实际的传输的效率来换取传输可靠性提高的。在信道已经确定的情况下，研究实时系统中的差错控制技术的一个重要任务是：如何以最小的代价来换取实时系统对实时性和可靠性的要求。

### 1.3.3 几种常用的检错码

差错控制编码的种类很多，但大体可分为两大类,即检错和纠错两类编码。所谓检错码，就是在接收端可以根据监督关系进行检查，并发现错误。所谓纠错编码，则是在接收端除发现错误外，还能进行自动纠正错误。两种码适用的环境不同：在高度可靠的信道上，比较合理的做法是适用检错码，当偶尔有错误发生时，只需重传整个数据帧即可。然而，纠错码适用于错误发生很频繁的信道上。

依据不同的分类标准，差错控制编码有不同的分类方式，如图 1.3 所示。

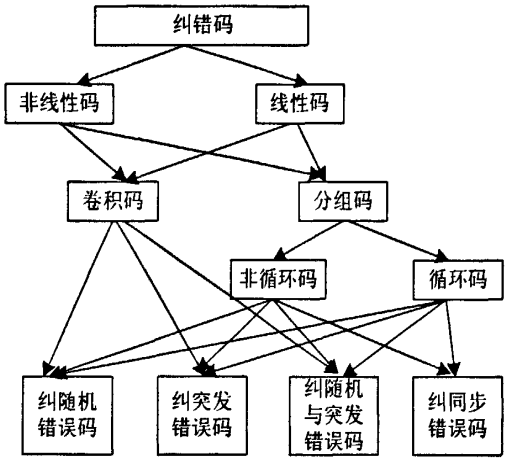


图 1.3 差错控制编码分类

根据监督码元与信息码元之间的关系分为线性码与非线性码。线性码的监督码元与信息码元之间的关系是线性关系，非线性码的监督码元和信息码元之间的关系是非线性关系。按照对信息元处理方法的不同而分为分组码和卷积码。分组码是把信息序列划分为 k 个码元一段，然后由这 k 个码元按照一定的规律产生 r 个监督元，这个 r 个监督元跟在 k 个码元后，构成 n=k+r 的一个码组，每一个码组的监督元只与本组的信息元发生关系，而与别组无关。卷积码是本组的监督元不仅与本组的信息元有关，还与前后若干组的信息元有关。分组码按照码的结构特点，又可分为循环码与非循环码下面简单介绍几种比较常用的检错码[12]：

(1)奇偶校验码

奇偶校验码，是一种简单的线性分组码。其方法是首先把信源编码后的信息数据流分成等长码组，在每一信息码组之后加入一位监督码元作为奇偶校验码，使得总码长 n（包括信息为 k 和监督位 r）中的码重为偶数称为偶检验，码重为奇数称为奇校验。如果在传输过程中任何一个码组发生奇数位错误，则收到的码组必然不再符合奇偶校验的规律，因此可以发现误码。奇校验和偶检验具有完全相同的工作原理和检错能力，原则上采用任一种都是可以的。

由于每两个 1 的模 2 相加为 0，故利用模 2 加法可以判断一个码组中码重是奇数或是偶数，模 2 加法等同于“异或”运算。对于偶校验，应满足 $a_{n-1}\oplus a_{n-2}\oplus\cdots\oplus a_1\oplus c_0=0$，故监督码元为 $c_0$ 可由下式求出：$c_0=a_1\oplus a_2\oplus\cdots\oplus a_{n-2}\oplus a_{n-1}$。不难理解，这种奇偶校验编码只能检出奇数个误码，而无法检知偶数个误码，对于连续多位的突发性误码也不能检知，故检错能力有限，另外该编码组的最小码距为 $d_0=2$，故没有纠错能力。

(2)行列监督码

行列监督码是二维的奇偶监督码，又称矩阵码，这种可以克服奇偶监督码不能发现偶数个差错的缺点，并且是一种可以纠正突发差错的简单纠正编码。其基

本原理与简单的奇偶监督码相似，不同的是每个码元要受到行和列的两次监督。具体编码方法如下：将若干个所要传送的码组编成一个矩阵，矩阵中每一行为一码组，每行的最后加上一个监督码元，进行奇偶监督。矩阵中的每一列则由不同码组相同位置的码元组成，在每列最后也加上也加上一个监督码元，进行奇偶监督。它的一致监督关系按行及列组成，每一行每一列都是一个奇偶监督码，当某一行（或某一列）出现偶数个差错时，该行（或该列）虽不能发现，但只要差错所在的列（或行）没有同时出现偶数个差错时，该行（或该列），则这种差错仍旧可以被发现。矩阵码不能发现的差错只有这样一类：差错数正好为 4 的倍数，而且差错位置正好构成矩形的四个角。因此，矩阵码发现错误的能力是十分强的，它的编码效率当然比奇偶监督码要低。

(3)恒比码

我国电信部门在国内通信中广泛采用的五单位保护电码就是就是一种恒比码。如表 1.1 所示，它采用五中取三的办法并只有是个数字（0-9）电码，每个数字电码均有三个传号“1”二位空号“0”，当发生一位差错时，五中取三的规则被破坏，从而可以发现差错[13]. 本来以 5 位码元组成的码组总共可以有 $2^5$=32 种，而恒比码规定只有确切地含有 3 个“1”， 2 个“0”的那些码组为准用码组，而有 3 个“1”，2 个“0”的 5 位码组共有 $C_5^3=\frac{5!}{(5-3)!3!}=10$ 种组合。一般情况下， 从“n 中取 m”(m＜n) 恒比码的码组数为：$C_n^m=\frac{n!}{(n-m)!m!}$，由此可以看出，恒比码实际上是 n 个码元传送 $\log_2 C_n^m$ 比特信息。这种恒比码比一般的奇偶校验码的检错能力稍微强些，适用于传输字母和符号。

表 1.1 “5 中取 3”恒比码编码

| 数字字符 | 普通的五单位码 | 恒比码 |
|---|---|---|
| 1 | 11101 | 01011 |
| 2 | 11001 | 11001 |
| 3 | 10000 | 10110 |
| 4 | 01010 | 11010 |
| 5 | 10101 | 10101 |
| 6 | 11100 | 11100 |
| 7 | 01100 | 01110 |
| 8 | 00011 | 10011 |
| 9 | 01101 | 01101 |

(4)循环冗余校验码（CRC）

循环码是一种分组码。在一个长度为 n 的码组中有 k 个信息位和 r 个监督位，监督位的产生只与该组内的信息位有关。通常称这种结构的码为（n，k）码，比值 $k/n$ 称为这种码的编码效率[14].其基本原理是：对于一个给定的（n，k）码，可以证明存在一个最高次幂为 n-k=r 的多项式 G(x)。根据 G(x)可以生成 k 位信息的校验码，而 G(x)叫做这个 CRC 码的生成多项式。

CRC 校验的基本思想是：利用线性编码理论，在发送端根据药传送的 k 位二进制码序列，以一定的规则产生一个校验用的监督码（即 CRC 码）r 位，构成一个新的二进制码序列数共 k+r 位，最后发送出去。在接收端，则根据信息码和 CRC 码之间所遵循的规则进行校验，以确定传送中是否出错。

## 1.4 串行数据通信系统中的差错控制方案设计

由于本文要设计的是嵌入式串行数据通信系统，是一个对可靠性和实时性要求很高的系统，信息传送时间远小于编码计算时间，而且是有线传输，即数据传输的误码率也比较（一般低于 $10^{-9}$）。假设误码率为 $10^{-10}$，即 $10^{10}$ 个时钟周期出一次错，接下来以 16 位 CRC 校验和汉明码校验为例，比较一下两种校验方式下的时间复杂度。用 Motorola 公司的 PowerPC8245 做为中央处理内核，用余式表计算一个字节的CRC校验的时间为0.1us,而汉明码校验一个字节的时间接近为0.15us，也就是说，每传送一个字节的数据采用汉明码校验要比采用 CRC 校验要多花费 0.05us 的时间,那么传送 $10^{10}$ 也就是多花 $10^{10} \times 0.05us/8 = 62.5s$ (若用 RS 码等纠错，则因为计算的复杂度更大，实际校验的时间会更长)，而采用 CRC 校验，出错重传时每个字节重传的延时为 $11/115200 = 0.95us$ ,远远小于 65.5s。

所以，有以上的分析和计算可以看出：在嵌入式串行数据通信系统中，即使 CPU 的工作主频很高，检错加重传比计算纠错的代价还是要低得低，因此，没有必要花费巨大的系统开销去实现纠错，只要对错误的数据帧进行重传即可。

而在前面介绍的几种检错码中，CRC 的性能开销比是最高的，而且实现方式简单，为了进一步提高系统的可靠性，系统对 CRC 进行了改进，作为数据传输的检错码。

综上所述，在嵌入式串行数据通信系统中，本文采用的是 CRC 加差错重传的方式来实现差错控制的。

## 1.5 文组织结构安排

本文对嵌入式串行数据通信系统中差错控制技术进行了研究，改进了CRC-16，提高了系统的可靠性；基于 Markov 链对 GBN-ARQ 进行了性能分析，实现了基于 VxWorks 嵌入式操作系统的串行数据通信系统，重点介绍了 GBN-ARQ 差错控制技术在该系统中的应用。文章共分为 6 章，下面阐述每章所做的各项工作。

第一章阐述了研究基于实时系统的差错控制技术的背景和意义，阐述了实时系统中 ARQ 技术的现状和不足，以及本文的研究方向，重点对比了几种常用的差错控制方式和差错控制编码，给出了本文中研究的差错控制方式。

第二章分析了实时嵌入式操作系统，主要介绍了 VxWorks 及其开发环境 Tornado，并开发基于 VxWorks 的串口驱动程序，保证了基于 VxWorks 的串口数据通信系统的正常通信。

第三章在对经典 CRC-16 进行分析和研究的基础上，在 VxWorks 下实现了改进型 CRC 校验，并对改进后的 CRC 校验的进行了性能分析。

第四章基于 Markov 链对 GBN-ARQ 的进行了性能分析，主要分析了每帧数据的帧长度对传输效率和传输时延的影响，从而能选择一个最佳帧长，提高 GBN-ARQ 的性能。

第五章实现了基于 VxWorks 的串口通信系统，重点实现了差错控制技术在串口通信系统中的应用。

第六章 对研究结果进行总结以及展望，提出有待于进一步研究的问题。

# 第二章 嵌入式实时操作系统

## 2.1 操作系统介绍

### 2.1.1 嵌入式系统介绍

早期的嵌入式系统中，软件和硬件密不可分，浑然一体，开发者多是电子工程、自动控制等领域的工程师，软件基本上都是用汇编实现。随着软硬件技术的发展，人们对嵌入式系统的功能要求越来越复杂（比如，手机可以照相，摄影还可以上网读新闻、玩游戏等），而性能（比如，可靠性，安全性、响应速度等）要求也越来越高，与此同时，嵌入式软件的开发和硬件仍然密不可分，从软硬件与平台选择、设计、开发到测试与集成，整个过程都是软硬件并行交互进行，这样嵌入式系统开发以及成为一项很复杂的系统工程。

嵌入式系统主要由嵌入式处理器、外围硬件设备、嵌入式实时操作系统（RTOS）以及特定的应用程序等四部分组成，是集软硬件于一体的可独立工作的“器件”。图 2.1 所示为典型的 32 位 ARM 嵌入式系统构成框图，如图所示嵌入式系统分为软/硬件两个部分，其中软件部分从下至上依次是驱动层、操作系统、和应用层。对于软件部分，可以依据系统设计的复杂度、稳定性和扩展性的要求决定是否采用操作系统。一般对于要求较低和功能单一的嵌入式系统可以不使用操作系统，而对于要求功能多样而且功能经常变化、可扩展的嵌入式系统，应该选择使用操作系统以简化设计。硬件部分是由嵌入式 ARM 处理器/处理器核构成的芯片/片上系统 SOC（System On Chip）/可编程片上系统 SOPC（system On Programmable Chip），加上 I/O 设备、存储器和人接口构成。在实际的嵌入式系统开发中通常根据应用领域和使用环境的不同，选择不同 IC 公司生产的基于 ARM 核的 SOC 作为系统的核心控制部件。在嵌入式系统硬件构成中，处理核心控制部件以外的各种存储器、输入/输出接口、人机接口等都被称为外围设备。

嵌入式系统开发通常采用软硬件协同设计，将软件设计和硬件设计作为一个整体并行设计，找到软硬件的最佳结合点，从而使系统高效工作，这种设计方法，可以充分利用现有的软硬件资源，缩短系统开发周期、降低开发成本、调高系统性能，避免由于独立设计软硬件体系结构而带来的弊端[14]。随着信息化、智能化、网络化的发展，嵌入式技术也将获得广阔的发展空间。

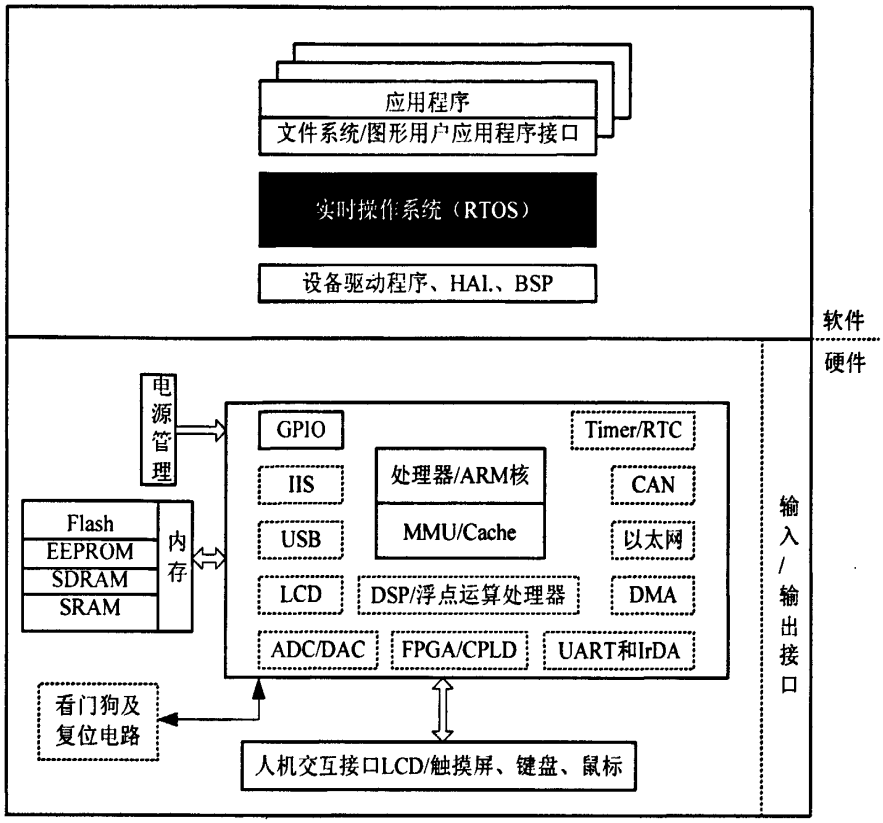


图 2.1 嵌入式系统构成框图

### 2.1.2 实时系统介绍

实时系统是嵌入式系统中非常重要的一环。所谓实时系统，是一种计算机系统的操作方式，它接受来自于外界的输入数据，并能在可预测的时间内作出反应，获得相应计算结果并输出[15,16].实时系统具有一下特点：

(1)时间约束性，实时系统的任务具有一定的时间约束（截止时间）。根据截止时间，按照系统对时间限制的满足程度，实时系统可分为硬实时（Hard Real Time）系统和软实时（Soft Real Time）系统。硬实时系统是指那些对每个人物调度时间要求严格的系统，如果不满足时间限制的要求，则会对系统带来毁灭性的后果。软实时系统是指那些对每个任务调度时间要求不是很严格的系统，即使超过了时间限制的要求，也不会对系统带来毁灭性的后果[17]。

(2)可预测性，可预测性是指系统能够对实时任务的执行时间进行判断，确定是否能够满足任务的时限要求。由于实时系统对时间约束要求的严格性，使可预测性称为实时系统的一项重要性能要求。除了要求硬件延迟的可预测性以外，还要求软件系统的可预测性，包括应用程序的响应时间是可预测的，即在有限的时间内完成必须的工作；以及操作系统的可预测性，即实时原语、调度函数等运行

开销应是有界的，以保证应用程序执行时间的有界性。

(3)可靠性，大多数实时系统要求有较高的可靠性。在一些重要的实时应用中，任何不可靠因素和计算机的一个微小故障，或某些特定强实时任务（又叫关键任务）超过时限，都可能引起难以预测的严重后果。为此，系统需要采用静态分析和保留资源的方法及冗余配置，使系统在最坏情况下都能正常工作或避免损失。可靠性已成为衡量实时系统性能不可缺少的重要指标。

(4)与外部环境的交互作用性，实时系统通常运行在一定的环境下，外部环境是实时系统不可缺少的一个组成部分。计算机子系统一般是控制系统，它必须在规定的时间内对外部请求做出反应。外部物理环境往往是被控子系统，两者互相作用构成完整的实时系统。大多数控制子系统必须连续运转以保证子系统的正常工作或准备对任何异常行为采取行动。

