ZMODEM 内联应用之文件传送协议 (1988 年版)

原著:Chuck Forsberg
译者: 洋文馆

2012 年 5 月 10 日

目录

前言

文件的重新发布不受限制, 但请保证文档的完整性.

请尽可能地传播该文档

第 1 章　预定的读者

本文档适合于电讯项目经理, 系统开发者和其他选择在拨号网络和对应环境下实现异步文件传送的人.

第 2 章　为什么要开发 ZMODEM?

在开发 ZMODEM 的前五年,Ward Christensen 的 MODEM 协议已经广泛用于计算机系统的数据互换, 它几乎是任何一个通讯程序必须支持的协议, 现在叫做 XMODEM.

随着计算能力的提升,modem 和网络在使用 XMODEM 协议时, 已经远远超出了它微秒级的初始设定, 而相关的一些应用也暴露出它的一些弱点:

+ 繁琐的用户接口, 只能适合于计算机爱好者, 必须在键盘上输入多个命令, 才能实现文件的传送.
+ 由于程序中必须给出多个命令, 而没有一个简单的可选菜单, 这不合理.
+ 当用于分时系统, 包交换网络, 卫星环路和缓冲式 (带错误校正)modem 时, 必须忍受短的块长度所引发的传送.
+ 8bit 的 checksum 和无保护性的监控, 将使得文件传送中包含未被发现的错误, 或是错误文件.
+ 每次命令只能发送一个文件, 文件必须给出两次, 首先是发送程序, 之后是接受程序.
+ 传送文件将会累积出 127byte 的无用字节.
+ 修正数据和其他文件属性会丢失.
+ XMODEM 需要 8bit 数据的明码, 也就是一共为 256 个编码, 因此它无法操作在使用 ASCII 流控制或是 escape 编码的网络中, 因为在呼叫过程中, 会设定网络的明码, 这时网络将失去重要的控制功能.

之后还开发了一些协议, 但都不能满足要求:

+ 一些私有协议缺少了使用文档和例子程序, 如 Relay, Blast, DART 和其他与供应商利益挂钩的协议, 这些协议无法从公共应用中得益.
+ 链路层协议, 例如 X.25, X.PC 和 MNP, 无法管理应用之间的文件传送.
+ 链路层协议无法消除端到端的错误, 在无错网络中的接口不一定是无错的, 但是无错网络本身就是不存在的.
+ Kermit 协议的文件传送环境, 不利于 XMODEM 的使用, 它可以适应限制其效率的传统主机环境, 这是一种折中的实现方式, 即使在完整透明的通道中,Kermit 控制字符对二进制文件传送效率的影响在 75% 以上.[1]

  在多种 Kermit 程序中使用了一些子模式, 其中包括了二进制文件传送的不同方法, 因此如果用户无法正确指定对应的子模式, 两种 Kermit 程序在互传时, 将出现莫名其妙的错误.

  Kermit Sliding Windows (”SuperKermit”) 在日益复杂的网络中改进了传输速率, SuperKermit 需要全双工的通讯, 并有能力在输入队列中检查字符, 但在一些操作系统中无法实现检查.

  SuperKermit 状态转换可使用一种特殊语言 (wart) 进行编码, 它可使用 c 编译器.

  SuperKermit 可为每个 96byte(只会包含少量的控制符) 的数据包, 发送一个 ACK 包, 这会降低高速 modem 的传输速率, 在测试中每秒需要包含 177-1350 个不等的控制符.

XMODEM 协议也有一些扩展, 对性能和 (某些情况下的) 用户接口进行了改进, 但这些改进只对部分应用有效,XMODEM 中未加入保护性的消息控制, 将损害它的可靠性, 一些复杂的专利技术, 如 Cybernetic Data Recovery(TM) 可以改进可靠性[2], 但是并不能随意使用. 部分 XMODEM 的变种协议还包含了重大的设计缺陷.

[1] 一些 Kermit 程序中支持运行长度的编码

[2] 比如 DSZ, ZCOMM, Professional-YAM 和 PowerCom

+ XMODEM-k 使用了 1024byte 的块, 以此来降低传输延时带来的开销, 相比于 XMODEM 可获取 87% 的性能提升, 但网络延时会降低它的性能, 有些网络无法传送不包含流控制的 1024byte 的数据包, 这对于需要完全透明通道的 XMODEM 来说, 是无法应用的,XMODEM-k 会在接收文件中加入无用数据.

+ YMODEM 可发送文件名, 文件长度和文件初始创建日期, 并可选用 1024byte 块, 以提供传输速率, 如果处理文件的大小不是 1024 或 128byte 的整数倍时, 将会很尴尬, 尤其是在文件长度未知, 或是在传送过程中发生变化时. 有大量不合格和非标准的程序声称会支持 YMODEM, 以适应未来的复杂应用.

+ YMODEM-g 支持高效的批量文件传送, 并能精确比对文件的长度和修改日期,YMODEM-g 是 YMODEM 的修订版, 但并没有为数据块使用 ACK, 从本质上说,YMODEM-g 对网络延时并不敏感, 由于 YMODEM-g 并不支持错误恢复, 因此必须使用一个硬链接或是一个可靠的链路层协议, 成功的高速应用特别在意透明的流控制, 当 YMODEM-g 发现一个 CRC 错误时, 数据传送将停止,YMODEM-g 其实更容易实现, 因为它更接近于标准的 YMODEM-1k.

+ WXMODEM, SEAlink, MEGAlink 可看成是 ZMODEM 的一个子集, 对” 经典 XMODEM” 进行改进, 在理想条件下, 它们可以提供最好的性能

另一种 XMODEM 的扩展, 比如 Omen 科技公司的 OverThruster(TM) 和 OverThruster II(TM), 在某些错误恢复的条件下, 可改进 XMODEM 的速率.

ZMODEM 协议可修正上述的这些缺陷, 并支持 XMODEM/CRC 特性以及现有技术.

第 3 章　ZMODEM 协议的设计规范

文件传送协议的设计, 即是在发生冲突的环境中使用一个折中的办法.

3.1　易用性

+ ZMODEM 允许程序初始化文件发送.
+ 发送者可向正在接收的程序发送命令或是修改.
+ 文件名只需输入一次.
+ 支持菜单选择.
+ 可在批量传送中使用通配符.
+ 在传送初始化时, 只需少量的键盘输入.
+ ZRQINIT 架构可使发送程序自动触发下载.
+ 在不支持 ZMODEM 的环境中, 可使用 YMODEM 协议.[1]

3.2　传输速率

所有文件传送协议都需要在速率, 可靠性, 通用性和基于设计者的知识和技术能承受的复杂性的一个折中.

在 ZMODEM 中, 有三类应用可获益:

+ 关心延时 (相对于字符传输时间) 和低误码率的网络应用.
+ 分时和缓冲式的 modem 应用, 它们关心延时和反向通道传输的快速跌落的速率. ZMODEM 可节省反向通道的带宽, 允许 modem 动态分配两个方向上的带宽, 使之获取最佳的速度. 特殊的 ZMODEM 特性允许在多种分时主机中简单高效地实现.
+ 传统的 modem 与 modem 之间的通讯, 会包含高误码率.

与 Sliding Windows Kermit 不同,ZMODEM 不会在高误码率和延时的情况下, 优化最佳的速率, 这可降低源码的复杂度和对内存的需要,ZMODEM 支持快速错误恢复, 并兼容 XMODEM 的实现

在网络无延时的情况下, 快速错误恢复也是需要的, 相比之下 MEGAlink 速度更快, 并兼容 YMODEM 和 X-MODEM.

文件传送会在程序启动之后立刻开始, 而不会有 XMODEM 的 10s 延时.

[1] 只适用于 X/YMODEM 的传输媒介

3.3 完整性和健壮性

当 ZMODEM 会话开始后, 所有的事务会有一个 16 或 32bit 的 CRC 进行保护.[2] 复杂的专利技术, 比如 Omen 的 Cybernetic Data Recovery(TM)[3]在可靠性传送中不是必须的, 为了 ZMODEM 的高可靠性 (相对于 XMODEM), 完整的数据保护和监控信息来自于协议, 其中包含 WXMODEM, SEAlink, MEGAlink 等.

可选 ADCCP(如 ANSI X3.66, 又称为 FIPS PUB 71 和 FED-STD-1003, 以及 CCITT's X.25) 给出的 32bit CRC 检查序列, 在正确应用的情况下 (预设为 -1, 逆向),32bit CRC 可使未检出错误, 降低至少 5 个数量级 (10 的 5 次方)

安全机制可预防类似"Trojan Horse" 消息, 模拟正常的命令或文件下载.

3.4 易于实现

ZMODEM 适合多种系统:

- 未实现磁盘和串行 IO 的微控制系统.
- 无法实现串行发送和接收的微控制系统.
- 需要 XON/XOFF 流控制的计算机和网络
- 无法检查串行输入队列, 以查看所需数据是否到达的计算机.

尽管 ZMODEM 为多个" 线程" 提供了" 钩子", 但并不想替代链路层的协议, 比如 X.25

ZMODEM 可容忍网络和分时系统的延时, 并实现连续的数据传送, 除非接收器中止, 并要求发送器重传数据. 事实上 ZMODEM 会将一个文件看成一个窗口[4], 并使用简单的窗口缓冲管理, 以避免窗口的溢出, 而影响 MEGAlink, SuperKermit 或其他组件.

ZMODEM 提供了一种通用功能, 用于文件传送协议, 它可以直接传送或是依赖于链路层协议, 比如 X.25, MNP, Fastlink 等, 在使用 X.25, MNP, Fastlink 时,ZMODEM 可在错误控制和通讯链路之间的接口中, 检出和修正错误.

ZMODEM 是在 Telenet 规约下开发的公开协议,ZMODEM 协议和 Unix rz/sz 程序都是开源的, 因此协议,Unix rz/sz 和 ZMODEM 名称的使用, 无需许可, 也不包含商标或版权的限制.

[2]除了 CAN-CAN-CAN-CAN-CAN 停止序列之前, 还需要 5 个连续的 CAN 字符
[3]比如 Professional-YAM, ZCOMM, PowerCom
[4]流规则将在后续章节中介绍

第 4 章　ZMODEM 的发展

在 1986 年,Telenet 资助了一个应用于文件传输协议的开源项目, 该协议部分提升了网络客户端在使用 XMODEM 和 Kermit 文件传送器时的速率问题.

最初我们考虑过对 XMODEM 进行少量修改, 使其在包交换网络中获得高性能, 并保持 XMODEM 的简单性.

初始想法是在 XMODEM 的 ACK 和 NAK 字符中加入块号. 其结果可使发送器在等待响应时, 可以发送多个块.

但如何在 XMODEM 的 ACK 和 NAK 中加入块号? 并使 WXMODEM, SEAlink, MEGAlink 和其他协议也加入二进制 byte 来表示块号.

在 ZMODEM 中并不适合使用纯的二进制, 因为它无法在某些 modem, 网络和操作系统进行双向传送, 有些操作系统无法识别回传命令[1], 除非一个中断信号或是一个系统已知的编码或序列出现, 为了将 XMODEM 中简单 ACK/NAK 序列存在的所有问题一起解决,ACK 和 NAK 字符只能变成一个真正的数据包.

对窗口的管理则是另一个问题[2], 随着 SuperKermit 协议源码的调试经验的增加, 可预估 1200bps 速率上, 窗口的尺寸大概是 1000 个字符, 但高速 modem 则需要 20k 甚至更多的字符数, 大多数 SuperKermit 都是比较低效复杂的, 而调试基本是针对环形缓冲和窗口管理, 这是一种最好的工作方式.

另 XMODEM 和后续扩展最伤心的一点是错误恢复, 接收器需要确定发送器是否已给出响应, 或是准备响应或处理重传请求?XMODEM 解决该问题的方法是, 在一定周期的静默中, 丢弃一些字符, 如果这个时间太短, 则使输出 (网络或分时处理) 得到一个暂停, 以模拟对应的错误恢复. 如果这个超时太长, 应将影响速率, 这时需要一条噪声线 (等同于开关), 来控制协议的锁定.SuperKermit 解决该问题的方式, 即在包字符 (SOH) 中包含一个显式的起始标记,WXMODEM 和 ZMODEM 都使用唯一的字符序列来标记帧的起始位,SEAlink 和 MEGAlink 不会有这类问题.

在流式协议中将会遇到其他类型的错误恢复, 接收器如何知道发送器已经识别到错误信号? 是在下一个数据包中给出错误信号的响应? 还是在" 队列" 的剩余部分? 由于发送器没有接收到错误信号, 新发送的数据包也将被丢弃? 连续发送另一个错误信号之前, 需要等待多久? 如何使用协议将混合信号简化成一个参数?

SuperKermit 中可选择重传, 因此它也可以接收好包, 每次 SuperKermit 在接收数据包时, 它必须确认接收的数据包是否是需要的, 并询问下一个数据包, 实际上复杂软件都会通过"hack" 的方法, 已达到可接受的健壮性.[3]

对于 ZMODEM, 我们决定放弃 SuperKermit 复杂的包组件结构, 以及关联的缓冲管理逻辑和内存需求.

XMODEM 和 WXMODEM 的另一个缺陷是会在文件中加入无用数据, 它可接受无准确长度的 CP/M 文件 (它不能用于新系统, 比如 PC-DOS 和 Unix),YMODEM 将会使用首块中给出的文件长度, 对输出文件进行修正, 但这容易造成传送文件的数据丢失.