早期的实时系统功能简单，包裹单片机，简单的嵌入式实时系统等，其调度过程也相对简单。随着实时系统应用范围的不断扩大，系统复杂性不断提高，实时系统具有一下新特点：

(1)多任务类型，在实时系统中，不但包括周期任务、偶发任务、非周期任务，还包括非实时任务。实时任务要求要满足时限，而非实时任务要求要使其响应时间尽可能的短。多种类型任务的混合，使系统的可调度性分析更加困难。

(2)约束的复杂性，任务的约束包括时间约束、资源约束、执行顺序约束和性能约束。时间约束是任何实时系统都固有的约束。资源约束是指多个实时任务共享有限的资源时，必须按照一定的资源访问控制协议进行同步，以避免死锁和高优先级任务被低优先级任务堵塞的时间（即优先级倒置时间）不可预测。执行顺序约束是指各任务的启动和执行必须满足一定的时间和顺序约束。例如，在分布式端到端（end-to-end）实时系统很重，同一任务的各子任务之间存在前驱/后驱约束关系，需要执行同步协议来管理子任务的启动和控制子任务的执行，使它们满足时间约束和系统可调度要求。性能约束是指必须满足如可靠性、可用性、可预测性、服务质量（Quality of Service,QoS）等性能指标。

(3)具有短暂超载的特点，在实时系统中，即使一个功能设计合理、资源充足的系统也可能由于系统元件老化、外围设备错误或者系统发生故障、环境的动态变化、应用规模的扩大等原因超载。

### 2.1.3 常见的几种嵌入式操作系统

(1)VxWorks

VxWorks 是美国 WindRiver 公司的产品，是目前嵌入式系统领域中应用很广泛，市场占有率比较高的嵌入式操作系。VxWorks 实时操作系统由 400 多个相对

独立、短小精悍的目标组成，用户可以根据需要选择适当的模块来裁剪和配置系统;提高基于优先级的任务调度、任务剑同步与通信、中断处理、定时器和内存管理等功能，符合 POSIX 规范的内存管理，以及多处理器控制程序；并且具有简明易懂的用户接口，在核心方面甚至可以萎缩到 8KB。

(2)Linux

Linux 作为下一代嵌入式系统，不论在技术上还是在商业上都是非常有优势的。从技术方面看，Linux 存在许多现实的、可以量化的优势，例如广泛的硬件支持、良好的可伸缩性、卓越的性能、极高的可靠性和开放的 API。从成本方面看，不用交版税、免费或廉价的软件组成以及免费的源代码，这些都会降低产品成本和提高产品灵活度上得到很大的优势。以上是 Linux 能够给嵌入式系统开发带来的优势。

(3)μC/OS - II

μC/OS - II 是在 μC/OS 的基础上发展起来的，是美国嵌入式系统专家 Jean J.Labrosse 用 C 语言编写的一个结构小巧、抢占式的多任务实时内核。μC/OS - II 能管理 64 个任务，并能提供任务调度与管理、内存管理、任务间同步与通信、时间管理和中断服务等功能，具有执行效率高、占用空间小、实时性能优良和可扩展性强等特点。

(4)Windows CE

虽然 Windows CE 是从整体上为有限资源的平台设计的多线程、完整优先权、多任务的操作系统。它的模块化设计允许它对于从掌上电脑到专用的工业控制器的用户电子设备进行定制。操作系统的基本内核需要至少 200k 的 ROM。

在上面介绍的几种操作系统中，VxWorks 被广泛地应用在通信、军事、航空、航天等高尖技术及实时性要求高的领域，尤其是在许多关键应用方面，VxWorks 还是一枝独秀，例如，美国波音公司就在最佳的 787 客机中采用了此操作系统；而在外层空间探索领域，VxWorks 则一直是美国太空总署 NASA 的最爱。结合 ELMC 本身对实时性和可靠性的要求，系统采用 VxWorks 做 RTOS 完成开发和设计。

## 2.2 嵌入式实时操作系统 VxWorks

### 2.2.1 VxWorks 的特点

(1)可靠性

VxWokrs 经过 20 多年的市场应用和验证，操作系统本身是高度可靠的，另外构造一个可靠性的系统是由整个系统设计来保证的，VxWorks 操作系统为了支持

软件可靠性设计，还提供了分布式消息队列和异常处理等机制支持系统的冗余设计和容错设计。

(2)强实时性

VxWorks 在 80486 66MHz 下的 tc（任务切换时间）、ts（系统调用时间）和中断响应时间都在几个微妙，如果 CPU 频率更快的话，这些性能指标会更好，而且更重要的是 VxWorks 在设计时保证了系统响应时间的确定性。另外 VxWorks 支持基于任务优先级的抢占调度和丰富的任务间通信机制，可以保证同步事件和异步事件并行处理，使得在同样的硬件环境下，实时性要求高的任务可以在规定的时间内完成，为应用程序的开发留下了更大的余地。

(3)可裁减性

VxWorks 适应嵌入式系统多样性的特点，采用微内核设计方法，操作系统的基本功能有 Wind 内核提供，其他系统功能则以系统组件的形式存在，提高了系统的可裁减性，使得系统最小可裁减到 8KB。这种裁减甚至可以做到函数级的裁减，即系统的函数功能如果在应用中不使用的话，就可以将该函数对应的代码裁减掉。

(4)可移植性

操作系统可以认为是提供了系统函数的函数库。VxWorks 在设计的时候，这些函数库只与目标机 CPU 的体系结构相关，（事实上，只有 Wind 内核与 CPU 的体系结构相关，其他的功能组件与 CPU 也是无关的），而与硬件板的设计师无关的。从而，使得 VxWorks 操作系统具有很好的移植性，与硬件相关的程序完全有 BSP(板级支持包）提供了。

### 2.2.2 VxWorks 的组成

VxWorks 操作系统包括了板级支持包（BSP）、进程管理、存储管理、设备管理、文件系统管理、网络协议以及系统应用等几个部分。VxWorks 只占用了很小的存储空间，并可高度裁减，保证了系统能以较高的效率运行。如图 2.2 所示，VxWorks 操作系统由以下几个主要部分组成：板级支持包（BSP）、微内核 wind、网络系统、文件系统和 I/O 系统。

(1)板级支持包

板级支持包（BSP）对各种板子的硬件功能提供了统一的软件接口，它包括硬件初始化、中断的产生和处理、硬件时钟和计时器管理、局域和总线内存地址映射、内存分配等。每个 BSP 包括一个 ROM 启动（Boot ROM）或其他的启动机制。

(2)微内核 wind

VxWorks 的核心作为 wind，包括多任务调度、任务间同步、进程间通信机制、中断处理、看门狗和内存管理机制。一个多任务环境允许实时应用程序以一套独

立任务的方式构筑，每个任务拥有独立的执行线程和它自己的一套系统资源。进程间通信机制使得这些任务的行为同步和协调。

(3)网络系统

VxWorks 的网络结构提供了对其他网络和 TCP/IP 网络系统的透明访问。

(4)文件系统

VxWorks 提供的快速文件系统适合于实时系统应用。它包括了几种支持使用块设备的本地文件系统。这些设备都是用一个标准的借口从而使得文件系统能够被灵活的移植到设备驱动程序上。VxWorks 也支持 SCSI 磁带设备的本地文件系统。VxWorks I/O 体系结构甚至还支持在一个单独的 VxWorks 系统上同时并存几个不同的文件系统。VxWorks 支持以下几种文件：DosFs、Rt1Fs、rawFs、tapeFs、TrueFFS、CdRomFs。

(5)I/O 系统

VxWorks 提供了一个快速的与 ANSI C 兼容的 I/O 系统；它包括 Unix 标准的缓冲 I/O 和 POSIX 标准的异步 I/O。VxWorks 包括以下驱动程序：网络驱动、管道驱动、RAM 盘驱动、SCSI 驱动、键盘驱动、显示驱动、磁盘驱动、串行口和并行口驱动。

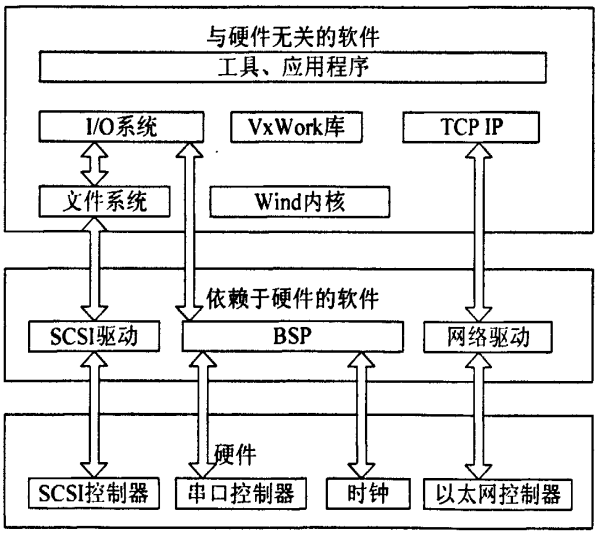


图 2.2 VxWorks 操作系统组成框图

### 2.2.3 VxWorks 的编程

1、多任务机制

VxWorks 实时内核 Wind 提供基本的多任务环境。在单个 CPU 系统中，多任务构造出多个线程并发的假象。事实上，系统根据某种调度算法，将 Wind 内核插入到这些任务中执行。每个独立的程序称为一个任务；每个任务拥有自己的上下

文（即 CPU 环境和系统资源等），并由系统内核调度运行。在上下文交换时，任务的上下文保存在任务控制模块（TCB）中。一个任务的上下文包括：

(1)任务的程序计数器；

(2)处理器的通用寄存器；

(3)局部变量和堆栈；

(4)时间片定时器；

(5)标准输入、输出和错误输出的 I/O 重定向；

(6)内核控制结构；

(7)信号处理程序；

(8)调试和性能监视。

2、任务调度机制

在 VxWorks 中下，一个任务可有多种状态，任务状态可以反应当前任务在系统中所处的情形。最基本的状态有 4 种：

(1)就绪（Ready）：处于这种状态的任务除了等待 CPU 外，不需要等待其他资源；

(2)阻塞（Pend：由于一些资源不可用而阻塞的任务状态；

(3)延迟（Delay）：暂时交出 CPU 的的使用权，在等待超时过后才能再次获得 CPU 的使用权。

(4)挂起（Suspend）：这种任务状态不能执行，主要用于调试，不会约束状态转换，仅仅约束任务的执行。

(5)运行（Run）：任务处于被执行的状态。

任务状态的变化是应用程序进行内核函数调用的结果。但任务创建时，它处于挂起状态。只有激活任务，才能进入准备状态。VxWorks 系统中任务间的状态跃迁如图 2.3 所示。由于处理器的独占性，在一个时刻只允许一个任务运行并享用处理器资源。任务间切换时会发生上下文切换，但是除了运行状态外，任务在状态之间切换并不会发生上下文切换。

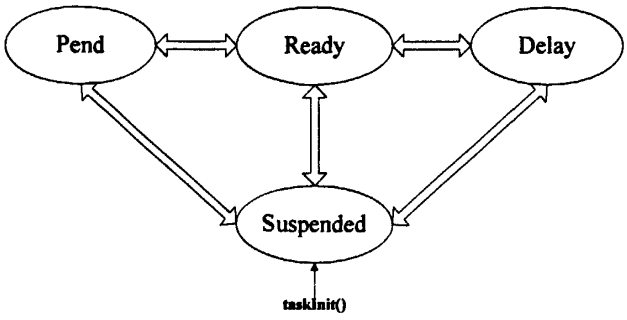


图 2.3 VxWorks 任务状态转换图

Wind 内核的多任务处理需要一个调度法则对 CPU 准备运行的任务进行分配。

Wind 内核提供了两种调度方法：优先级抢占式调度和时间片轮转调度方法。基于优先级的抢占调度方式是系统的默认工作方式。当然，也可以根据应用程序的需要选择时间片轮转的调度方式。

3、任务间通信机制

VxWorks 为了保证各个独立的任务之间可以协同工作，提供了一整套任务间的通信机制。主要包括了以下几种：

(1)共享数据结构：最直接有效的方法。由于所有 VxWorks 任务共存于单一的线性地址空间，多任务间共享数据结构非常方便。

(2)信号量：它是提供任务间通信、同步和互斥的最优选择，提供任务间最快速的通信，也是提供任务间同步和互斥的主要手段。VxWorks 提供了 3 种信号量来解决不同问题。如表 2.1 所示。

表 2.1 VxWorks 的三种信号量类型

| 类型 | 功能描述 |
| --- | --- |
| 二进制信号量 | 实现任务间的同步和互斥操作的最佳方式，需要的系统开销最小。 |
| 互斥信号量 | 特殊的二进制信号量，用于解决内在的互斥问题：优先级继承、删除安全等。 |
| 计数信号量 | 除了同二进制信号量那样工作外，还保持对信号量释放次数的跟踪。 |

(3)消息队列：消息机制使用一个被各有关进程共享的消息队列，任务之间这个消息队列发送和接收消息，任务之间全双工信息传送。

(4)管道：它是一种虚拟的 I/O 设备。任务能调用标准的 I/O 函数打开、读出、写入管道。当任务试图从一个空的管道中读取数据，或向一个满的管道中写入数据时，任务被阻塞，和消息队列类似。

4、中断机制

VxWorks 操作系统下的中断处理程序运行在特定的上下文中，不涉及任何上下文的切换。这种中断处理机制给 VxWorks 带来了快速响应时间。同时，对中断处理程序的编写有几个规则：

(1)不能调用可能引起调用阻塞的函数；

(2)不能通过 VxWorks 驱动执行 I/O 口；

(3)不能调用使用浮动协处理器的函数。

在 VxWorks 操作系统下，为了使中断服务程序与任务完成通信，提供了几种方法：信号量、消息队列、管道以及网络套接字等。中断服务程序与任务之间的通信时任务级通信，可以参考任务之间的通信方式。信号量是诸多通信机制中常

用的方法，通过中断服务程序释放某一信号量，响应任务等待该信号量，从而完成一次通讯[17,19]。

### 2.2.4 VxWorks 交叉开发环境 Tornado

1、Tornado 的架构

Tornado 是 Wind River 公司为 VxWorks 操作系统专门开发的集成开发调试工具[19],集成了设计开发和分析等特性，使之成为一有机整体。它提供一种方便的应用程序开发方式，使得对目标机的影响做到很小。如图 2.4 所示是 Tornado 开发环境架构。

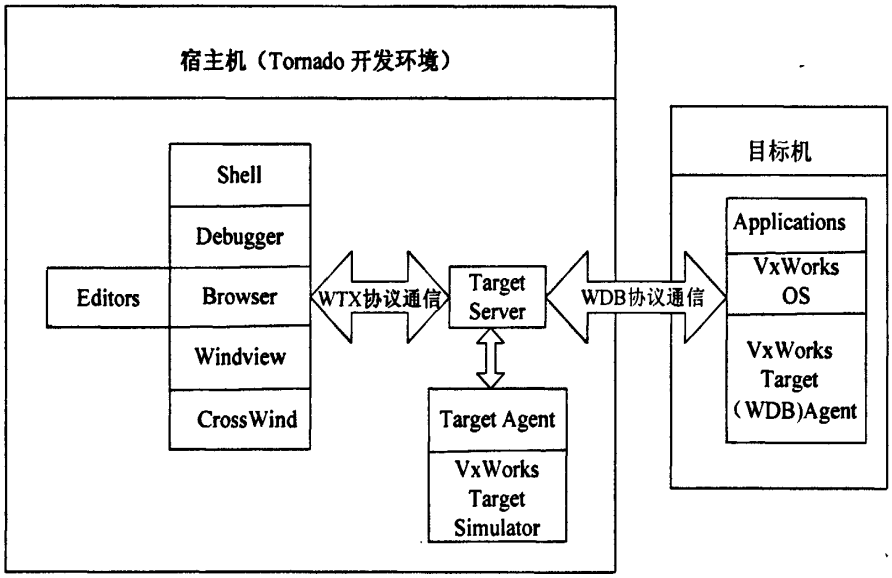


图 2.4 Tornado 开发环境架构

Tornado 所有的工具都是驻留在开发平台上的，并且通过一个中央服务器处理与目标机的通信。无论连接方式是以太网、串口线还是仿真器，都可以使用。Tornado 的开发架构如图 2.4 所示。Tornado 主要包括 3 个高度集成的部分，分别是：

(1)运行在宿主机上的交叉开发工具；

(2)运行在目标机上的高性能的实时操作系统 VxWorks；

(3)连接宿主机和目标机的多种通信方式，如：以太网、串口线、仿真器等。

在宿主机上运行的是 Tornado 集成开发工具，它包括编译器、命令执行工具 Windshell、Browser 浏览器、软件逻辑分析仪 WindView 和目标服务器 Target Server。其中，Target Server 主要管理符号表并处理它和目标机之间的通信，所有的主机工具均通过它来访问目标机。目标机代理（Target Agent）是运行在目标机上的组件，它是一个小监控程序，以 tWdbTask 任务形式运行，并管理主机到目标机内存和其他工具的访问。Tornado 的核心工具主要包括以下几种[20]：