在某些情况下, 文件长度可以是未知的, 等同于从进程中获取数据, 而变长的数据包还可解决一些问题.

由于某些字符就是会发生丢失, 从而将错误的字节填入到定长的数据包, 或是变长的数据包中, ZMODEM 中子数据包的长度都由数据包尾部的 escape 序列标记, 这类似于 BISYNC 和 HDLC

一个 ZMODEM 头部会包含一个" 帧类型",4 个字节的监控信息, 以及 CRC, 头部中的数据帧可包含在一个或

[1] 发送响应时, 无需停止当前任务
[2] 窗口是指发送器和接收器之间的数据传送
[3] 当 SuperKermit 需要错误时,wndesr 函数将被调用, 并请求下一个块, 而突发的错误也需要生成一个请求, 用错误数据的重传

多个数据子包中. 若不存在传送错误, 一个文件可在一个数据帧中进行传递.

由于发送系统可能对多个控制字符或是奇偶校验敏感, 在逆向数据路径中, 接收器发送的头将使用 hex 格式, 一个通用底层程序将接收该头部, 并允许主程序处理该头和数据子包.

二进制 (高效) 和 16 进制 (与应用友好) 帧是等价的, 发送程序可发送一个” 邀请接收” 的序列, 激活接收器, 也不会因使用了未定义的控制字符, 而是远程应用崩溃.

” 重新出发” 的协议设计提供了一个机会, 可向多种协议进行学习, 而形成一个最新的协议

从 Kermit 和 UUCP 可获知, 能使用互换系统参数的初始会话的概念.

ZMODEM 也借鉴了 Compuserve B 协议中主机传送所使用的单命令 AutoDownload 和 downloading, 使用的安全机制也可防止密码被盗取,Trojan Horse 对 ZMODEM 电源的危害.

我们也能理解这类伤感, 并看着越来越多的通讯中, 因文件传送的混乱, 所造成的通讯和分时错误, ZMODEM 的文件传送恢复和高级文件管理可用于替换同类的协议.

虽然 ZMODEM 出现的时间很短, 但已经明显展示出:ZMODEM 用户可友好地使用自动下载的功能, 但用户必须为每次发送和接收程序分配对应的传送选项, 现在传送选项也可由发送程序指定, 并使用 ZFILE 头传递给接收程序.

第 5 章　罗塞塔石碑

这里会有一些定义会反映在计算机的媒介中, 并只会标记文件传输协议, 而不是其他特殊的程序.

- **FRAME** 一个 ZMODEM 帧中会包含一个头部和 0 个或多个数据子包.
- **XMODEM** 即是 1977 年 Ward Christensen 的 MODEM2 程序中给出的文件传送器, 它也可称为 MODEM 或 MODEM2 协议,MODEM7 中还包含不常用的批量文件模式, 它的一些别名可在"CP/M Users's Group" 和"TERM II FTP3" 中找到, 该协议会被大多数通讯程序支持, 因为它更容易实现.
- **XMODEM/CRC** 用于替换 1bit checksum 的 XMODEM, 包含 2 个字节的 CRC(CRC-16), 改进了错误检出的功能.
- **XMODEM-1k** 即为可选择 1024byte 块的 XMODEM-CRC
- **YMODEM** 将使用 XMODEM-CRC 协议, 并可完成批量传送和可选的 1024byte 块, 在 YMODEM.DOC 文件中进行了介绍.[1]

[1]它是 TeleGodzilla YZMODEM.ZOO 中的一部分

第 6 章　ZMODEM 的需求

ZMODEM 需要 8bit 传送单位[1],ZMODEM 可转义网络控制字符, 因此可用于包交换网络, 通常 ZMODEM 可在支持 XMODEM 的路径中使用, 其他协议则无法实现.

同时支持流式传送[2], 传输路径中可使用断言流控, 或是数据无损的全速传送, 否则 ZMODEM 发送器必须管理窗口的尺寸.

6.1　文件类型

6.1.1　二进制文件

ZMODEM 中放入的二进制文件没有约束, 而文件的 bit 数将是 8 的倍数.

6.1.2　文本文件

由于 ZMODEM 可用于不同类型的计算机系统之间的传送, 文本文件是必须满足的最低要求, 因为在大多数系统和环境中都可以使用文本文件.

文本行能包含可打印的 ASCII 字符, 空格,tab 和 backspace

行尾的 ASCII 字符　ASCII 代码定义中允许使用一个 CR/LF(015,012) 序列, 或是一个 NL(012) 字符终止文本行, 文本行还可使用单个 CR(013) 进行逻辑终止, 但这不是一个 ASCII 文本.

CR(013) 不会实现换行, 只意味着加印, 并且不能视为一个文本行的逻辑分隔符, 加印行可打印所有之后传递的重要字符, 并允许 CRT 显示这些重要的文本, 重复打印的字符可包含 backspace 和 CR(015) 的加印行之后, 但不能包含在 LF 之后.

重复打印字符可与 backspace 一起生成, 并能与重要字符一起发送, 但 CRT 中不能显示重复打印字符, 发送程序可使用 ZCNL bit 告知接收程序, 将接收的行尾转成文本行的实际行尾.[3]

[1]ZMODEM 设计允许为低通透传输媒介设置编码数据包
[2]包含 XOFF 和 XON, 或是带外流控, 比如 X.25 或 CTS
[3]已被转义的文件在长度被修改后, 无法使用 ZCRECOV 转换选项进行更新

第 7 章　基本的 ZMODEM

7.1　数据包

ZMODEM 帧与 XMODEM 块有些不同, 不使用 XMODEM 块的理由如下:

- 块号受限于 256
- 不支持变长块
- 链路层如果存在劣化的协议信号, 将直接导致文件传送的错误, 尤其是 modem 错误有时会生成虚假的块号,EOT,ACK, 其中虚假的 ACK 是最麻烦的, 它会使发送器丢失与接收器之间的同步.

  而一些优秀的程序, 例如 Professional-YAM 和 ZCOMM 可使用灵活的专用代码克服这些弱点, 以实现一个期望的健壮协议.
- 当链路层的同步信号丢失, 确认 XMODEM 块的起始和结束很困难, 这也妨碍了快速的错误恢复.

7.2　链路层的转义编码

ZMODEM 可以实现明码, 能通过转义序列来扩展 8bit 的字符集 (256), 并且是基于 ZMODEM 链路层的转义字符 ZDLE.[1]

链路层的转义编码允许变长的数据子包, 并且不会有独立字节计数的开销, 并可以检出帧起始位, 且无需特别的时序要求, 这使得快速错误恢复更容易实现.