(1)集成仿真器 VxSim：

它可以使开发者在没有实际目标机的情况下，先进行仿真开发。通常 VxSim 模拟下的系统性能和运行在目标机上真实的 VxWorks 系统性能是一样的。开发者可在 VxSim 中链接应用程序和编译 VxWorks 映象。

(2)命令执行工具 WindShell：

WindShell 界面允许用户与目标组件相互作用。它可以解释和执行几乎所有的 C 语言表达式，包括到函数的调用和名字在系统符号表中的变量引用。WindShell 还可以解释常规的工具命令语言 TCL。

(3)浏览器 Browser：

Tornado 中的 Browswer 是 Shell 的相应的图形化工具。像 Shell 一样，它提供符号化的信息。Browser 的主窗口提供目标系统的整个状态，允许开发者发起对个别的目标操作系统的对象的状态监测信息的显示，如任务、信号量、消息队列、内存对象和看门狗定时器等。这些显示可以根据开发者的要求而更新或定时更新。

(4)软件逻辑分析仪 WindView：

WindView 提供了详细的运行在集成模拟器上的嵌入式 VxWorks 应用的动态行为，图形化地显示了任务、中断和系统对象之间的复杂的相互作用关系。监测目标硬件上的系统行为。

(5)调试器 Debugger：

Tornado 调试器集成了图形和命令行调试接口最好的特征。最普通的调试行为，如设置断点、控制程序执行可以在 Debugger 中执行。调试器包含一个观察窗口，允许用户在表格窗口观察一组表达式。在调试器的图形用户接口中可以快速地修改变量、寄存器的值和局部变量，指定不同变量组的基数。同时，程序列表和数据监视窗口为应用程序关键信息提供了一个可视内容界面。

(6)C 和 C++编译环境的支持：

Tornado 提供了交叉编译器、iostreams 类库和各种其他的工具，支持 C 语言和更新的 C++语言。交叉编译器包含一些优化，允许开发者产生快速、有效和紧凑的代码。

(7)项目管理器 Project：

可定制的项目管理特性（Project Management Facility）。简化了 VxWorks 应用开发中的组织、配置和创建工作。它包括配置 VxWorks 特性的图形工具 WindConfig，允许用户在上百个可裁剪的特性中选择（包括选择编译选项分析代码相关性和大小），构成适合用户特性的 VxWorks 操作系统运行环境，并生成相应的 BSP 配置。同样也提供了一个通用设置管理工具 ClearCase 用来进行基本配置。还有一个动态链接特性的装载器 Loader，可在调试中动态加载用户模块或系统模块。

2、交叉开发环境的建立

在 Tornado 集成环境上的开发，其基本点上是动态链接和加载，即 Tornado 允许开发者将目标模块加载到目标系统上去，这种动态链接和加载功能是 Tornado 系统的核心功能，它可以使开发者省去通常的开发步骤——在主机上将应用程序与内核链接起来，然后将整个应用程序下载到目标系统上去。这样，编辑-测试-调试的周期就会大为缩短；而且，所有的模块都是共享的，主机的应用程序模块也不需要重新进行链接，所以，把目标模块加载到运行 VxWorks 的目标系统只中可以达到调试和重新配置的目的。

通常，设计的主板没有软件的自开发能力，所以需要一台通用计算机来辅助开发，这台通用计算机可以是 PC 或者工作站；我们称辅助软件开发的通用计算机为宿主机（Host），而用户自己开发的主板为目标机（Target），显示器用于软硬联调时，显示程序代码的打印信息。如图 2.5 所示,宿主机上要有一个集成开发环境来辅助我们的软件开发，该开发环境包括交叉编译器（Cross Complier）和交叉调试器（Cross Debugger）。所谓交叉编译器就是在宿主机上编译，从而生成可以在目标机上运行的二进制代码镜像文件（Image）；交叉调试器就是在宿主机和目标机之间的某种耦合方式实现前后台调试。在开发 VxWorks 操作系统上，宿主机上运行的集成开发环境就是 Tornado。Tornado 可以编译生成在目标机上运行 VxWork 的可执行代码程序镜像文件。在系统安装时，集成调试环境和 VxWorks 的目标文件（obj 文件）都安装到了宿主机上，编译生成的目标机上运行的可执行代码程序镜像文件。

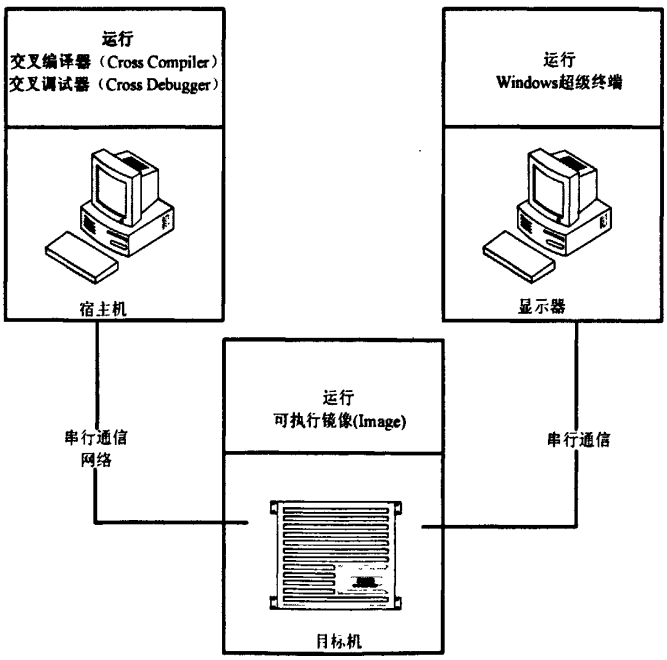


图 2.5 Tornado 交叉编译环境

## 2.3 本章小结

本章主要介绍了嵌入式实时操作系统，重点介绍了嵌入式实时操作系统VxWorks。具体介绍了 VxWorks 实时系统的特点组成结构，以及其交叉开发环境Tornado。

# 第三章 改进型 CRC-16 校验码的实现及性能分析

ARQ 差错控制方式中主要采用的就是 CRC 校验，CRC 校验是一种检错能力较强的检错码，在差错控制系统中已得到广泛应用。通常情况下，CRC 校验的实现方法是逐比特计算 CRC 校验码的。考虑到基于 VxWorks 的串口通信系统的实时性和数据传输量的要求，本文采用了一种按字节计算CRC校验码的方法，并对CRC校验码进行了一些改进，进一步提高了 CRC 校验码的检错能力。

## 3.1 经典 CRC 校验的原理与实现

### 3.1.1 CRC 校验的一般原理

CRC 校验的基本思想[21]是利用线性编码理论，在发送端根据要传送的 k 位二进制码序列，以一定的规则产生一个校验用的监督码（CRC 码）r 位，附在信息码后面，构成一个新的二进制码序列数共（k+r）位，最后发送出去。在接收端，则根据信息码和 CRC 码之间所遵循的规则进行校验，以确定传送中是否出错。

在代数编码理论中，将位串看成是系数为 0 或 1 的多项式。一个 k 位的帧看作是一个 k-1 次多项式的系数列表，该多项式共有 k 项，从 $x^{k-1}$ 到 $x^0$。在进行 CRC 编码时，设编码前的原始信息码多项式位 m(x)，m(x)的最高幂次为 k-1，生成多项式位 g(x)，g(x)的最高幂次为 r；CRC 码多项式位 r(x)；编码后的带 CRC 的信息码多项式位 C(x)，C(x)的最高幂次为 n（n=k+r）。

计算校验和的基本算法如下：

(1)在原始信息码 m(x)的低位端加上 r 个 0 位，所以该帧现在包含 k+r 位，对应多项式位为 $x^r m(x)$。

(2)利用模 2 除法，用对应于 g(x)的位串去除对应于 $x^r m(x)$ 的位串。

(3)利用模 2 加法，从对应于 $x^r m(x)$ 的位串中加上余数（总是小于等于 r 位）。结果就是被传输的带校验和的帧 C(x)。用公式可以表示为：$C(x)=x^r m(x)+r(x)=q(x)g(x)$，式中：q(x)为商式。

接收端收到了信息后用生成多项 g(x)去做模 2 除，若得到余数为 0，则码字无误。

### 3.1.2 CRC 校验的性能分析

假设信道的错误信息码为 E(x)，则接收端在收到带校验和的帧之后，用生成

多项式 g(x)去除 C(x)+E(x)，即(C(x)+E(x))/g(x)。C(x)/g(x)是 0，所以计算的结果是 E(x)/g(x)。当 E(x)/g(x)不等于 0 时，且 E(x)也不等于 0，CRC 可以检错。

如果只有一位发生错误，即 $E(x)=x^i$，这里 i 决定了错误发生在哪一位上。如果 g(x)包含了两项或者更多项，则它永远也不会除尽 E(x)，结论是：所有一位的错误都将被 CRC 检测到；

如果有奇数个错误,E(x)有奇数项，并且可以被 x+1 除尽。于是，将 E(x)分解成(x+1)Q(x),令 x=1,则得到 E(1)=(1+1)Q(1)，所以 E(1)=0.如果 E(x)有奇数项的话，则用 1 代替所有的 x，结果总是 1.因此，奇数多项式不可能被 x+1 除尽。结论就是：以 x+1 作为 g(x)的一个因子，就可以捕捉到所有奇数个位变反的错误。

如果有长度为 m（m<r）的突发性错误，则可以用 $x^i(x^{m-1}+x^{m-2}+\cdots 1)$ 来表示，这里 i 决定了突发性错误的位置离帧的最右端的距离有多远。如果 g(x)包含一个 $x^0$ 项，则它不可能有 $x^i$ 作为因子，所以，如果括号内表达式的阶小于 g(x)的阶，则余数永远不可能为 0。结论就是：如果 x 不是 g(x)的因子，那么长度小于等于 g(x)次数的突发错误都可以检测出来。

如果有长度为 m(m=r+1)突发性错误。则当且仅当错误多项式等于 g(x)的时候，错误多项式除以 g(x)的余数才为 0.根据突发性错误的定义,第一位和最后一位必须为 1，所以它是否与 g(x)匹配要取决于其他 r-1 个中间位。如果所有的组合被认为是等概率的，则这样一个不正确的帧被当作有效帧而接收的概率是 $(\frac{1}{2})^{r-1}$。

同样也可以证明，当一个长度大于 r+1 位的突发性错误发生的时候，或者几个短一点突发性错误发生的时候，一个坏帧被当作有效帧而通过检测的概率为 $(\frac{1}{2})^r$，这里假设所有的位模式是等概率的。

综合这些性质，有如下结论：若适当选取 g(x),使其含有（x+1）因子、常数项不为 0 且周期大于等于 n，那么，由此 g(x)作为生成多项式产生的 CRC 码可检测所有的奇数位错、双错和突发长度小于等于 r 的突发错误以及突发长度为 r+1 的突发错误和突发长度大于 r+1 的突发错误。具体到 CRC-16 ，依据该多项式满足的要求，CRC-16 对所有奇数个错误、所有突发长度小于等于 16 的突发错误以及所有的双错的检错率为 100%。据文献[22]分析，大约是 65000 个数据包中才会发生一个不可检测错误，这对于中低速和可靠性要求不高的串行数据通信已经足够好了，甚至可以忽略。但是，近年来随着嵌入式数据通信的飞速发展，对可靠性的要求越来越高，不可检错错误的影响已经不可忽视。下一节将对 CRC-16 进行改进，以进一步改善系统的可靠性。

## 3.2 改进型 CRC-16 校验码在串行通信系统中的实现

改进型 CRC-16 校验的基本思想是：用查余式表的方法按字节计算源数据帧的 CRC 校验码附在数据帧后，形成数据帧 A；然后用列奇偶校验的方式（一个字节作为一行）对数据帧 A 进行奇偶校验，附在数据帧 A 后，形成数据帧 B，作为最终的发送数据帧。

### 3.2.1 用余式表计算 CRC 校验码的原理

按比特算法简单且容易实现，对任意 CRC 宽度都适用，在发送的数据不长的时候可以使用。但是，如果发送的数据帧很长的话，这种方法就不合适了。按比特算法就算 CRC，每次只能计算一位，效率太低。为了提高效率，可以一次处理 4 位，8 位、16 位、32 位。

用余式表计算二进制序列的 CRC 码的原理如下所示：

对一个二进制序列可以按字节表示为：

$$B(x)=B_n(x).2^{8n}+B_{n-1}(x).2^{8(n-1)}+\cdots+B_1(x).2^8+B_0(x) \qquad \text{式(3-1)}$$

式（3-1）中 $B_n(x)$ 为一个字节。

求此二进制的 CRC-16 码时，先乘以 $2^{16}$，再除以多项式 g(x),所得的余数既是所要求的 CRC 码，如式(3-2)所示

$$\frac{B(x).2^{16}}{g(x)}=\frac{B_n(x).2^{16}}{g(x)}.2^{8n}+\frac{B_{n-1}(x).2^{16}}{g(x)}.2^{8(n-1)}+\cdots+\frac{B_1(x).2^{16}}{g(x)}.2^8+\frac{B_0(x).2^{16}}{g(x)} \qquad \text{式(3-2)}$$

设 $$\frac{B_n(x).2^{16}}{g(x)}=Q_n(x)+\frac{R_n(x)}{g(x)} \qquad \text{式(3-3)}$$

式(3-3)中 $Q_n(x)$ 为商，是整数，$R_n(x)$ 位 16 位的二进制余数。

将式(3-3)代入式(3-2)，得到：

$$\frac{B(x).2^{16}}{g(x)}=Q_n(x).2^{8n}+\left[\frac{R_n(x).2^8+B_{n-1}(x).2^{16}}{g(x)}\right].2^{8(n-1)}+\cdots+\frac{B_1(x).2^{16}}{g(x)}.2^8+\frac{B_0(x).2^{16}}{g(x)} \qquad \text{式(3-4)}$$

又如下：

$$R_n(x).2^8=R_{nH8}(x).2^{16}+R_{nL8}(x).2^8 \qquad \text{式(3-5)}$$

其中 $R_{nH8}(x)$ 是 $R_n(x)$ 的高 8 位，$R_{nL8}(x)$ 是 $R_n(x)$ 的低 8 位。

将式(3-5)代入式(3-4)，经整理得到：

$$\frac{B(x).2^{16}}{g(x)}=Q_n(x).2^{8n}+\{\frac{[R_{nH8}(x)+B_{n-1}(x)].2^{16}}{g(x)}+\frac{R_{nL8}(x).2^{8}}{g(x)}\}2^{8(n-1)}+\cdots$$

$$+\frac{B_1(x).2^{16}}{g(x)}.2^{8}+\frac{B_0(x).2^{16}}{g(x)} \qquad \text{式(3-6)}$$

再设 $$\frac{[R_{nH8}(x)+B_{n-1}(x)].2^{16}}{g(x)}+\frac{R_{nL8}(x).2^{8}}{g(x)}=Q_{n-1}(x)+\frac{R_{n-1}(x)}{g(x)} \qquad \text{式(3-7)}$$

将式(3-7)代入式(3-6)，如此类推，最后得到：

$$\frac{B(x).2^{16}}{g(x)}=Q_n(x).2^{8n}+Q_{n-1}(x).2^{8(n-1)}+\cdots Q_0(x)+\frac{R_0(x)}{g(x)} \qquad \text{式(3-8)}$$

根据 CRC 的定义，很显然，16 位二进制余数 $R_0(x)$ 即是我们要求的 CRC 码，式(3-7)是编程计算 CRC 的关键，它说明计算本字节后的 CRC 码时等于上一字节余式 CRC 的低 8 位左移 8 位后，再加上上一字节右移 8 位和本字节之和后求得的 CRC 码。如果把 8 位二进制序列数的 CRC 码全部计算出来，放在一个表里，采用余式表法，可以大大提高计算速度。

同理，对于一个二进制序列也可以按照半字节的方式表示为：

$$B(x)=B_n(x).2^{4n}+B_{n-1}(x).2^{4(n-1)}+\cdots+B_1(x).2^{4}+B_0(x) \qquad \text{式(3-9)}$$

按照一个字节的方法推论，可以得出结论：计算本字节后的 CRC 码等于上一字节余式 CRC 的低 12 位左移 4 位后，再加上上一字节右移 4 位和本字节之后求得的 CRC 码。如果把 4 位二进制序列数的 CRC 码全部计算出来，放在一个表里，采用半字节的余式表法，可以大大的提高计算速度同时还可以减少存放余式表的空间。

### 3.2.2 改进型 CRC-16 在 VxWorks 下的实现

对发送的数据帧，先按照半字节余式表法计算 CRC 校验码，然后按列计算奇偶校验码，奇校验按每行 1 个字节计算。假设数据帧长度为 len 个字节，存放的首地址为 ptr；crc 存放 crc 计算后的结果；odd 存放奇校验的结果；crc_odd 存放改进型 crc 的结果；则按照半字节查余式表的方法，在 VxWorks 下的实现改进型 CRC-16 实现流程如图 3.1 所示。

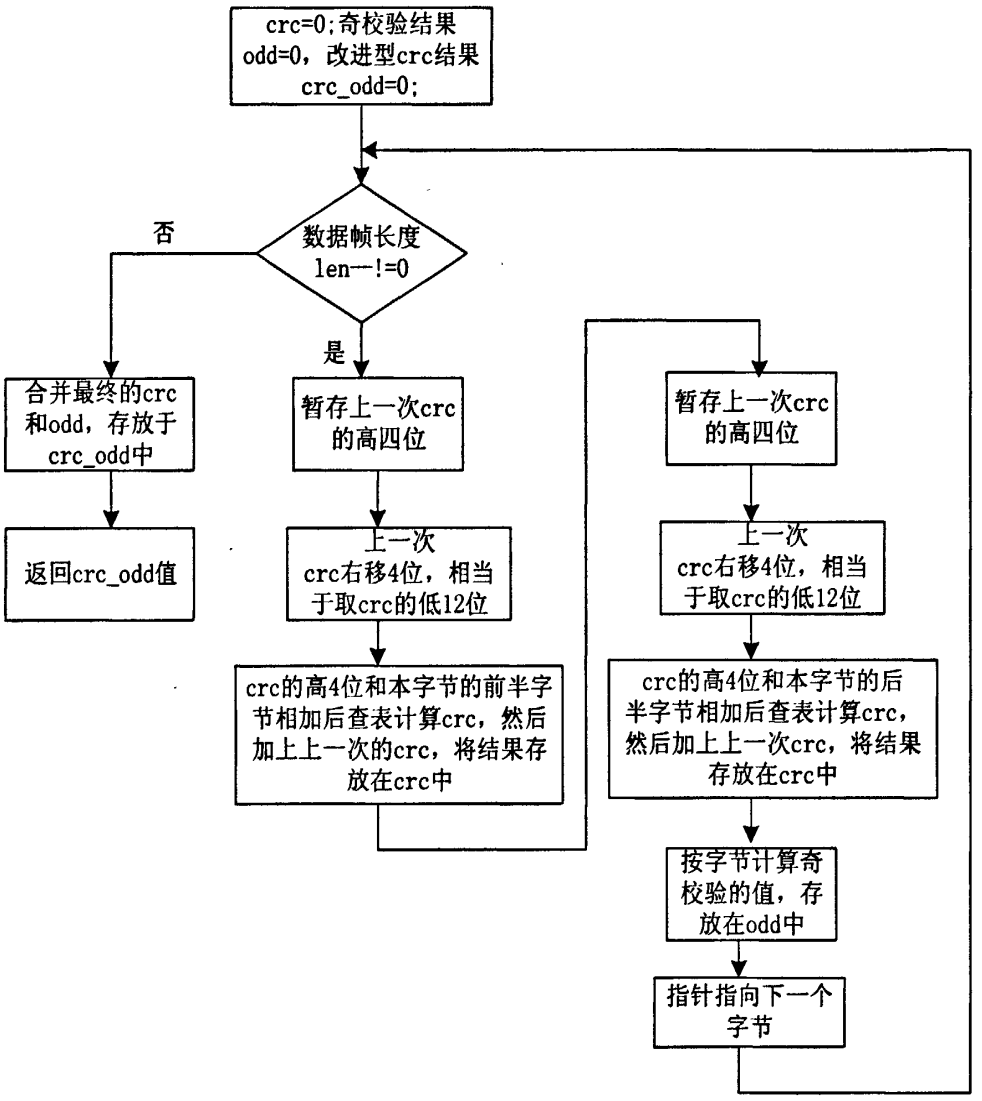


图 3.1 改进型 CRC-16 实现流程

## 3.3 改进型 CRC 在 labview 下的仿真

在 labview 环境下编程实现数字信号的 CRC 校验（假设被检测出的错误信号都能被纠正），发送如图 3.2 所示的加 CRC 校验之后的数据帧，加入信噪比为 5 的高斯噪声后的信号如图 3.3 所示，接收并判决后的波形如图 3.4 所示，对比图 3.2 和图 3.4 可发现，会有出错的信号。将该帧重复发送 500 次，分别计算信噪比不同时，未校验时的平均误帧率、CRC 校验后的平均误帧率和改进型 CRC 后的平均误帧率如表 3.2 所示。

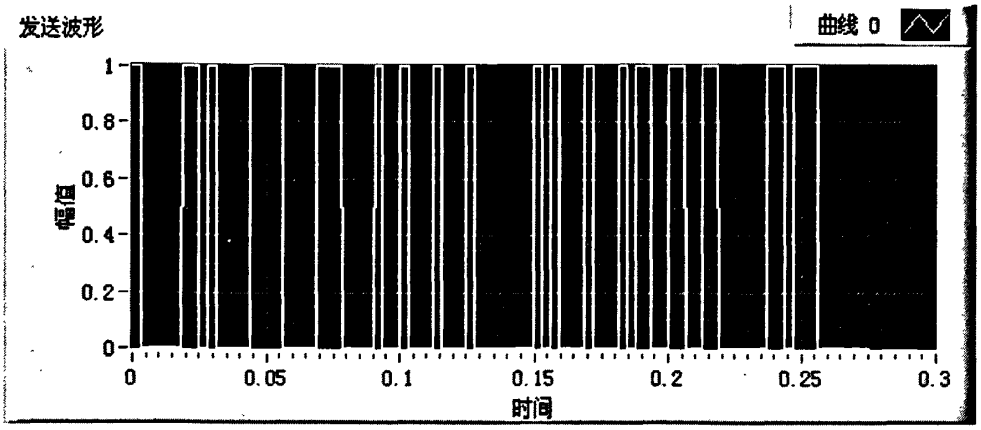


图 3.2 发送的源数字信号的波形（加入 CRC 校验码）

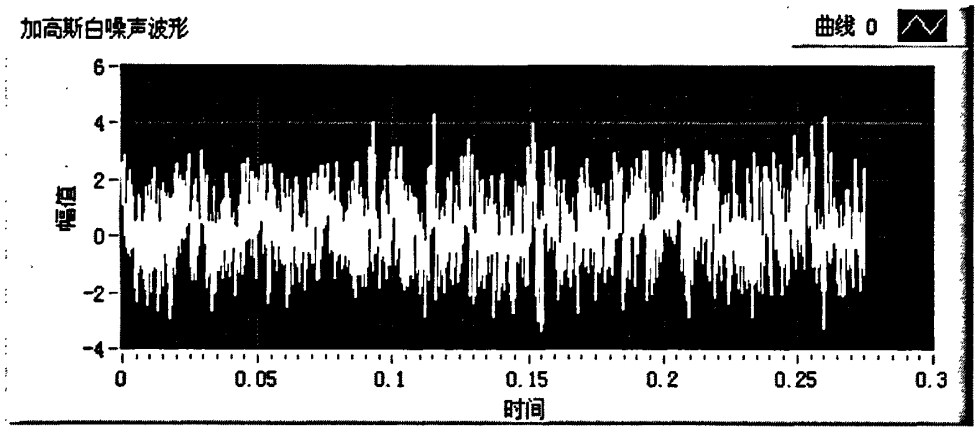


图 3.3 加入高斯白噪声后的数字信号的波形

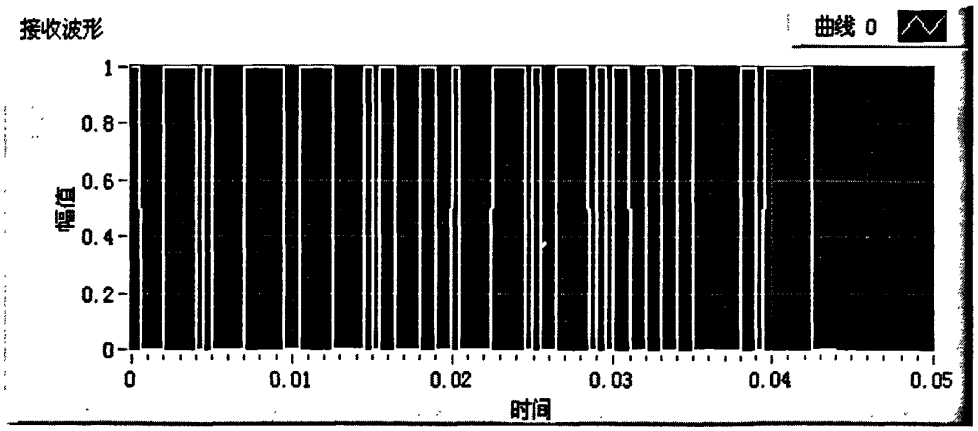


图 3.4 接收后未校验时的波形

如表 3.1 所示，随着信噪比的增加，误帧率逐渐下降。表中信噪比为 3 和 4 时，未校验时的平均误帧率为 1，表示误码率太高，导致每帧数据都会出错。对比 CRC 校验后的平均误帧率和改进型 CRC 校验后的平均误帧率，显然改进型 CRC 的平均误帧率得到进一步的减小。

表 3.1 误帧率对比表

| 误码率 / 信噪比（db） | 未校验时的平均误帧率 | CRC 校验后的平均误帧率 | 改进型 CRC 校验后的平均误帧率 |
|---|---|---|---|
| 4 | 1 | 0.148 | 0.142 |
| 5 | 0.962 | 0.266 | 0.260 |
| 6 | 0.8 | 0.322 | 0.32 |
| 7 | 0.42 | 0.24 | 0.2 |
| 8 | 0.22 | 0.128 | 0.124 |
| 9 | 0.068 | 0.032 | 0.03 |
| 10 | 0.012 | 0.01 | 0.01 |
| 11 | 0.002 | 0.002 | 0.002 |

## 3.4 本章小结

本章对经典 CRC-16 校验码进行了改进，并对其进行了性能分析。首先对经典 CRC 校验的原理进行了简单 的介绍，并对 CRC 校验进行了算法分析，得出了 CRC 的检错范围。在此基础上，对 CRC-16 做出了改进，并用余式表法在 VxWorks 下实现改进型的 CRC-16，最后，用 labview 软件对改进后的 CRC-16 和经典 CRC-16 进行了性能比较，得出结论：改进后的 CRC-16 的检错能力在经典的 CRC-16 基础上稍微有所提高。

# 第四章 基于 Markov 链的 GBN-ARQ 性能优化分析

## 4.1 三种基本的 ARQ

在数据通信中，常用的 ARQ 技术有三种类型：停止-等待式（SW-ARQ）、返回-n 式（GBN-ARQ）和选择式（SR-ARQ），后两种类型 ARQ 是滑动窗口技术的两个特例[23]。

1、停止-等待式 ARQ（SW-ARQ）

SW-ARQ（如图 4.1 所示）数据传输系统的发送端发送一个数据帧之后，启动定时器，开始倒计时，同时等待接收端的确认应答（ACK）。在等待过程中，停止发送新的数据帧。如果收到确认应答，则继续发送新的数据帧。若等待的一定时间没有收到 ACK 或者收到 NACK，则重新发送该数据帧，直到收到 ACK 这个时间为止。发送端等待的时间应至少大于数据发送到接收到 ACK 的时间，在实际应用中，等待时间是 2-3 倍。SW-ARQ 实现方式简单且在许多数据通信系统中得到了应用。但是，SW-ARQ 的缺点是发送完一帧数据后需要较长的等待时间导致低的数据传输速度太低。在低速传输时，对信道的利用率比较好，但是在告诉传输时，信道的利用率就会显著下降，即实时性太差。

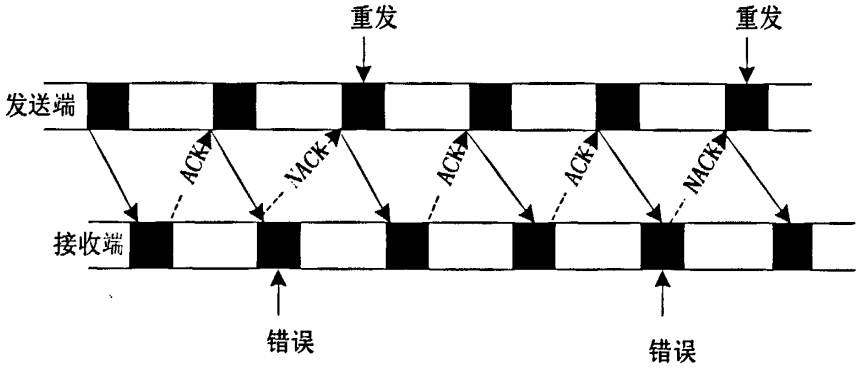


图 4.1 停止-等待式 ARQ

SW-ARQ 协议中，若接收端发送的 ACK 丢失了，超时之后，发送端仍旧重发然来的帧，接收端就会受到两个相同的数据帧，即出现重帧现象。为了避免重帧，一种很显然的做法是，让发送方在它所发送的每一帧的头部放上一个序列号。然后接收端可以检查它所接收到的每一帧的序列号，看它是新帧还是要丢弃的重复帧。

通信系统中通过允许多个数据帧的同时传输来提高信道的利用率。这种类型的 ARQ 也被称为连续型 ARQ。连续 ARQ 包括两种：GBN-ARQ 和 SR-ARQ。

2、返回-n 式 ARQ（GBN-ARQ）

GBN-ARQ 采用滑动窗口协议。所有滑动串口协议的本质是，在任何时刻，发送方总是维持着一组序列号，分别对应于允许它发送的帧。称这些帧落在发送窗口之内，接收方也维持着一个接收窗口，对应于一组允许它接收的帧。发送窗口和接收方的窗口不必有同样的上下界，甚至不必有同样的大小。GBN-ARQ 的具体实现如图 4.2 所示，发送端连续按序号发送数据帧并且把发送的数据帧存储起来等待接收每帧的 ACK 信号。在等待每个帧的 ACK 信号期间，又可以发送后面的 N-1 帧。如果数据帧 i 的应答帧超时或者收到 NACK，则停止发送新的数据帧，重新发送数据帧 i 和在它后面已经发送的其它 N-1 帧数据。接收端抛弃了数据帧 i 和它后面无论错误与否的 N-1 帧数据。如此重发直到收到数据帧 i 的 ACK 为止。

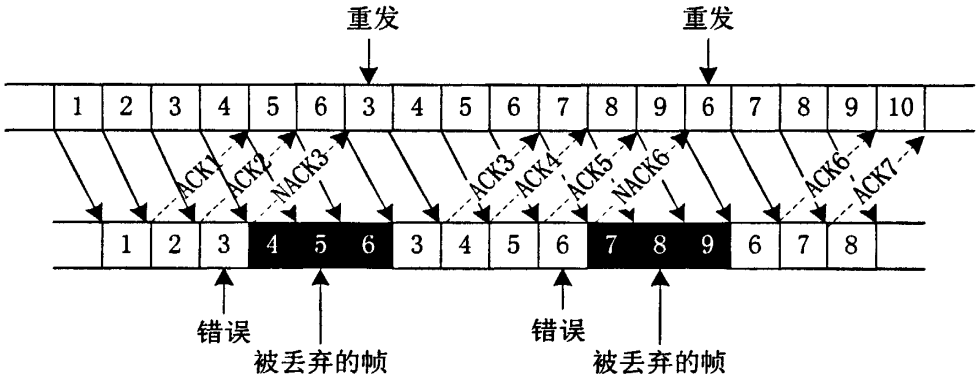


图 4.2 返回-n 式 ARQ

与 SW-ARQ 相比较，GBN-ARQ 效率要高得多。缺点是在某个数据帧 i 出错以后，接收端就抛弃该帧以及该帧后已经收到的所有数据帧，即使它们是正确的。这就导致在误码率比较高的环境中，系统的吞吐率迅速下降。

3、选择式 ARQ（SR-ARQ）

为了解决 GBN-ARQ 中吞吐率低的问题，当某一帧数据出错时，设法只重新发送出错的数据帧，而该帧后接收正确的数据帧被接收端存储起来，这种策略就是 SR-ARQ。在 SR-ARQ 中，发送端不再重新发送出错帧后已经发送的正确帧，省下来的时间用来传送新的数据帧。SR-ARQ 的具体实现过程如图 4.3 所示。

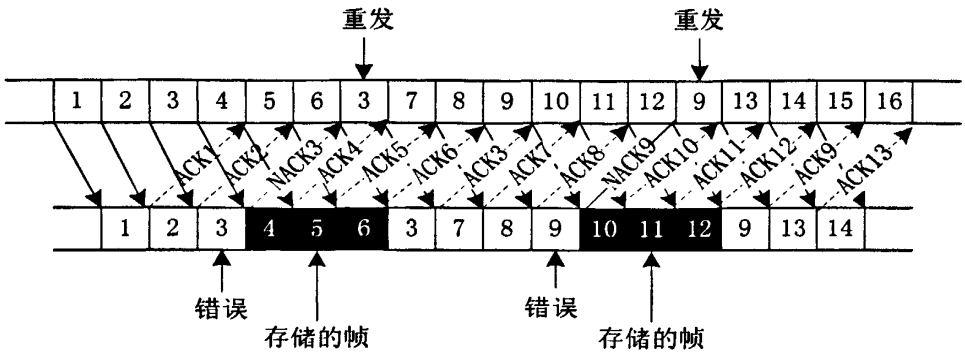


图 4.3 选择式 ARQ

与 GBN-ARQ 相比，SR-ARQ 的吞吐量明显得到提高。但是由于存储那些被检测出错误的数据帧正确的数据帧，SR-ARQ 的接收端必须要提供一个足够大容量的存储器，以保证不会有溢出。如图 5-2 和 5-3 所示，GBN-ARQ 方式的接收端只能按顺序接收，而 SR-ARQ 则可以乱序接收。