链路层转义代码还是会带来一些开销, 在最坏的情况下, 如果文件只包含转义编码, 将会增加 50% 的开销.

ZDLE 字符是比较特别的, 它可表示某些类型的控制序列, 如果 ZDLE 字符出现在二进制数据中, 并使用了 ZDLE 前缀, 将以 ZDLEE 发送.

ZDLE 值为八进制数 030 (ASCII CAN), 该特殊的值可允许 5 个连续的 CAN 字符组成一个字串, 停止一次 ZMODEM 会话, 这与 YMODEM 会话的停止是兼容的.

由于 CAN 不会用于正常的停止操作, 交互应用和通讯程序都将监控数据流是否出现 ZDLE, 其后的字符能被扫描, 以确定 ZRQINIT 头, 它可请求自动下载命令或是文件.

收到 5 个连续的 CAN 字符后, 将会停止一次 ZMODEM 会话, 并会发送 8 个 CAN 字符.

接收程序可确认 ZDLE 的任意序列, 并在之后的一个字节中的 bit 6 进行置位,bit5 进行清位, (都是大写字符, 两者是等价的), 并等同于控制字符 bit 6 的取反, 它允许传送器转义任何不能在通讯媒介中发送的控制字符, 另外接收器还可识别出 0177 和 0377 的转义符.

ZMODEM 软件可转义 ZDLE, 020, 0220, 021, 0221, 023, 0223, 如果之前出现 0100 或 0300(@), 015 和 0215, 也可实现转义, 但必须保护 Telenet 命令, 转义为 CR-@-CR. 接收器将在数据流中忽略 021, 0221, 023, 0223 字符.

zm.c 中的 ZMODEM 程序可接受一个选项, 用于转义所有控制符, 并允许在低通透度的网络中操作, 该选项可由发送程序或接收程序给出.

[1] 该常量和其他常量将在 zmodem.h 头文件中定义, 注意它是一个八进制数.

7.3 帧头

所有 ZMODEM 帧都是从一个帧头开始, 它可以使用二进制或 16 进制发送,ZMODEM 将使用一个单独的程序来识别二进制和 16 进制头, 无论是那种帧头都会包含以下相同的流信息:

- 一个类型字节[2] [3]
- 四个字节数据, 用于表示与帧类型相关的标志和数值

TYPE: 帧类型
F0: 最低有效位的标志
P0: 最低位置的文件
P3: 最高位置的文件

TYPE F3 F2 F1 F0

TYPE P0 P1 P2 P3

图 7.1: 头中的字节顺序

7.3.1 16bit CRC 的二进制头

该二进制头将被发送程序传送给接收程序,ZDLE 编码适用于 XON/XOFF 流控, 二进制头将从 ZPAD, ZDLE, ZBIN 序列开始, 帧的类型字节是 ZDLE 编码,4 个标志/位置字节也是 ZDLE 编码,16bit CRC 可包含 0 个到多个数据子包, 这与帧类型有关.

函数 zsbhdr 可发送一个二进制头, 函数 zgethdr 可接收一个二进制或 16 进制头.

* ZDLE A TYPE F3/P0 F2/P1 F1/P2 F0/P3 CRC-1 CRC-2

图 7.2: 16bit CRC 的二进制头

7.3.2 32bit CRC 的二进制头

32bit CRC 的二进制头会将 ZBIN (A) 字符替换成 ZBIN32 (C) 字符, 还将发送 4 个 CRC 字符, 32bit CRC 可包含 0 个到多个数据子包, 这与帧类型有关.

只有当接收器表示具备 CANFC32 bit 的能力后, 公共变量 Txfcs32 将能被设为 True, 使能 32 bit CRC, 函数 zgethdr,zsdata,zrdata 都能自动调整使用此帧校验模式.

* ZDLE C TYPE F3/P0 F2/P1 F1/P2 F0/P3 CRC-1 CRC-2 CRC-3 CRC-4

图 7.3: 32bit CRC 的二进制头

7.4 16 进制头

接收器可发送 16 进制头, 而发送器也可使用 16 进制头, 但不能跟随二进制数据子包.

由于随机控制符,16 进制编码可保护回传通道, 而 16 进制头的接收程序将忽略奇偶校验.

Kermit 编码风格中会保留控制符, 而丢弃奇偶校验, 因为与一些分时系统的行编辑函数的交互机会将越来越多, 若在接收程序中使用 16 进制头, 可在检出错误时, 允许使用控制符中断发送器. 用 16 进制头替代二进制头, 其实更方便, 在 16 进制头之后的数据包, 应使用 CRC-16 进行保护.

[2]帧类型的数值可从 0 开始, 或为状态变换表的最小内存需求
[3]ZMODEM 的后续扩展会使用类型字节的高位, 表示线程的选择

16 进制头将从序列 ZPAD, ZPAD, ZDLE, ZHEX 开始,zgethdr 程序将与 ZPAD-ZDLE 序列同步, 扩展的 ZPAD 字符允许发送程序检出一个同步头 (表示一个错误条件), 之后将调用 zgethdr 接收该同步头.

类型字节,4 个位置/标志字节和 16bit CRC 也可使用 16 进制发送, 即是 01234567890abcdef 字符集, 不允许大写的 16 进制数字, 因为可能会触发 XMODEM 和 YMODEM 程序, 因为这种 16 进制的编码类型需要检出任何的模式错误, 尤其是字符丢失, 所以 16 进制头中需要一个未定义的 32bit CRC.

返回和换行也可发送 16 进制头, 接收程序可期望得到至少一个字符, 如果是两个字符, 那么首个字符应是 CR,CR/LF 可从打印输出的信息中进行调试, 并有助于克服某些操作系统的相关问题.

XON 可附加在所有 16 进制的数据包上, 但除了 ZACK 和 ZFIN 之外,XON 可使发送器因线路噪声生成的虚假 XOFF 流控制符, 所触发的一次公共事件被释放.XON 不能在 ZACK 头之后进行发送, 它是用于包含流状态下的流控制符.XON 也不能在 ZFIN 头之后进行发送, 但允许进行会话的清理.

0 个或多个数据子包将依赖于帧类型,zshhdr 函数可发送一个 16 进制头.

图 7.4: 16 进制头

(TYPE, F3...F0, CRC-1, CRC-2 每次发送只能是两个 16 进制数 (8bit))

7.5 二进制数据子包

二进制数据子包直接跟随在二进制头之后, 每个子包可包含 0 到 1024 个 byte, 推荐长度是 2400bps 速率以下是 256byte,2400bps 速率是 512byte,4800bps 速率以上是 1024byte, 或者数据链路层已知的相对无错的传送长度.[4]

二进制数据子包中不会使用填充, 数据字节数将使用 ZDLE 编码, 并进行传送,ZDLE 和帧尾发送后, 将跟随两个或四个 ZDLE 编码的 CRC 字节,CRC 也会将数据类型和帧尾加入校验.

zsdata 函数可发送一个数据子包,zrdata 函数可接收一个数据子包.