综上所述，在这三种 ARQ 机制中，（GBN-ARQ）由于具有比停-等式 ARQ（SW-ARQ）较高的系统效率以及比选择式 ARQ（SR-ARQ）较低的系统实现复杂度。实时系统的差错控制技术的关键在于在不破坏系统的可靠性是前提下，提高的系统的实时性，即尽量减少数据传输的时延。但是，已有的大量参考文献中对于 ARQ 主要是以提高吞吐量位目的，重点研究如何对系统传输进行有效的控制及相应算法的改进。

## 4.2 GBN-ARQ 的 Markov 描述

在 GBN-ARQ 差错控制系统中，假设系统是平稳的，接收端接收数据帧有三种可能事件：A 表示正确接收；B 表示出现可检错误；C 表示出现不可检错误。用 Pa、Pb、Pc 分别表示这三种事件的概率，则有 Pa+Pb+Pc=1.用 X 和 Y 分别表示接收端的回执信号为“ACK”和“NACK”的事件，Px 和 Py 表示各自出现的概率。显然 Pa+Pb+Pc=Px+Py=1.当发送端发送一个数据帧且收到该帧的回执信号，有且仅有事件 AX、AY、BX、BY、CX、CY 六个事件之一发生。其中每个事件所表示的意思[24]为：

(1)F1（AX）事件发生：发送的数据帧没有出错且被接收端正确接收，同时回执信号 ACK。ACK 信号通过反向信道也被发送端正确接收，系统接下来发送新的数据帧。

(2)F2（AY）事件发生：发送的数据帧没有出错且被接收端正确接收，同时回执信号 ACK。但是回执信号 ACK 在反向信道上畸变位 NACK，发送端误认为数据帧出错而重新发送该数据帧和后继的 N-1 帧数据。这一事件出现后带来加字和正确接收两种可能，到底是哪一种还要取决于后继出现的事件。

(3)F3（BX）事件发生：发送的数据帧出现“可检错误”，且被接收端拒绝接收，同时回执信号 NACK。NACK 信号通过反向信道被发送端正确接收，发送端重新发送该数据帧和后继的 N-1 帧数据。这是正常事件。

(4)F4（BY）事件发生：发送的数据帧出现“可检错误”，且被接收端拒绝接收，同时回执信号 NACK。但是回执信号 NACK 在反向信道上畸变位 ACK，系统接下来发送新的数据帧，结果丢掉 N 帧数据，即出现“漏字”。

(5)F5（CX）事件发生：发送的数据帧出现“不可检错误”，接收端错误地接收，并返回 ACK。ACK 信号通过反向信道也被发送端正确接收，系统接下来发送新的

数据帧，结果造成系统误字。

(6)F6（CY）事件发生：发送的数据帧出现“不可检错误”，但是接收端正确的接收了该帧，并返回 ACK。ACK 在反向信道中传输过程中畸变位 NACK，使发送端重新发送该帧和后后继的 N-1 帧数据。这一事件的发生会带来加字和错误接收两种可能，到底是哪一种，取决于后继出现的事件。

分别用 f1=PaPx，f2=PaPy,f3=PbPx,f4=PbPy,f5=PcPx,f6=PcPy 表示以上六种独立事件发生的概率。设随机变量 x∈{0，1，2，3，4，5，6}表示传送某一帧时系统所处的状态。其中 x=0 时表示系统发送一个新的数据帧；x=1，2，3，4，5，6 分别表示系统发生事件 F1，F2，F3，F4，F5，F6。x=2，3，6 之一时都将引起重发数据帧；x=1，4，5 系统要发送新的数据帧。

经过进一步的分析，系统的整个工作过程可用一组齐次 Markov 链{Xi（n)；n=i,i+N,i+2N...}（i=0，1，2，...，N-1，N 为环路延迟时间内的帧数）来描述。这 N 个 Markov 链的状态转移图都相同，如图 4.4 所示。

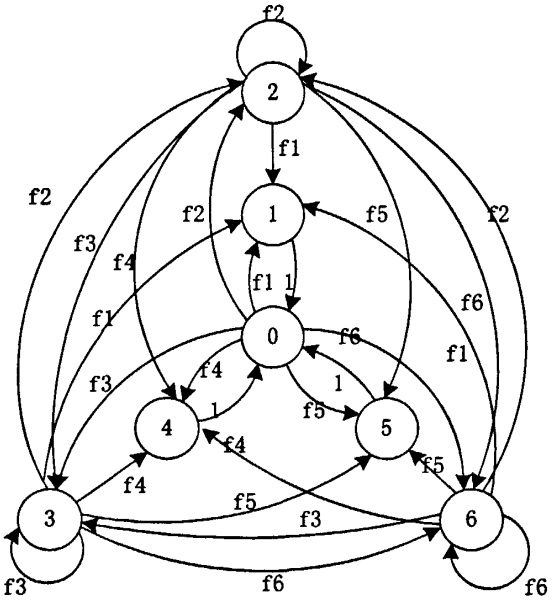


图 4.4 GBN-ARQ 的状态转移图

该齐次 Markov 链的一步转移矩阵为：

$$P=\begin{bmatrix} 0 & f1 & f2 & f3 & f4 & f5 & f6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & f1 & f2 & f3 & f4 & f5 & f6 \\ 0 & f1 & f2 & f3 & f4 & f5 & f6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & f1 & f2 & f3 & f4 & f5 & f6 \end{bmatrix}$$

## 4.3 GBN-ARQ 的性能指标

### 4.3.1 GBN-ARQ 的有效传输效率

有效传输效率定义为接收端正确接收到的数据比特数与发送端在同一时间内发出的总比特数之比[25]。设 GBN-ARQ 系统的平稳状态概率为 $\pi=(\pi_0,\pi_1,\pi_2,\cdots\pi_6)$，则根据 $\pi$ 的性质有：

$$\begin{cases}\pi P=\pi\\ \pi_0+\pi_1+\pi_2+\pi_3+\pi_4+\pi_5+\pi_6=1\end{cases}$$

求解可得：

$$\pi_0=(f1+f4+f5)/(1+f1+f4+f5)$$

$$\pi_i=fi/(1+f1+f4+f5)$$

其中 $i=1,2,3,4,5,6$

$\pi_0$ 是系统重新发送新的数据帧的概率，$\pi_1$ 是系统正确收发的概率，则每一帧数据重发的概率为 1-$\pi_0$，该数据帧本身的重传次数为 $\frac{1-\pi_0}{\pi_1}$。在此系统中发送端每重发一帧数据时，必然重发后继的 N-1 个数据帧。因此接收端每收到一帧数据时，发送端需要重传的所有数据帧的平均重传次数为：

$$M=\frac{N(1-\pi_0)}{\pi_1}-(N-1)$$

$$=\frac{N}{f1}-(N-1)$$

$$=\frac{N}{PaPx}-(N-1)$$

设数据帧总长度为 L，帧的数据位长度为 n，其它的开销为 H，则 GBN-ARQ 系统的传输效率为：

$$\eta=\frac{n}{ML}=\frac{PaPx}{N-NPaPx+PaPx}(\frac{n}{L}) \qquad \text{式(4-1)}$$

式(4-1)中若考虑反向信道是理想的，则 Px=1，则传输效率为:

$$\eta=\frac{n}{ML}=\frac{Pa}{N-NPa+Pa}(\frac{n}{L}) \qquad \text{式(4-2)}$$

4.3.2 GBN-ARQ 的传输时延

设$t_f$使一个数据帧的发送时间，且数据帧的长度是固定不变的，则数据帧的发送时间$t_f$是数据帧的长度 L(bit)与数据的发送速率 R(bit/s)之比，即

$$t_f = L/R \qquad \text{式(4-3)}$$

在 GBN-ARQ 系统中，正确传送一个数据帧所需的平均时间$t_{av}$是：

$t_{av}=t_f+t_T\times$该帧的平均重传次数

$$=t_f+t_T\times\frac{1-\pi_0}{\pi_1}$$

$$=t_f+\frac{t_T}{PaPx} \qquad \text{式(4-4)}$$

其中$t_T$为成功发送一个数据帧所需要的时间，设$t_{out}$为数据帧重传的等待时间，则$t_T$可表示为：

$$t_T=t_f+t_{out}=t_f+t_{p1}+t_a+t_{p2}+t_{pr}=t_f+t_a+2t_{pr}+t_{p1}+t_{p2} \qquad \text{式(4-5)}$$

式(4-5)中，$t_{p1}$为发送数据帧的传播时延；$t_{p2}$为应答帧的传播时延；$t_a$为应答帧的发送时间；$t_{pr}$为节的处理时间。 通常情况下，应答帧的发送时间$t_a$和处理时间$t_{pr}$都远小于传播时延，这样就可以简单地将重传时延取为$t_{p1}+t_{p2}$，设信道的传输速度为 W，则

$$t_T=t_f+\frac{L+H}{W} \qquad \text{式(4-6)}$$

将式(4-3)和式(4-6)代入式(4-4)可得：

$$t_{av}=\frac{L}{R}+\frac{\frac{L}{R}+\frac{L+H}{W}}{PaPx}$$

$$=\frac{L}{R}+\frac{LW+(L+H)R}{PaPxRW} \qquad \text{式(4-7)}$$

设有一组总长度为 C(bit)的数据 data 需要发送，则数据 data 的平均发送时间为 Tc，Tc 有以下关系式：

$$\text{Tc}=t_{av}\times\frac{C}{L} \qquad \text{式(4-8)}$$

把式(4-7)代入式(4-8)可得：

$$\text{Tc}=\frac{C}{R}+\frac{WLC+R(L+H)C}{PaPxRLW} \qquad \text{式(4-9)}$$

## 4.4 数值分析和仿真

### 4.4.1 有效传输效率分析

在 GBN-ARQ 系统中，数据的传输效率$\eta$反映了系统的可靠性。为了分析信道误码率,帧长度和传输效率的关系，设信道的误码率为$P_e$，则有

$$Pa=(1-P_e)^L \qquad \text{式(4-10)}$$

应答帧一般无数据位，可使应答帧的长度近似为开销位的长度 H，则有

$$Px=(1-P_e)^H \qquad \text{式(4-11)}$$

将式(4-10)和式(4-11)代入式(4-1)可得：

$$\eta=\frac{(1-P_e)^{(L+H)}}{N+(1-N)(1-P_e)^{(L+H)}}\left(\frac{H-L}{L}\right) \qquad \text{式(4-12)}$$

根据式(4-12)，在 GBN-ARQ 系统中，假设 N=7，H=16，信道误码率、帧长度 L 对传输效率的影响如图 4.5 和表 4.1 所示，可以得出以下结论：

(1)在误码率低的信道中，从图 4.5 中的 Pe=0.000001 曲线可以得出随着帧长度的增加，GBN-ARQ 系统的有效传输效率是随着帧长度 L 的增加而增加的，即误码率低的环境下，系统重传次数很少，增加帧长度可以提高系统的有效传输效率。

(2)表 4.1 对比帧长度固定为 100 时系统的误码率和传输效率的关系，随着误码率的下降，GBN-ARQ 系统的传输效率急剧的下降，当误码率增加到一定程度时，每帧数据的重传次数不断提高，系统的传输效率几乎为 0，即几乎不能传送有效数据。

(3)从图 4.5 中的 Pe=0.01 的曲线可以得出，在误码率高的信道中，随着帧长度的增加，系统的传输效率增加到一定程度后开始急剧的下降，在误码率高的环境下，为了保证传输效率存在一个最佳帧长度。

表 4.1 帧长度固定为 100 时，误码率和传输效率的关系

| 误码率 Pe | 0.000001 | 0.0001 | 0.001 | 0.01 |
|---|---|---|---|---|
| 传输效率 | 0.8266 | 0.7693 | 0. 4455 | 0.0530 |

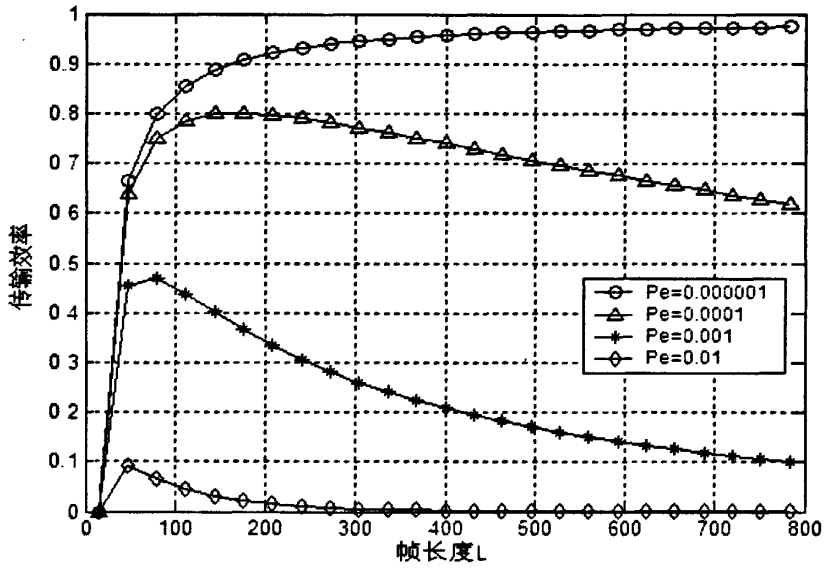


图 4.5 传输效率随误码率 Pe 和帧长度 L 的变化曲线

### 4.4.2 传输时延分析

同样为了分析信道误码率、帧长度和传输时延的关系，将式(4-10)和式(4-11)代入式(4-7)可得在 GBN-ARQ 系统中，传输总长度为 C 的数据的时延为：

$$Tc = \frac{C}{R} + \frac{WLC + (H+L)RC}{(1-P_e)^{(H+L)}RLW} \qquad \text{式(4-13)}$$

根据式(4-13)，在 GBN-ARQ 系统中，假设系统要传送数据总长度 C=120000(bit)，发送速率 R 和传输速率 W 都为 115200(bit/s)，可得出如下结论：

(1)在误码率固定为 Pe=0.001 时，传输时延随帧长度的变化曲线如图 4.6 所示，显然，在噪声信道中，存在一个最佳帧长使得传输延迟最短。

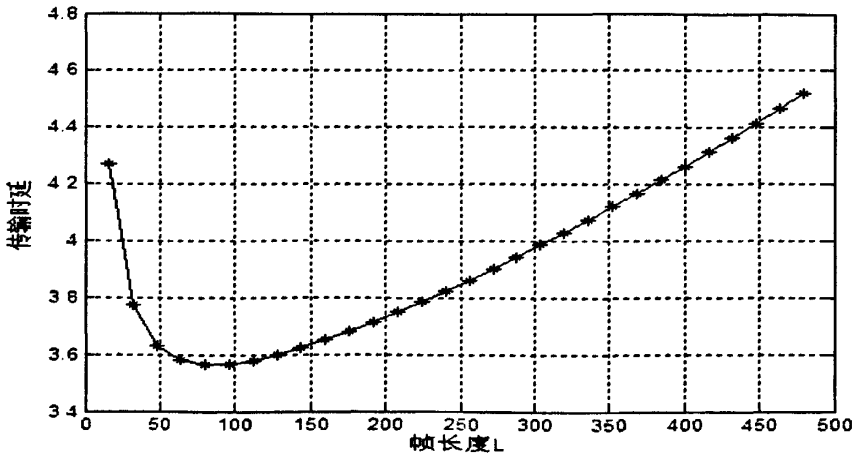


图 4.6 误码率 Pe=0.001 传输时延随帧长度的变化曲线

(2)在误码率固定为 Pe=0.001 的噪声信道中，如图 4.7 所示，额外开销增大后，数据帧长也随之增大，传输时延也随之变长。提取不同额外开销时，传输时延最短所对应的最佳帧长如表 4.2 所示，可看出最佳帧长与额外开销位数成正比，这是因为，为了提高效率必须使数据帧中有效位数随着额外开销位数正比的变化。

表 4.2 额外开销不同时，Tc 所对应的最佳帧长值

| 额外开销 H(bit) | 48 | 32 | 16 |
|---|---|---|---|
| 最佳帧长 L(bit) | 134 | 108 | 64 |