7.6 ASCII 编码数据子包

对于 ASCII 编码的数据子包, 当前并没有定义, 它可用于服务命令, 或是 7bit 的传送环境.

[4]子包长度的调整策略是基于实际误码率的最优结果, 短包可加速错误的检出, 但会增加协议的开销

第 8 章　协议事务的预览

如 XMODEM 推荐的一样,ZMODEM 的时序也是由接收器驱动的, 发送器在任何情况下不能产生超时, 除了帧头在一个扩展的时间周期内 (又称为" 一分钟" [1]), 未被接收后的程序中止.

8.1　会话启动

启动一个"rz"文件传送会话之后, 发送程序将调用所需的文件名和选项名.

发送程序可发送一个字串"rz", 告知接收程序, 此刻将进入命名模式,rz 可激活 ZMODEM 接收程序的回传, 或是激活命令模式.

发送器可显示一个消息, 用于人工操作, 例如一个请求文件的列表等等.

之后发送器可发送一个 ZRQINIT 头,ZRQINIT 头将使之前启动的接收程序, 无延时的回传它的 ZRQINIT 头.

在交互或会话模式下, 接收应用可监控数据流的 ZDLE,B00 之后的字符将被扫描, 并可表示一个 ZRQINIT 头, 一个用于下载命令或数据的命令.

发送程序将等待接收程序的命令之后, 开始文件传送, 如果收到"C", "G" 或 NAK, 则使用 XMODEM 或 YMODEM 文件传送, 之后会使用 YMODEM 协议, 注意:ZMODEM 和 YMODEM 需要为发送程序提供文件名, 而 XMODEM 不需要.

如果是无用数据, 发送程序将重新请求, 直到会话开始 (接收程序给出正确信息).

当 ZMODEM 接收程序启动后, 将立即发送 ZRINIT 头, 初始化 ZMODEM 文件传送, 或是一个 ZCHALLENGE 头, 用于校验发送程序, 在接收程序回退使用 YMODEM 协议之前, 会在一个响应时间内 (默认是 10s) 重新发送 ZRINIT 头, 总共的重传时间周期为 40s (也就是会重传 4 次).

若接受程序收到一个 ZRINIT 头, 将会重新发送它的 ZRINIT 头, 如果发送程序收到的是一个 ZCHALLENGE 头, 会将数据放入应答 ZACK 头中的 ZP0...ZP3.

如果接收程序收到的是一个回传的 ZRINIT 头 (它自己发出的 ZRINIT 头), 则表明发送程序不可用.

最后发送程序将正确接收到 ZRINIT 头.

发送器可发送一个可选的 ZSINIT 帧, 用于定义接收程序的 Attn 序列, 或是指定完整的控制符转义.[2]

如果 ZSINIT 头中指定了 ESCCTL 或 ESC8, 可使用一个 16 进制头, 在读取后续的数据子包之前, 接收器可激活一个指定的 ESC 模式.

并且接收器还将发送一个 ZACK 头, 用于响应, 其中可包含接收程序的串行数或是 0 值.

[1] 发送命令时的特殊应用

[2] 如果接收器指定相同或更高级的转义时,ZSINIT 无需发送, 除非需要一个 Attn 序列

8.2 文件传送

之后发送器可发送一个 ZFILE 头, 其中包含 ZMODEM 转换, 管理和传输选项.[3] 在头之后, 将跟随一个 ZCRCW 数据子包, 其中包含文件名, 文件长度, 修正日期, 其他与 YMODEM Batch 相同的信息.

接收器将在指定的传输选项, 文件系统的当前状态和本地的安全需求中, 检查发送器提供的文件名, 文件长度和日期, 接收程序必须保证路径名和选项可兼容操作系统和本地的安全需求.

接收器将响应一个 ZSKIP 头, 使得发送器可批量处理对应的文件.

当接收器获得一个同名和相同长度的文件时, 可响应一个 ZCRC 头, 其中会包含一个字节计数, 将使发送器发送应答 ZCRC 头时, 包含接收器指定的文件字节数的 32bit CRC, [4]使得接收器可确认是否接受获得的文件, 并跳过它. 这个次序将会触发 ZMCRC 管理选项.

接收器可使用 ZRPOS 头, 指定文件数据起始传送的文件偏移量. 通常接收器指定的数据传送的偏移量为 0.

所以接收器可从文件中间开始传送, 因此可在丢失或媒介和系统崩溃后, 在下一次连接时, 无需请求完成整个文件的重传[5]. 如果分时系统中的文件下载变慢时, 可中止传送之后重新传送, 也不是丢失数据.

发送器可发送一个 ZDATA 二进制头 (其中包含文件偏移位), 并跟随一个和多个数据子包.

接收器可比对 ZDATA 二进制头中的文件位置, 以确定接收文件中哪些字符已经成功接收, 如果该位置未被接受, 将生成一个 ZRPOS 错误响应给发送器, 使之从文件的正确位置进行传送[6].

除非检出了一个错误, 否则使用 ZCRCG 和 CRC 中止的数据子包, 不会给出响应, 而随后的数据子包将立即中止.

接收器获得 ZCRCQ 数据子包之后, 会在无错的情况下, 在 ZACK 响应中包含文件的偏移量, 否则将使用 ZRPOS, 并包含最后成功接收的文件偏移量, 随后的数据子包将立即中止, 如果接收器未在 CANFDX 位上设定 FDX 功能, 将无法使用 ZCRCQ 数据子包

ZCRCW 数据子包必须在下一帧的发送前获取到响应, 如果接收器未在 CANOVIO 位上设定重叠 IO 的功能, 或是设定缓冲尺寸, 发送器可在发送更多数据之前, 允许接收器将数据写入缓冲.

零长度数据帧可视为一个空闲包, 以防止接收器由于超时, 而不再有效.

在没有致命错误的情况下, 发送器将会到达文件尾, 如果文件尾被包含在一个帧中, 可使用 ZCRCE 数据包, 并在无错接收的情况下不给出响应.

发送器可发出 ZEOF 头, 并包含文件最终偏移量, 它应与整个文件的字符数相同, 接收器将会进行比较, 如果接收器完成了整个文件的接收, 将关闭该文件. 如果文件已被关闭, 接收器将发送 ZRINIT 响应, 如果接收器未接收到文件的所有字节, 接收器将忽略 ZEOF, 因为新的 ZDATA 将来来到, 如果接收器无法正确关闭文件, 将发出一个 ZFERR 头.

当所有的文件都被处理后, 任何协议错误都不能阻止发送程序返回成功状态.

8.3 会话清除

发送器可发出 ZFIN 头, 关闭一个会话. 接收器将响应一个其自身的 ZFIN 头.