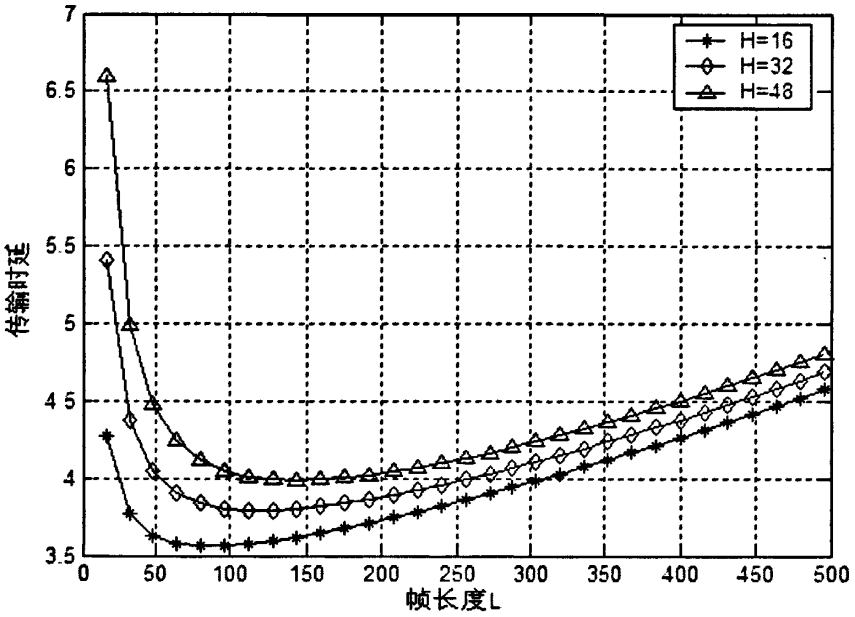


图 4.7 所示 传输时延随额外开销和帧长度的变化曲线

(3)在噪声信道中，如图 4.8 所示，传输时延随着误码率的增加，不断增加。提取不同误码率时传输时延最短所对应的最佳帧长如表 4.3 所示，最佳帧长与信道的误码率成反比，随着信道误码率的增加，每一帧出错的概率也随即增大，使得重传次数增多，因此必须使帧中的有效位数减小，才能不使实际传送时间增大。

表 4.3 误码率不同时，Tc 最小所对应的最佳帧长

| 误码率 Pe | 0.001 | 0.05 | 0.01 |
|---|---|---|---|
| 最佳帧长 L(bit) | 55 | 40 | 24 |

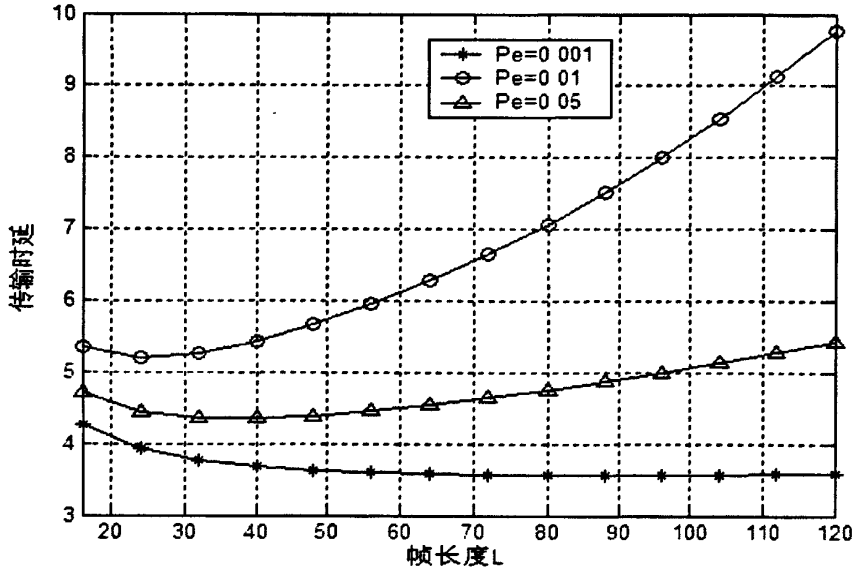


图 4.8 传输时延随误码率和帧长度的变化曲线

(4)在无误码率的理想信道中，如图 4.9 所示，数据帧几乎不需要重发，帧长度越大，则传输延迟越短。

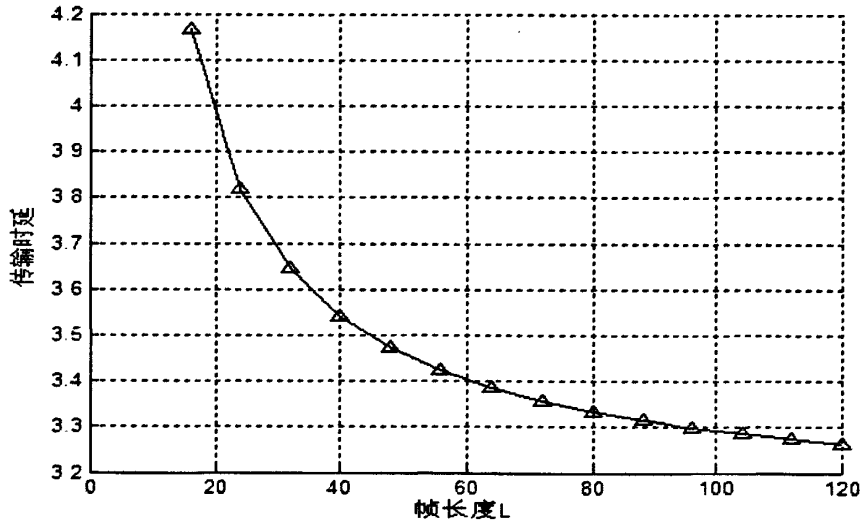


图 4.9 理想信道下，传输时延随帧长的变化曲线

### 4.4.3 最佳帧长分析

在 4.4.1 节中对误码率 Pe 和帧长 L 在 GBN-ARQ 系统中对传输效率$\eta$的影响进行了分析，得出结论：在噪声信道下存在一个最佳帧长使得传输效率最高；同时，在 4.4.2 节中又对误码率 Pe 和帧长 L 对传输时延 Tc 的影响进行了分析，得出结论：在噪声信道下也存在一个最佳帧长使得传输时延达到最低。那么是否存在

一个最佳帧长值同时满足传输效率$\eta$最大和传输延时 Tc 最小的条件呢？

设 $\lambda=\eta/Tc$ 式(4-14)

式(4-14)中，如果存在帧长度使得$\lambda$取得最大值，就能同时满足$\eta$取最大值，Tc 取最小值，即这个帧长度就是最佳帧长度。将式(4-12)和式(4-13)代入(4-14)可得：

$$\lambda=\frac{(1-P_e)^{(L+H)}}{N+(1-N)(1-P_e)^{(L+H)}}\left(\frac{H-L}{L}\right)/\left(\frac{C}{R}+\frac{WLC+(H+L)RC}{(1-P)^{(H+L)}RLW}\right) \quad \text{式(4-15)}$$

同样，设定传送数据总长度 C=120000(bit)；发送速率 R 和传输速率 W 都为 115200(bit/s)；N=7；H=16，根据式(4-15)可得出以下结论：

(1)如图 4.10 所示，在无噪声的理想信道中，即 Pe=0 时，传输效率和传输时延$\lambda$的值随着帧长度 L 的增大到最大后，就几乎不再发生变化，趋于直线。

(2)在误码率固定为 Pe=0.001 时，$\lambda$随帧长度 L 的变化曲线如图 4.11 所示，显然，在噪声信道中，存在一个最佳帧长使得$\lambda$最大。

(3)如图 4.12 所示，是在不同误码率条件下，$\lambda$随帧长度 L 的变化曲线。取不同误码率条件下，$\lambda$取最大值时所对应的最佳帧长度如表 4.4 所示，可得出误码率越大，最佳帧长度越小，即误码率和最佳帧长度成反比。

表 4.4 误码率不同时，$\lambda$所对应的最佳帧长度

| 误码率 Pe | 0.001 | 0.05 | 0.01 |
|---|---|---|---|
| 最佳帧长 L(bit) | 55 | 40 | 24 |

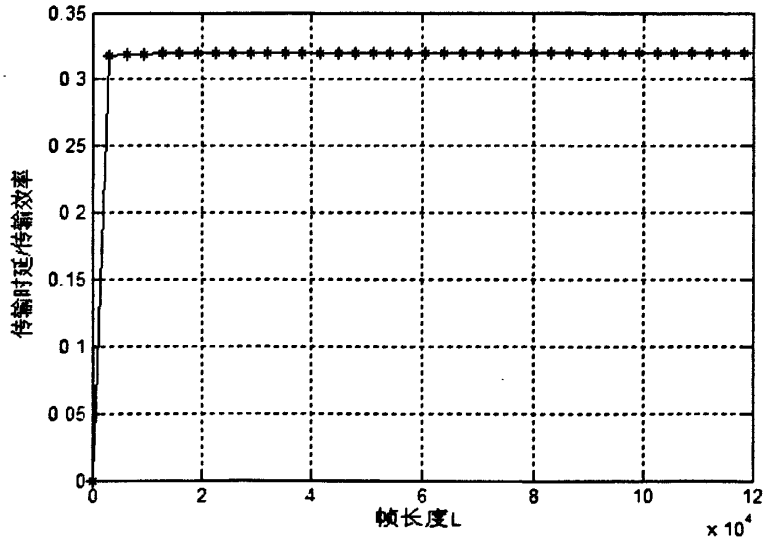


图 4.10 理想信道中$\lambda$随帧长度的变化曲线

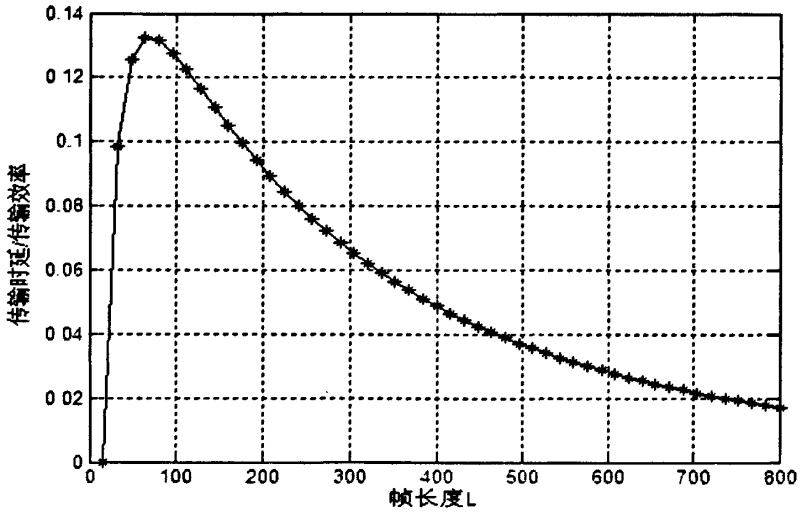


图 4.11 噪声信道下，$\lambda$ 的值随帧长度的变化曲线

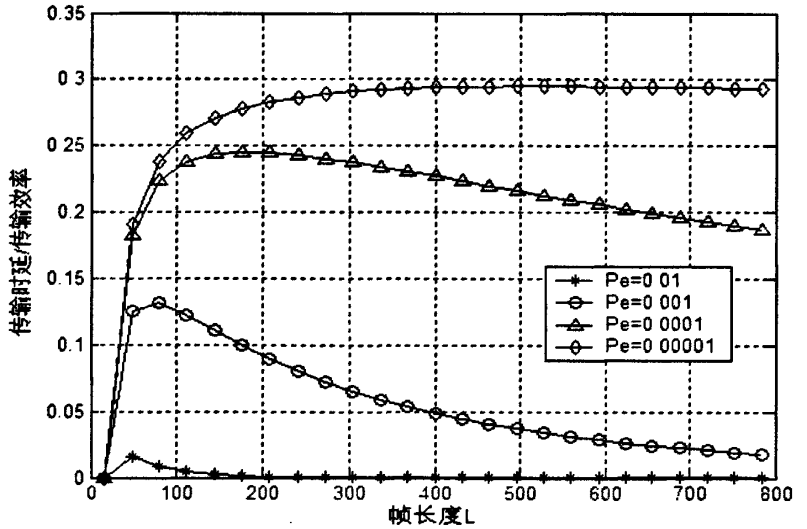


图 4.12 不同误码率条件下，$\lambda$ 随帧长度 L 的变化曲线

## 4.5 本章小结

本章基于 Markov 链对 GBN-ARQ 做了性能分析。首先介绍了三种基本的 ARQ，在三种 ARQ 中，GBN-ARQ 具有比 SW-ARQ 较高的系统效率以及比 SR-ARQ 较低的系统实现复杂度。在此基础上，用 Markov 链描述了 GBN-ARQ 的工作过程，把 GBN-ARQ 的工作过程分为七个状态，画出了状态转移图，从而方便的计算出了 GBN-ARQ 系统的有效传输效率和传输时延。最后，在 matlab 上分别对各种环境下的传输效率和传输时延进行了分析，得出存在一个一个最佳帧长度使得系统的传输效率最大的同时，系统的传输时延可以降到最小。

# 第五章 基于 VxWorks 的串行通信系统的设计与实现

针对某系统设计了一个基于 VxWorks 的串行通信系统，实现了与上位机(PSP)和固态功率控制器（SSPC）的通信功能。

## 5.1 串行通信系统的总体设计

### 5.1.1 串行通信系统的功能分析

在进行嵌入式系统开发时，一般过程是：首先明确系统的功能，然后结合实时操作系统的特性，对应用软件要实现的任务进行大小适当的划分，即把应用软件的功能按照一定的原则划分为若干个任务模块，并对各个任务间的通信和时延进行仔细的确认。

本课题中，VxWorks 操作系统底层平台为系统实现提供了底层软件支持，串行通信系统的功能需要应用程序来实现，而应用软件开发首要目标就是明确应用软件需要实现的功能。基于 VxWorks 的串行通信系统的主要任务通过 RS485 总线接收来自上位机（PSP）的控制信息，对二次配电中心的分布式电力汇流条和固态功率控制器（SSPC）进行监测与控制，并负责把检测并处理完的信息通过 ARINC429 总线送到 PSP 中。串行通信系统具体实现如下功能：

(1)通过 RS485 总线与 PSP 通信，接收上位机（PSP）的控制命令；

(2)向上位机（PSP）传送采集到得模拟量、离散量以及 SSPC 的状态电流值；

(3)向上位机（PSP）传送存储在 Flash 中的故障信息和工作日志数据；

(4)与 SSPC 进行通信，控制负载的接通关断，接收 SSPC 的状态电流值。

### 5.1.2 串行通信系统中任务的划分

在基于嵌入式操作系统的应用软件设计中，任务是一个很重要的概念。所谓的“任务”就是能够完成一个或者一组相关功能的一个或者一组程序的组合，它的概念与操作系统的进程概念相同，一个任务是独立的执行进程，可以与其它的并发任务竞争 CPU 的时间。在开发过程中，任务是一个无限循环的程序。任务不能有返回，不能有退出出口，但是任务可以被销毁，包括被别的任务销毁或自销毁。

上一节已经对串行通信系统的功能进行了分析，下面将对系统进行任务的划分、建立以及优先级的分配。任务划分一方面是为了显示系统对于所要处理问题的轻重缓急的对应策略，另一方面则是提高系统性能。合理的任务划分会大大改

善系统的性能，不合理的划分不仅不利于系统的进一步的设计，还会降低系统性能，甚至使得系统不能满足要求。任务划分的矛盾在于：任务划分过细会引起频繁的任务切换，从而导致任务的开销增加；而任务划分不彻底又会使得原本可以并行的操作只能按准许串行完成，从而导致系统的运行效率降低。因此，在任务划分之前，还需要考虑一下几点规则：

(1)周期相同的任务应分配于同一个任务中；

(2)每一种类型的算法过程应尽量建立单独的任务，这样可以避免程序代码重复，提高软件执行效率；

(3)系统中各紧密相关的功能应合成于一个任务中，这样可以减少任务间的通信开销，降低系统设计的难度；

(4)如果操作依赖于事件，应选择一个事件操作对应一个任务，因为此类任务的运行受限于事件而不是处理器。

依据上述的任务原则，串口通信系统的应用软件可以划分以下几个任务：

(1)通道检测任务（tComQuery）：依次查询与 SSPC 通信的各个 422 通道，判断通道是否有数据进来。若在轮询任务中发现某个串口缓冲区有数据到来，则触发相应任务执行。若发现缓冲区为空，则继续判断下一个串口缓冲区。如此反复，周期执行。

(2)与上位机通信的任务（tPCDeal）：检测与 PSP 通信的 485 通道是否有数据进来。若有数据来，则从 485 缓冲区取出数据，首先判断校验和是否正确，若正确则判断信息类型，根据判断结果执行相应操作。任务完成后释放相应信号量使轮询任务运行，并等待信号量触发再次运行。

(3)50ms 周期任务（tSendTCCB）：周期性向各个 SSPC 阵列板发送供电请求命令，以获得某路负载的电流和状态值，保证各负载得到实时监控；周期性采集模拟量、离散量、汇流条，实现对汇流条的实时监控。

(4)SSPC 数据处理任务（tDealCCB）：将从 422 缓冲区收到的 SSPC 阵列板发送来的数据一一取出，首先判断校验和是否正确，若正确则判断属于何种信息，并执行相应操作。任务完成后释放相应信号量使轮询任务继续运行，并等待信号再次触发该任务运行。

(5)工作日志上传任务（tUpDlyLogn）：按照 GBN-ARQ 协议向上位机 PSP 上传工作日志数据。

(6)故障数据上传任务（tUpFault）：按照 GBN-ARQ 协议向上位机 PSP 上传故障数据。

任务划分完毕后，需要规定各个任务的优先级。任务的优先级是按照任务的重要性来划分的，保证重要任务先执行，上面各个任务的优先级如表 5.1 所示。

表 5.1 任务优先级分配表

| 任务名 | 任务优先级 | 任务属性 | 任务介绍 |
|---|---|---|---|
| tComQuery | 240 | 周期 | 通道检测任务 |
| tPCDeal | 210 | 周期 | 与上位机通信任务 |
| tSendTCCB | 220 | 周期 | 50ms 周期任务 |
| tUpDlyLogn | 210 | 非周期 | 工作日志上传任务 |
| tUpFault | 210 | 非周期 | 故障数据上传任务 |
| tDealCCB | 230 | 非周期 | SSPC 数据处理任务 |

Wind 内核提供了 256 个任务的优先级，优先级最高为 0，最低为 255。由于操作系统启动后，系统任务已在运行，为使用户任务不与系统任务发生冲突，用户任务的优先级一般最高为 150。多任务系统中各个任务之间通信与同步方式有很多种，该系统中采用了两种方式：全局变量、信号量。由于系统中各个任务频繁调用两个数据结构文件，综合考虑效率和空间，状态数据和控制数据均采用全局变量的形式来存储。

### 5.1.3 软件流程

基于 VxWorks 的串行通信系统包括 3 个周期任务和 3 个非周期任务。周期任务和非周期任务的不同点在于：周期任务有定时机制；非周期任务在任务开始时都设有等待信号量的操作，由于信号量在开始的时候设置为不可用，所以非周期任务在系统启动后被挂起，等待相关任务释放所需信号量，非周期任务才会运行。任务之间按基于优先级的抢占式任务调度策略来运行。

软件的主要流程如图 5.1 所示，系统启动以后，先进行一系列的初始化操作；初始化完毕后，创建信号量设置属性；开启辅助时钟中断；最后创建任务，启动多任务环境。进入多任务程序以后，如果辅助时钟 1ms 中断到达，则释放信号量 semId[2],系统按照优先级首先判断任务 tPCDeal、任务 tUpDlyLogn 和任务 tUpFault 是否可以执行，tUpDlyLogn 和任务 tUpFault 的信号量 semId[4]和 semId[6]为不可用则挂起，执行任务 tPCDeal。任务 tPCDeal 执行完毕后释放信号量 semId[4],则执行任务 tUpDlyLogn；释放信号量 semId[6]，则执行任务 tUpFault。如果辅助时钟 50ms 中断到达，释放信号量 semId[5]且任务 tSendTCCB 的优先级最高，则执行任务 tSendTCCB。如果当前优先级最高的任务是 tComQuery，系统将执行该任务并判断 422 串口缓冲区是否有数据到来，有则释放信号量 semId[0]唤醒任务 tDealCCB；没有或者任务 tDealCCB 执行完毕则释放信号量 semId[1]继续进行通道检测。

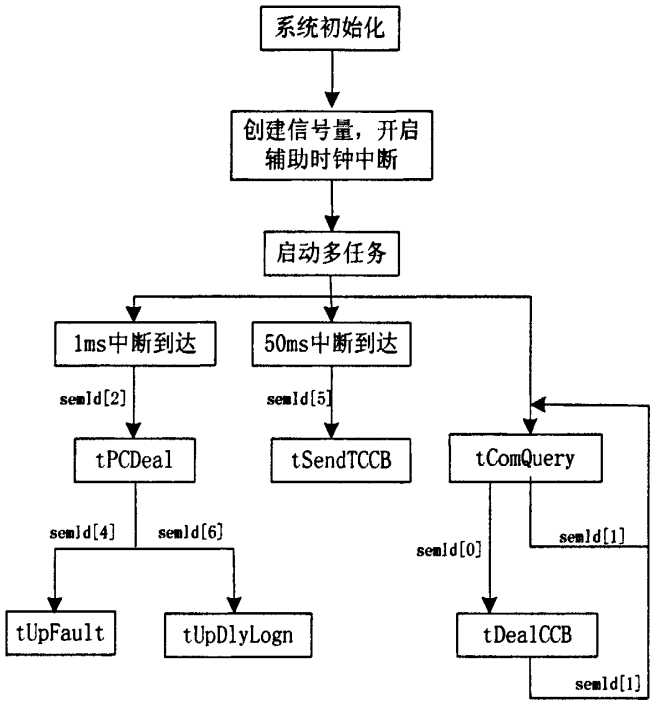


图 5.1 软件流程图

## 5.2 系统各个任务模块设计

系统软件主要包括主程序、各个任务模块和中断服务子程序。主程序主要完成初始化串口、开辟数据表内存、创建变量、信号量及激活任务等操作。在主程序运行完后，将开始多任务程序的运行，执行各个任务模块，辅助时钟中断服务子程序在本软件中主要是对周期任务进行定时，释放信号量激活相应的周期任务。以下将具体介绍软件的主要模块设计与实现。

### 5.2.1 主程序模块（Mainapp）

在系统启动后，程序将从主程序开始运行。主程序的流程如图 5.2 所示。

如图 5.2 所示，系统启动后，首先初始化 VxWorks 启动操作系统，然后对 AD 采集卡、串口通讯卡进行初始化，然后设定中断服务程序并创建任务，最后启动多任务程序，系统开始正常运行。

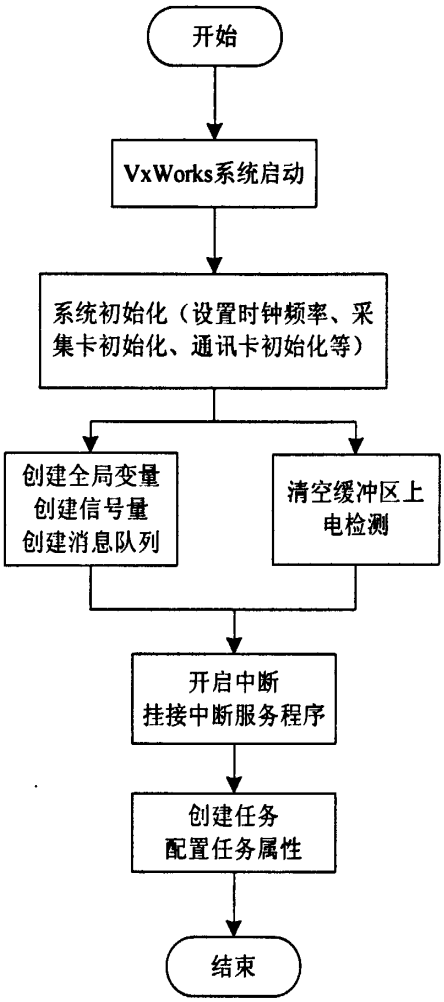


图 5.2 主程序流程图

### 5.2.2 通道检测任务（tComQuery）

该任务的目的是查看与 SSPC 通信的各个串口的环形缓冲区是否有数据到来。当有数据来时，采用信号量机制唤醒相应任务。从图 5.1 可以看出，任务启动时，首先调用函数判断环形缓冲区的字节数，依次对所有的通道进行检测，判断是否有数据到来，若有数据则释放信号量 semId[0]，否则释放信号量释放信号量 semId[1]继续检测。任务 tComQuery 是一个优先级最低的循环任务，从而保证 CPU 不会空闲，其他任务都不执行的时候，任务 tComQuery 会一直占用 CPU。

### 5.2.3 与上位机通信的任务（tPCDeal）

该任务主要完成统与上位机间的通讯工作，通过任务 tPCDeal 调用多个功能子函数实现。该任务主要包括两个方面：一是通过外部 RS485 总线接收来自上位机

的命令，并把有关 SSPC 监测与控制的要求进行处理后，按照与下位机的协议发送给 SSPC 阵列板；二是通过功能子函数的操作将处理后的数据按照与上位机间的协议上传给上位机；三是上传工作日志数据和故障数据信息。该任务的具体实现流程如图 5.3 所示。

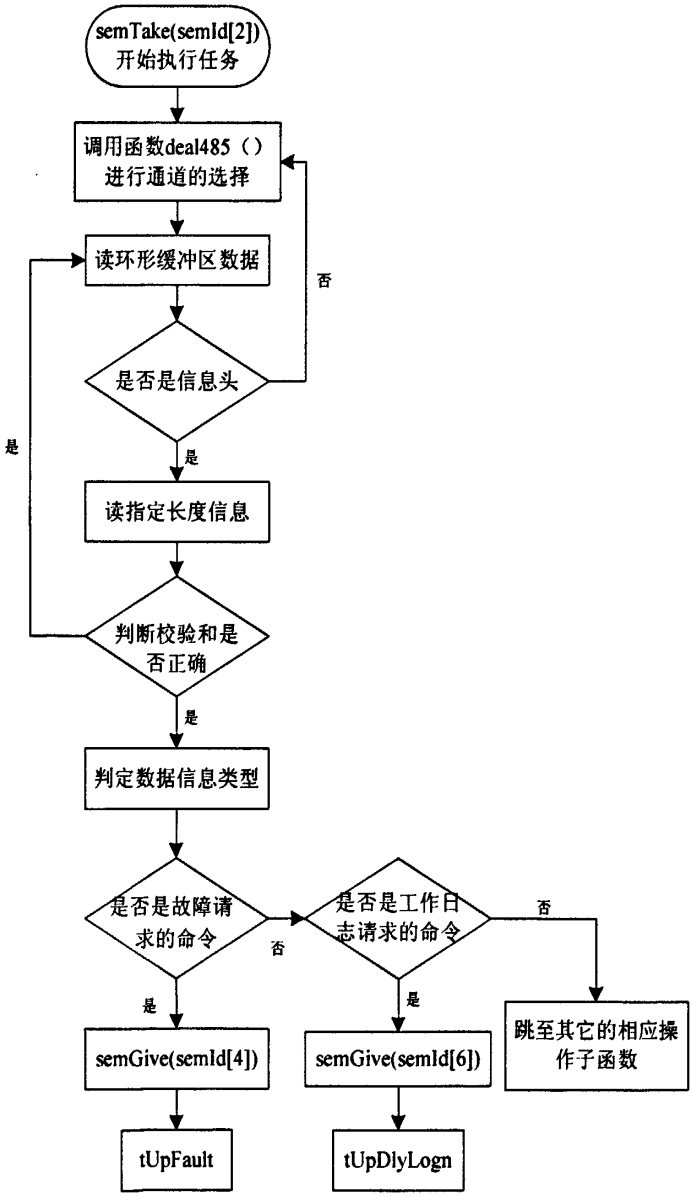


图 5.3 与上位机通信任务 tPCDeal 的流程图

### 5.2.4 50ms 周期任务（tSendTCCB）

50ms 周期任务是实现周期性请求 SSPC 状态电流数据；周期性采集模拟量、离散量、汇流条信息；对 CPU 进行周期性的自检。具体的实现流程如图 5.4 所示。

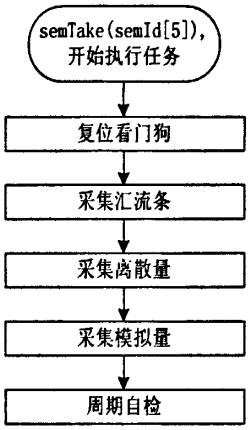


图 5.4 50ms 周期任务 tSendTCCB 的流程图

### 5.2.5 SSPC 数据处理任务（tDealCCB）

该任务的功能是接收来自 SSPC 的信息，并根据系统与下位机间的通信协议对信息内容进行一系列判断与处理操作，处理的信息内容主要包括电流及状态值、故障处理、滤波运算等。SSPC 数据处理任务（tDealCCB）流程如图 5.5 所示。

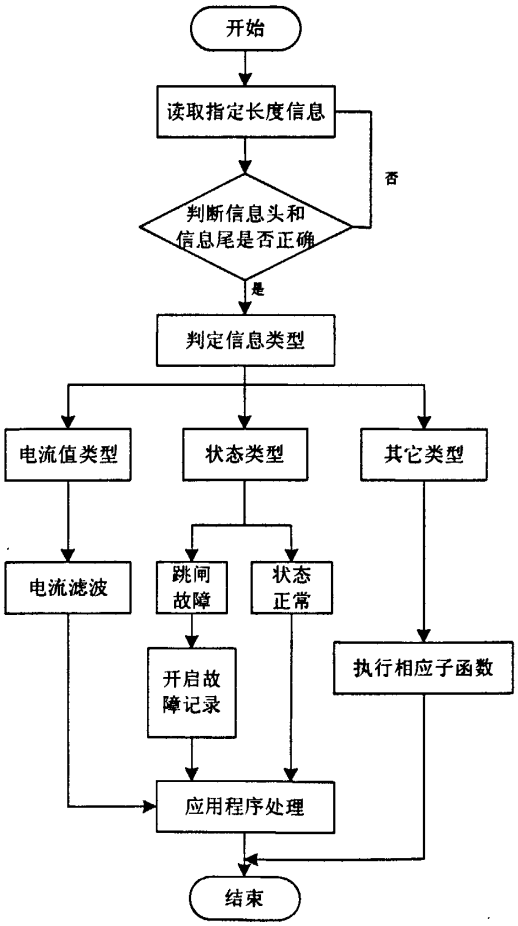


图 5.5 SSPC 数据处理任务流程图

从图 5.5 中可以看出，该任务启动后将首先从与 SSPC 通信的 422 总线的环形缓冲区中读取指定长度的数据，并把数据存放于预先申明的存储区中。接着判断该帧内容的信息头和信息尾是否正确，若不正确将继续读缓冲区中的数据；若正确则进行信息类型的判定，判定结果主要为三种：电流值类型、状态类型或者其它类型。若判定为电流值类型，则进行电流滤波，滤波完成后将进入相应的应用程序；若是状态类型则继续分析该路的状态是否正常，当发现状态不正常时，将分析具体的状态异常内容，如果是跳闸位异常将进行故障处理，当处理完异常的状态信息后将进行具体的应用程序处理过程；若判定结果为其它的类型，如控制命令的确认/不确认应答、额定电流的确认/不确认应答等，则执行对应的子函数。

## 5.3 GBN-ARQ 差错控制方法在系统中的应用

### 5.3.1 GBN-ARQ 协议的定制

本系统中差错控制技术主要应用在工作日志上传任务 tUpDlyLogn 和故障数据上传任务 tUpFault 中，本节主要介绍差错控制技术在工作日志上传任务 tUpDlyLogn 中的实现。

根据功能要求，工作日志上传任务 tUpDlyLogn 要上传 24 小时工作日志，每 500ms 更新一次，上传的数据有以下特点：数据量大；可靠性要求高；实时性要求高。串口通信模块与 PSP 是通过双 485 总线进行通信的，采用 115200bps 的波特率传输数据，航电系统的环境比较恶劣，误码率较高。因此，在上传工作日志的过程中，要进行差错控制，根据上一节的 GBN-ARQ 的 性能分析，对传输效率和传输时延的比值 $\lambda$ 求导，代入具体参数可得最佳帧长度近似为 32*8bit，即 32 个字节。

工作日志上传采用改进型 CRC-16 和 GBN-ARQ 的差错控制方式，每个数据帧具体的协议格式如表 5.2 所示。其中起始字是表示是该系统发送的数据；控制字 1 和控制字 2 表示是工作日志上传；帧编号表示该数据帧按照 GBN-ARQ 协议传输的编号；工作日志数据表示有用数据；校验存放改进型 CRC 校验码。

表 5.2 工作日志上传数据帧协议格式

| 名称 | 起始字 | | 控制码 1 | 控制码 2 | 帧编号 | 工作日志数据 | 校验 |
|---|---|---|---|---|---|---|---|
| 数据长度（字节） | 1 | 1 | 1 | 1 | 1 | 24 | 3 |
| 内容 | 0x55 | 0xaa | | | | | CRC 校验 |

### 5.3.2 GBN-ARQ 的实现流程

工作日志上传任务 tUpDlyLogn 中 GBN-ARQ 的实现流程如图 5.6 所示，设定 MAX_SEQ（发送窗口大小）为 7，创建环形缓冲区 DRamID 暂存 7 帧数据，计数器 nbuffered 记录 DRamID 中已经存储的数据帧的帧数。创建缓冲区 DFRamID 暂存从存储器中读出的需要发送的数据。FlagStart[i]标志开启帧编号为 i 的数据帧的超时定时器， FlagTimeout [i]标志帧编号为 i 的数据帧已经超时，则 GBN-ARQ 的实现过程如下：

(1)若 DRamID 已满，则停止发送数据，否则 CPU 继续调用函数 send_data() 发送新的数据帧。在 send_data()中首先判断缓冲区 DFRamID 是否为空，若为空则从 Flash 中读出数据填充缓冲区 DFRamID，然后再从该缓冲区中取出数据，组合成新的数据帧发送，同时把该数据帧存入缓冲区 DRamID 中，计数器 nbuffered 加 1，置位对应定时器开启标志 FlagStart[i]，开启工作日志数据上传定时器，同时增加帧编号的值；