当发送器接收到这个应答头, 将会发送两个字符"OO"(Over and Out), 并退出操作系统或是调用它的应用. 接收器将会短暂等待"O" 字符之后, 决定是否退出.

[3]可参看后续的 ZFILE 头类型
[4]CRC 将初始化为 0xFFFFFFFF, 字节计数为 0 则表示整个文件
[5]这不能应用于已被转义的文件中
[6]若已使用了 ZMSPARS 选项, 接收器可使用它, 替换 ZDATA 头的文件位置

8.4 会话中止序列

如果接收器是在流模式中接收数据, 将在 cancel 序列发送之前, 执行 Attn 序列中止数据传送, cancel 序列将包含 8 个 CAN 字符和 10 个 backspace 字符,ZMODEM 只需要 5 个 Cancel 字符, 其他三个可视为” 保险”.

如果是被一个命令解析器接收时, 后续的 backspace 字符可尝试消除 CAN 字符的影响.

```
1 static char canistr[] = {
2   24,24,24,24,24,24,24,24,8,8,8,8,8,8,8,8,8,8,0
3 };
```

第 9 章　流技术和错误恢复

事实上在今天的主流计算机和通讯环境中, 不可能存在单一的流化模式,ZMODEM 可提供多种数据流化模式, 以适应发送环境, 接收环境和传送通道的限制.

9.1　带采样的完全流化

如接收器可重叠串行 IO 和磁盘 IO, 而发送器可对逆向通道进行采样, 且不存在等待时间, 这时可使用无 Attn 序列的完全流化, 发送器可从 ZDATA 头开始数据传送, 后续将包含 ZCRCG 数据包, 当接收器检出一个错误时, 它可执行 Attn 序列, 并发送一个 ZRPOS 头, 并包含文件中的正确偏移量.

当每次数据包的传送之后, 发送器可检查是否出现来自接收器的错误头, 为了这个功能, 发送器需要采样回传数据, 以确定是否存在 ZPAD 或 CAN 字符[1]. 流控制符 (如果存在) 将被激活.

若在发送器等待接收器的响应时, 其他字符 (可以是线路噪声) 致使计数器复位, 如果计数器溢出, 发送器将发送下一个数据包 ZCRCW, 并等待响应.

接收器发出 ZPAD 头, 可包含一些错误类型,CAN 即是暗示用户可键入 CAN 字符进行中止, 如果这些字符出现时, 将发出一个空的 ZCRCE 数据包, 通常接收器可发送一个 ZRPOS 或其他错误头, 并告知发送器从不同的地址重新传送, 或是执行其他的任务. 而 ZPAD 或 CAN 字符不太可能伪造, 因此当接收器超时后, 将发送一个 ZRPOS 头[2].

之后接收器的响应头将被读取并处理[3].

ZRPOS 可使发送器的文件偏移量恢复到正确的位置, 发送器将清空输出缓冲或网络上的所有未处理的输出数据, 而收到正确的文件偏移量之前, 接收器将丢弃无用数据, 下一次发送数据帧必须是 ZCRCW 帧, 并保证网络内存已被完全刷新.

如果接收器给出的 ZACK 头中包含的地址, 与发送器的地址不同, 将被忽略, 并等待下一个头, ZFIN, ZABORT 或 TIMEOUT 可中止会话,ZSKIP 可中止文件的处理.

对逆向通道采样时, 出现了来自接收器的其他头[4], 如果被检出,getinsync() 将再次被调用, 并读取其他的错误头, 否则重新传送, 使用一个包含文件偏移量 (也可能复位它) 的 ZDATA 头, 并跟随数据包.

9.1.1　窗口管理

当通过网络进行数据传送时, 有些网络节点会保存发送给接收器的数据, 可能是 7000 byte 甚至更多的临时存储, 如此大量的存储会出现接收器速度快于发送器的情况, 这对于 MEGAlink 和其他协议来说是致命的, 因为它依赖于接收器给出的时序上的错误通告, 但对 ZMODEM 并不是致命的, 因为它可以实现慢速的错误恢复.

为管理窗口尺寸, 发送程序将使用 ZCRCQ 数据包, 并触发接收器的 ZACK 头, 回传的 ZACK 头可告知发送器, 接收器的处理状态. 当窗口尺寸 (当前传送的文件偏移量 -接收器报告的文件偏移量) 超过某个指定值时, 发送器将等待接收器的 ZACK[5]包, 以降低窗口的尺寸.

[1] sz.c 中 rdchk() 将执行该功能
[2] 明显 ZCRCW 数据包, 可使接收器发出一个 ZACK, 如果通道存在长的传播延时, 也不会用它来传输多个帧.
[3] sz.c 中的 getinsync() 可执行该功能
[4] 采样应保证是正确的

Unix sz 版本始于 1987-5-9, 可使用"-w N" 选项控制窗口的尺寸,N 即为最大的窗口尺寸, Pro-YAM, ZCOMM 和 DSZ 的版本始于 1987-5-9, 可使用"zmodem pwN" 控制窗口的尺寸, 这些程序之前的版本都是兼容的[6].

9.2 可逆向中断的完全流化

如果逆向数据流在未进入 IO 等待时, 则不可采样的情况下, 以上模式则不可用, 有一种替代模式, 接收器在检出错误时, 可中断发送程序.

接收器可使用一个控制符, 中断信号, 或是定义的 Attn 序列, 将发送器中断, 当执行 Attn 序列后, 接收器可发出一个 16 进制的 ZRPOS 头, 强迫发送器重新发生丢失的数据.

当发送程序收到这个中断时, 可读取接收器的 16 进制头 (通常是 ZRPOS), 并触发之前介绍的动作, Unix sz.c 可使用 setjmp/longjmp 调用捕获 Attn 序列生成的中断, 而捕获的中断可激活 getinsync(), 读取接收器的错误头, 并执行相应的动作.

在标准的 SYSTEM III/V Unix 编译的 sz.c 在使用 Ctrl-C Attn 序列时, 将跟随发送器中断的 1s 暂停, 给出发送器 (Unix) 处理接收器错误头的预备时间.

9.3 窗口可变的完全流化

如果以上的方法都不适用, 希望依旧在, 如果发送器可缓冲来自接收器的响应, 发送器可使用 ZCRCQ 数据包, 并从未发出中断的接收器获取 ACK, 当足够数量的 ZCRCQ 数据包发送之后, 发送器可读取, 它的接收中断缓冲内的头.

这类方法的问题是, 将会浪费接收器错误头的响应次数, 还可能造成接收器在发送 ZRPOS 头之前, 接收器 Attn 序列对发送器的中断缓冲进行刷新.

对 XON 和 XOFF 字符的控制, 以及传递妨碍流控的 XON 字符, 应保证接收程序版本为 5-9 或后续版本.