(2)若收到上位机（PSP）的确认帧，则从缓冲区 DRamID 中取出被确认的帧丢弃该帧，表明该帧已经发送成功，不需要再存储，并复位对应定时器开启标志 FlagStart[i]和超时标志 FlagTimeout [i]，增加期望收到的确认帧的帧编号；

(3)若收到上位机的停止上传工作日志的命令，则跳出工作日志上传的任务；

(4)若期望得到确认的数据帧已经超时，则重新发送 DRamID 中所有已经发送但是还没有得到确认的数据帧，发送数据帧的同时还是要存储该数据帧，置位对应定时器开启标志 FlagStart[i]，开启工作日志上传定时器，同时复位超时标志 FlagTimeout [i]。

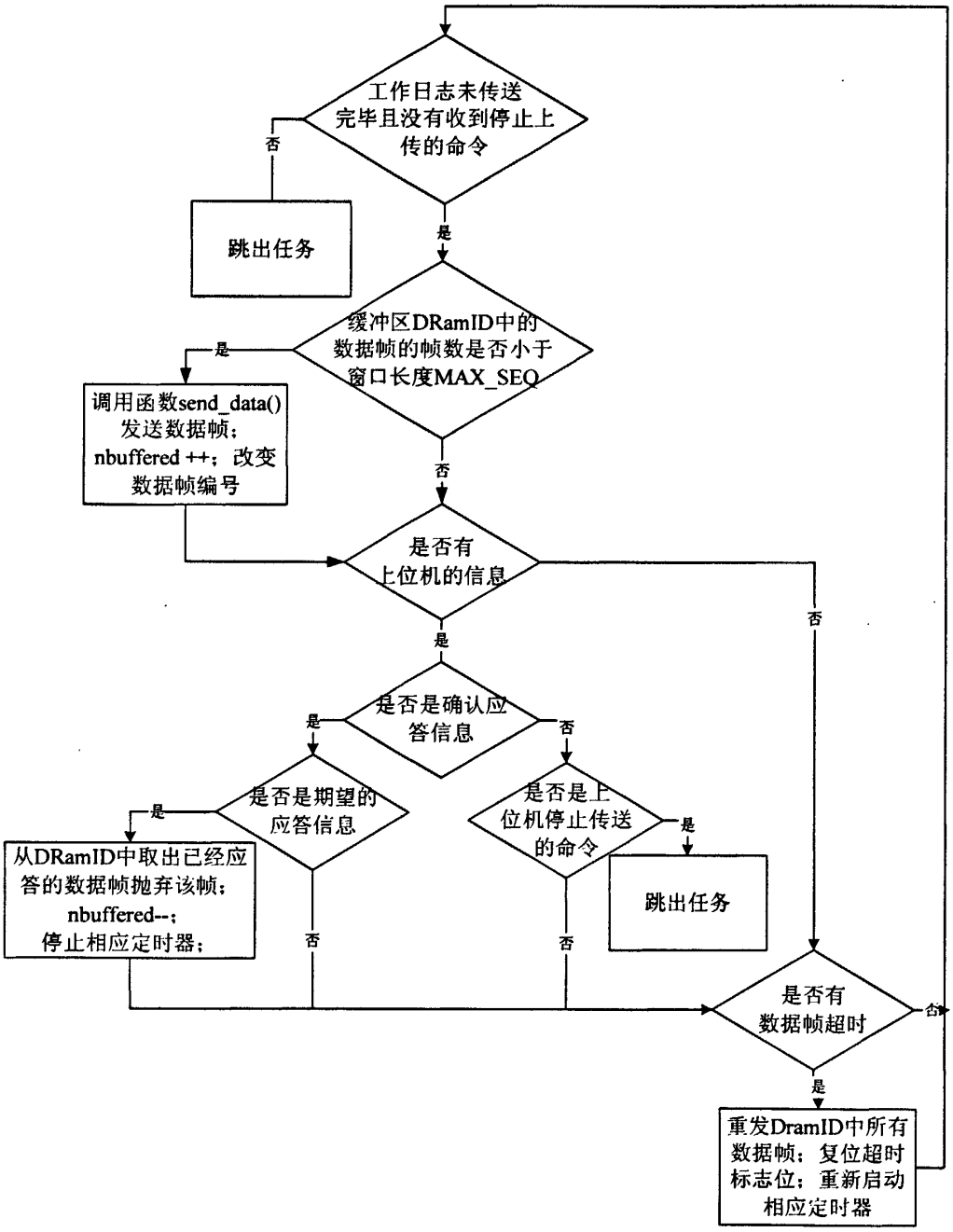


图 5.6 GBN-ARQ 在工作日志上传中的流程图

### 5.3.3 工作日志上传时 GBN-ARQ 机制的验证

记录使用 GBN-ARQ 重传机制时，在 RS485 总线的 115200bps 波特率条件下，系统一次上传 10Mbyte 工作日志数据的平均时间，不使用任何重传机制连续传输时的平均传输时间为 18 分钟。选择最佳帧长度为 32 个字节时，平均传输时间为

23.20 分钟，记录帧长度不同时，使用 GBN-ARQ 机制传输 10Mbyte 工作日志数据的平均时间如表 5.3 所示。

表 5.3 帧长不同时，工作日志上传的时延

| 帧长度（字节） | 20 | 25 | 28 | 30 | 32 | 40 | 45 | 50 |
|---|---|---|---|---|---|---|---|---|
| 平均传输时延（分钟） | 46.18 | 36.23 | 31.76 | 27.73 | 23.20 | 37.67 | 48.65 | 59.26 |

如表 5.3 所示，使用 GBN-ARQ 重传机制时，系统传输工作日志的的平均传输时延，比不使用任何重传机制时的传输时延都高，但是在最佳帧长度条件下，系统的传输延时最短。对比系统本身的工作日志数据和上位机接收到的工作日志数据，可得系统的错误率很低，即数据的传输性也很高。得出的结果进一步表明，在 GBN-ARQ 系统中选择正确的最佳帧长，不仅能保证数据的可靠性，而且能提高系统的实时性，减少数据传输的总时延。

## 5.4 本章小结

本章完成了基于 VxWorks 的串行通信系统的设计，首先对系统进行了功能分析，针对系统的功能要求对应用软件进行了总体设计，将将软件划分为多个模块，并建立了相应的任务，根据实际需要设置了各个任务的优先级，并给出了软件的主流程。在此基础上，对系统的各个任务模块进行了详细的设计，重点在工作日志上传任务中的实现了对 GBN-ARQ 差错控制方法的应用，根据工作日志上传要求设计了实现了 GBN-ARQ 差错控制方式，经过系统的测试与验证得出，选择最佳帧长可以保证系统的可靠性和实时性。

# 第六章 总结和展望

## 6.1 总结

本文对嵌入式串行数据通信系统中差错控制技术进行了研究，实现了基于VxWorks 嵌入式操作系统的串行数据通信系统，重点介绍了 GBN-ARQ 差错控制技术在在该系统中的应用。在此过程中，通过对经典 CRC-16 的分析改进了CRC-16，同时基于 Markov 链对 GBN-ARQ 进行了性能分析，提出了在 GBN-ARQ 协议中存在最佳发送帧长的结论。最后，通过在基于 VxWorks 的串行数据通信系统中选择改进型 CRC-16 作为检错码，提高了系统的检错率；同时选择最佳帧长的 GBN-ARQ 进行数据传输保证了系统的可靠性和实时性。

本文完成的主要工作在于：

1、通过分析现有差错控制技术的研究现状，阐述了研究基于实时嵌入式系统的差错控制技术的背景和意义，介绍了实时嵌入式系统中 ARQ 技术的现状和不足，重点介绍了几种常用的差错控制方式和差错控制编码。

2、分析了几种嵌入式操作系统，通过对比了几种嵌入式操作系统的优缺点，选用 VxWorks 操作系统作为本文研究的串行通信系统的内核。重点研究了 VxWorks 操作系统的组成结构，以及在 VxWorks 下应用软件的编写过程，最后对宿主机开发环境的建立和串口驱动的开发过程进行了详细阐述。

3、分析了经典 CRC 校验的原理和性能，并研究了现有的 CRC 校验的实现方法，在此基础上，改进了 CRC 校验，在 VxWorks 下实现了改进后的 CRC，并对 CRC 校验检错后的误码率进行了仿真分析，验证了改进型 CRC 的检错能力确实优于经典 CRC。

4、分析了 GBN-ARQ 的工作过程，将 GBN-ARQ 的工作过程分为七个状态，得出这七个状态的一步 Markov 状态转移图；依据该状态转移图，推导出了使用 GBN-ARQ 传输数据时的有效传输效率的表达式 1，对该表达式进行分析和仿真，得出存在一个最佳传输帧长使得传输效率最大的结论；同时还依据该状态转移图，列出了使用 GBN-ARQ 传输一定长度数据时的传输时延的表达式 2，同样对该表达式进行分析和仿真，得出存在一个最佳传输帧长使得传输时延最短的结论；最后，通过对表达式 1 和表达式 2 进行分析和仿真，得出存在一个最佳传输帧长使得同时满足传输效率最大和传输时延最短的要求。

5、对基于 VxWorks 的串口通信系统进行了功能分析，依据功能要求对系统应用软件进行了任务的划分，并对各个任务模块进行了详细的设计。重点分析了 GBN-ARQ 差错控制技术在系统工作日志上传任务中应用，并在系统中传输了一定

长度的工作日志数据，对最佳帧长的 GBN-ARQ 差错控制方式进行了验证。结果表明，选用最佳帧长的 GBN-ARQ 进行实时嵌入式系统的串口数据通信，在满足系统实时性要求的前提下，还可以提高系统的可靠性。

论文的重点在于基于 Markov 链对 GBN-ARQ 进行了性能分析，得出了使用 GBN-ARQ 传输数据时的有效传输效率和传输时延的表达式，为计算 GBN-ARQ 的最佳帧长提供了依据。本文是在基于 VxWorks 的串行数据通信系统中验证了存在最佳帧长的结论，该结论也可用于其它的对传输效率和传输时延有要求的嵌入式系统中。且本文的分析方式还可用于对其他的 ARQ 重传机制进行性能分析。

## 6.2 展望

本文所提出的差错控制方式虽然可以提高系统的可靠性和实时性，但是仍然存在一些需要改进的地方，主要由以下几点：

1、改进型的 CRC-16 的检错性能改善不是很明显，需要对 CRC-16 进行进一步的研究，得出更好的改进算法。而且对改进型的 CRC-16 的仿真只是在 labview 环境中加入高斯噪声的条件下进行的，不能完全反应真实信道的环境。

2、环境很恶劣的信道中，数据帧出错概率比较大，如果使用 GBN-ARQ 作差错控制方式，重传的数据帧急剧增多，会导致系统的传输效率大幅下降，即使使用取到最佳帧长的 GBN-ARQ 也不能大幅度改善系统性能；

3、基于 VxWorks 的串口数据通信系统中，并没有对系统实现的其它功能进行详细的阐述和验证，而只是对使用 GBN-ARQ 上传工作日志的任务进行了验证，而且工作日志上传的时延也只是在 RS485 总线上测试出的平均值，会有微小的偏差。

4、文没有针对出现链路故障的情况，提出解决方式，如果链路出现故障，数据通信将直接中断。下一步的工作可以围绕如何实现链路冗余进行。

由于作者水平有限，论文中的缺点和不足之处在所难免，恳请各位老师、专家和学者批评指正。

# 致 谢

本论文的研究工作是在导师相征教授的悉心指导下完成的。相老师渊博的知识、严谨的治学态度、不断创新的科研精神和孜孜不倦的工作作风，令我受益匪浅、终身难忘。在学习和科研过程中，相老师悉心指导，言传身教，同时营造了一种宽松而不失严谨、科研与学术相结合的研究氛围，使我在理论和实践上都得到了很大的锻炼和提高。我今天在各方面取得的成绩与导师都有着密不可分的联系，在此表示深深的感谢！

我要向工作在西安旭彤电子有限公司的王卫斌老师表示感谢！

我还要感谢我的同学黎石磊、赵楠、曾峥、张一鸣、刘校伟、徐连军、齐佩汉、琚祥、周一波，王加祥等，通过近三年的学习和生活的相处，我们互相帮助，共同提高，结下了浓厚的友谊，这段共同奋斗的日子将使我终身难忘！

我还要感谢我的师弟师妹们，祝他们学业有成！

三年的学习和研究工作得以顺利完成，也离不开父母和弟弟妹妹给予的支持和鼓励，藉此机会向他们表示由衷的谢意！

最后，我还要向审阅本文并提出宝贵意见的专家和教授们致以谢意，感谢你们的辛勤劳动！

# 参考文献

[1] 李建东.信息网络理论基础[M].西安：西安电子科技大学出版社，2001

[2] Chu W .W. Optimal message block size for computer communications with error detection and retransmission strategies ,IEEE Trans ,Commun ,vol COM-22,Oct,1974

[3] Morris J.M. Optimal blocklengths for ARQ error control schemes ,IEEE Trans, Commun .,vol .COM-27 ,Feb-1979

[4] Martins A.C.ARQ protocols with adaptive block size perform better over a wide range of bit error rates ,IEEE Trans ,Commun.,vol .38.June 1990

[5] MINN H,ZENG M,ANNAMALAI A,etal. An efficient ARQ protocol for adaptive error control over time-varying channels[J].Wireless Personal, Communications, 2001,17.pp:3-20

[6] VOJINVI'C R,PETROVI'C G,PETROVI'C Z. The analysis of the adaptive three-mode ARQ GBN scheme using retransmission cycles mechanism[J].Int J Electron Commun(AEU),2002,56(6).pp:389-395

[7] Varthis E G,Fotiadis D I. A comparison of stop-and-wait and go-back-n ARQ schemes for IEEE 802.11e wireless infrared networks [J].Computer Communications ,2006,29(8).pp:1015-1025

[8] Cam R,Leung C. Throughput analysis of some ARQ protocols in presence of feedback errors[J].IEEE Transactions on Communications ,1997,45(1):35-44

[9] Konheim A G .A queueing analysis of two ARQ protocols[J].IEEE Comunication, 1980,28(7).pp:1004-1014

[10] Yoshimoto M, Takine T,Takahashi Y. Waiting time and queue length distributions for go-back-n and selective-repeat ARQ protocols[J].IEEE Transactions on Communications ,1993,41(11).pp:1687-1692.

[11] 李善平. LINUX 嵌入式设计与应用. 北京：清华大学出版社，2006

[12] 金巍. 一种并行 CRC 计算原理及 FPGA 实现. 第八届计算机工程与工艺全国学术年会.2003,8

[13] 熊桂喜，王小虎.计算机网络.清华大学出版社.1998,7

[14] 王新梅，肖国镇. 纠错码—原理与方法. 西安：西安电子科技大学出版，2001.4

[15] 夏丽文，差实生，相洁. 微型计算机接口技术. 电子工业出版社，2002,5

[16] 王爱英. 计算机组成与结构. 清华大学出版社，2000.12

[17] 龚剑，林夏菲. 浅析嵌入式系统体系结构与开发流程. 科技信息（学术研究）2008.11

[18] Maarten Boasson .Control systems software.IEEE Transations on Automatic Control,1993,38(7).pp:1094-1106

[19] Stuart Bennett.Real-time Compter Control.London:Prentice Hall International Ltd,1988.pp:80-86

[20] 季志均,马文丽,陈虎等. 四种嵌入式实时操作系统关键技术分析, 2005

[21] WindRiver. Tornado device workshop. WindRiver. Inc,1999

[22] WindRiver. Tornado User's Guide(Windows Version). WindRiver System Inc,1999

[23] 王忠, 李延社, 游智胜. CRC 算法设计与程序实现. 电子测量技术. 2007.12

[24] 张平安.16 位循环冗余校验码（CRC）的原理和性能分析.山西科技.2005.5

[25] Shu Lin and Philip.S.Yu,"A hybrid ARQ scheme with parity retransmission for error control of satellite channels",IEEE transactions on communications,vol.com -30,No.7,July 1982

[26] 王仲文, 赵晓群. Go-back-N ARQ 通信系统的性能研究. 东北重型机械学院学报. 1987, 3, 11(1)

[27] 龚天富, 徐开来等. ARQ 通信系统的性能优化与帧长度的动态选择. 计算机应用. 1994, 3

# 作者在读期间研究成果

一、专利成果

1、相征，万娟，汤书苑，苑峰等，基于 VxWorks 操作系统的混合任务集调度方法，申请号：200910023361.7。

2、相征，苑峰，万娟，汤书苑等，基于 Vxworks 操作系统的 ARINC429 通信冗余方法，申请号：200910023264.8。

二、参加的科研项目

1、“新一代飞机航电通信系统关键技术研究”项目，项目编号：GJ0203100102，项目来源：空军重点型号背景预研，参与时间：2009.10-2010.6，作为参与人，主要完成基于 VxWorks 的应用软件的设计，目前该系统已交付。