9.4 无错通道上的完全流化

文件传输协议可在通讯中的某个时间段内实现端到端的无错通道, 在理论上证明类似通道比实际操作更容易, 随着 modem 加密器占用比 CRC 更多的 bit 时, 未检出的错误会越来越多.

ZMODEM 发送器会假定一个包含端到端流控的无错通道, 它可以在一个帧中发送整个文件, 并无需检查逆向的数据流, 如果该通道是完整透明的, 只有 ZDLE 需要转义, 那么协议的开销在长文件传送中, 平均将小于 1%[7].

9.5 分段流化

如果接收器无法重叠串行和磁盘 IO, 需要使用 ZRINIT 帧指定一个缓冲长度, 才能使得发送器不会造成溢出, 发送器可发出一个 ZCRCW 数据包, 并下一个文件分段发送之前, 等待 ZACK 头

如果发送程序支持逆向数据流采样, 或是中断, 则错误恢复将比发送大数据块的协议 (比如 YMODEM) 更快.

一个足够大的接收缓冲, 可允许实现更接近完全流化的速率, 例如当环形延时为 5s 时,16kb 的分段流化比完全流化的 ZMODEM 文件传送, 要增加 3% 的时间

[5]ZRPOS 和其他错误包将会正常处理
[6]可用于 modem 或是同步断言流的的网络
[7]对 ZDLE 的转义只占用 256 个编码之一, 以及 1024 数据包的 CRC 为 2 个字节 (如果使用 32bit CRC 则是 4 个字节)

第 10 章　ATTENTION 序列

接收程序可在检出错误后, 并需要中断发送程序时, 发送 Attn 序列.

默认的 Attn 字串为空 (也就是没有 Attn 序列), 接收程序需要在每次传送会话之前, 将 Attn 复位为空.

发送器可选择 ZSINIT 帧, 指定 Attn 序列, 而 Attn 字串会用 null 字符结尾.

两个元字符 (meta-character) 可执行该特殊功能:

- \335(八进制数) 可发送一个中断信号
- \336(八进制数) 可暂停 1s

第 11 章　帧类型

以下的数值将在 zmodem.h 中以黑体字给出, 头 (ZP0...ZP3) 中不用的 bit 和 byte 将设为 0.

11.1　ZRQINIT

可由发送程序发出, 并触发接收程序发送自己的 ZRINIT 头, 它可避免与 XMODEM 和 Kermit 传送时, 启动延时的加剧. 如果响应未在第一时间获得, 发送程序可重复接收该响应 (其中包含了 ZRQINIT).

如果程序尝试发送一个命令时,ZF0 将包含 ZCOMMAND, 否则将为 0.

11.2　ZRINIT

可由接收程序发出,ZF0 和 ZF1 的位将包含接收器的性能标志.

```
#define CANCRY      8   /* Receiver can decrypt */
#define CANFDX     01   /* Rx can send and receive true FDX */
#define CANOVIO    02   /* Rx can receive data during disk I/O */
#define CANBRK     04   /* Rx can send a break signal */
#define CANCRY    010   /* Receiver can decrypt */
#define CANLZW    020   /* Receiver can uncompress */
#define CANFC32   040   /* Receiver can use 32 bit Frame Check */
#define ESCCTL   0100   /* Receiver expects ctl chars to be escaped */
#define ESC8     0200   /* Receiver expects 8th bit to be escaped */
```

ZP0 和 ZP1 将包含接收器缓冲的字节数, 如果允许连续 IO, 将为 0 字节.

11.3　ZSINIT

发送器可发出该标志, 并以一个二进制 ZCRCW 数据子包结尾.

```
/* Bit Masks for ZSINIT flags byte ZF0 */
#define TESCCTL   0100   /* Transmitter expects ctl chars to be escaped */
#define TESC8     0200   /* Transmitter expects 8th bit to be escaped */
```

数据子包中可包含 null 结尾的 Attn 序列, 若包含结尾的 null 字符, 最大长度为 32byte.

11.4　ZACK

用于响应 ZSINIT 帧, ZCHALLENGE 头, ZCRCQ 或 ZCRCW 数据子包,ZP0 到 ZP3 可包含文件偏移量, ZCHALLENGE 的响应可包含与 ZCHALLENGE 头相同的 32bit 数.

11.5　ZFILE

该帧可表示文件传送的起始点,ZF0, ZF1, ZF2 中可包含对应的选项, 若在这些字节中包含 0, 则表示无需特殊处理, 指定给接收器的选项可覆盖发送器的选项, 除了 ZCBIN 之外, 来自发送器的 ZCBIN 可覆盖接收器获取的转换选项, 除了 ZCRESUM 之外, 来自接收器的 ZCBIN 可覆盖发送器发出的转换选项.

11.5.1 ZF0: 转换选项

如果接收器无法识别转换选项时, 应用可依赖于默认的转换方式.

- **ZCBIN**
  ” 二进制传送” - 可无条件抑制转换

- **ZCNL**
  将接收到的行尾转换成本地约定的行尾, 支持的行尾即为 CR/LF(这是大多数操作系统支持的 ASCII 字符, 除了 Unix 和 Macintosh), 或是 NL(Unix 支持). 这两种行尾约定都允许使用回车和换行/新行, 所定义的 ASCII 字符,ZMODEM ZCNL 只能使用回车实现行分隔, 在接收器中, 可对 ASCII 文本文件进行其他处理, 以及本地操作系统可应用的操作.[1]

- **ZCRECOV**
  恢复/重传已中断的文件传送,ZCRECOV 可用于更新文件的远程覆盖, 且无需重新发送已成功的旧数据, 如果存在的目的文件小于源文件, 可在接收器文件尾的偏移量处, 重新发送数据, 并附加到目的文件中. 该选项无法用于超短的源文件, 已转换的文件 (例如 ZCNL) 或是设定了单次传送选项的任务, 则无法使用传送恢复.

11.5.2 ZF1: 管理选项

如果接收器无法识别管理选项, 文件将使用正常模式进行传送.

如果接收器中并无一个与当前文件同名的文件时,ZMSKNOLOC 位将告知接收器忽略当前文件.

ZMMASK 中的 5 个位将定义以下管理选项的集合.

- **ZMNEWL**
  若无同名的目的文件将进行文件传送, 如果源文件比目的文件更新或更长时, 传送文件将覆盖目的文件.

- **ZMCRC**
  将比较源文件和目的文件, 若文件长度或属性不同时, 将进行覆盖传送.

- **ZMAPND**
  将源文件附加到已存在的目的文件的尾部.

- **ZMCLOB**
  覆盖已存在的目的文件

- **ZMDIFF**
  如果目的文件不存在, 则进行文件传送, 如果目的文件包含了不同的长度和日期, 将进行覆盖传送.

- **ZMPROT**
  保护目的文件, 只有当目的文件不存在时, 才进行文件传送.

- **ZMNEW**
  如果目的文件不存在, 如果源文件更新, 将进行覆盖传送.

11.5.3 ZF2: 传送选项

如果接收器无法实现特殊的传送选项, 为了后续的处理, 副本文件不能进行转换.

- **ZTLZW**
  Lempel-Ziv 压缩, 传送数据将使用 VAX 字节顺序,12bit 编码的 4.0 压缩版本.

- **ZTCRYPT**
  加密, 将一个初始为 null 结尾的字串作为 key, 其他细节待定.

- **ZTRLE**
  可变长度编码, 细节待定.

- **ZCRCW**
  数据包中将包含文件名, 文件长度, 修改日期和其他之前已介绍的信息.

[1] 可对 RUBOUT, NULL, Ctrl-Z 等字符进行过滤.

11.5.4 ZF3: 扩展选项

扩展选项将使用位编码.

- **ZTSPARS**
  为单个文件或是发送器可选的转发功能的特殊处理, 每个发送文件分段都被视为一个单独的帧, 而这些帧并不要求是连续的, 发送器将使用 ZCRCW 包发送每个文件分段, 并获取 ZACK, 并保证数据没有丢失,ZTSPARS 中不能使用 ZCNL

11.6 ZSKIP

可由接收器发出, 并作为 ZFILE 的响应, 可使发送器跳到下一个需传送的文件.

11.7 ZNAK

表示最后的帧头已出错 (参见 ZRPOS)

11.8 ZABORT

可由接收器发出, 并能中止用户请求的批量文件传送, 发送器可响应一个 ZFIN 序列.[2]

11.9 ZFIN

可由发送器发出, 并能中止一次 ZMODEM 会话, 接收器将响应一个自己的 ZFIN

11.10 ZRPOS

可由接收器发出, 并强制发送器从 ZP0...ZP3 给出的文件偏移量开始传送文件.

11.11 ZDATA

ZP0...ZP3 中将包含文件的偏移量, 并后跟一个或多个数据包.

11.12 ZEOF

发送器报告的文件尾部,ZP0...ZP3 将包含文件尾部的偏移量.

11.13 ZFERR

在读取或写入文件时出错, 等同于 ZABORT

11.14 ZCRC

接收器和发送器使用的 CRC 多项式,ZP0...ZP3 将包含该多项式.

11.15 ZCHALLENGE

请求发送器在 ZACK 帧,ZP0...ZP3 中回传一个随机数, 该帧是由接收器发给发送器, 用于校验彼此的连接, 一旦发现是伪造数据或是木马消息时, 将停止运行.

[2]或是服务模式下的 ZCOMPL

11.16 ZCOMPL

请求完成.

11.17 ZCAN

它是 gethdr() 对会话终止序列的响应, 并且是一个伪帧.

11.18 ZFREECNT

发送器可用该帧, 请求一个 ZACK 帧, 并在其 ZP0...ZP3 中包含接收器文件系统中空闲的字节数, 如果该值为 0, 则表示空闲空间未知.

11.19 ZCOMMAND

将使用二进制帧发送 ZCOMMAND,ZF0 将为 0 或是 ZCACK1(参见以下介绍)

其后跟随的 ZCRCW 数据包, 其 ASCII 文本命令字串是以 null 字符结尾, 如果该命令需要在接收器的操作系统中执行 (例如 shell), 它的首字符必须是"!", 否则该命令只能在应用程序中执行.

如果接收器确认该命令非法或是一个错误格式, 将立即响应 ZCOMPL 头, 并在 ZP0...ZP3 包含错误码.

如果 ZF0 为 ZCACK1, 接收器将立即响应一个 ZCOMPL 头, 并将状态位设为 0.

否则只有当操作完成时, 接收器才会响应 ZCOMPL 头. 完成命令的退出状态将保存在 ZP0...ZP3,0 值表示命令已经完成.

如果命令引发了一次文件传送, 命令发送器将看见, 来自其他计算机数据发送的 ZRQINIT 帧.

发送器将会检查 ZRQINIT 头中的 ZF0, 以确认它是否是返回的 ZRQINIT 头, 若对于发送器来说, 它是非法的, 将会发送 ZCOMMAND, 命令接收器重新发送.

如果接收器无法实现命令下载, 它将在标准的错误输出显示该命令, 并返回一个 ZCOMPL 头.

第 12 章　会话事务的例子

12.1　一次简单的文件传送

一次简单的传送, 将包含一个文件, 无错误, 无加密, 重叠 IO:

发送器	接收器
”rz”/ZRQINIT(0)	-
-	ZRINIT
ZFILE	-
-	ZRPOS
ZDATA data .../ZEOF	-
-	ZRINIT
ZFIN	-
-	ZFIN
OO	-

12.2　加密和命令下载

发送器	接收器
”rz”/ZRQINIT(ZCOMMAND)	
	ZCHALLENGE(random-number)
ZACK(same-number)	
	ZRINIT
ZCOMMAND, ZDATA	
	(Performs Command)/ZCOMPL
ZFIN	
	ZFIN
OO	

第 13 章　ZFILE 帧的文件信息

ZMODEM 发送的 ZFILE 帧, 与 YMODEM Batch 发送的 0 号块, 都包含了相同的文件信息.

注意: 路径名 (文件名) 是强制要求.

路径名

路径名 (通常为文件名) 将使用一个 null 结尾的 ASCII 字串表示, 它的格式可被 MSDOS(TM) 函数和 C 库函数 fopen 处理, 汇编语言的例子如下:

```
DB    'foo.bar',0
```

在路径名中不能包含空格, 通常只有文件名会被发送, 除非发送器选定了 YAM 参数, 才会发送完整的路径名, 而驱动标记符 (A:, B: 等) 通常不会发送.

文件名的规则

- 文件名可转换成小写字母, 除非发送系统支持文件名的大小写转换, 这可方便一些系统 (比如 Unix) 用户保存.
- 接收器适用于小写或大写文件名.
- 在不同操作系统中传送文件时, 文件名则必须被两边的操作系统接收, 如果不行, 则应进行必要的转换, 如果转换不成功, 接收程序应给出一个新文件名.

如果包含目录, 应使用/分隔, 例如”subdir/foo” 是接受的, 而”subdir\foo” 不能接受.

文件长度

长度和随后的域都是可选参数[1]. 长度将是一个十进制字串保存, 将使用文件的字节数表示.

ZMODEM 接收器将会使用文件长度进行评估, 也可用于评估传送时间, 以及对比空余的磁盘容量, 接收文件的实际长度将由数据传送器确定, 若在传送开始后, 文件增大了, 这时所有数据都将被传送.

修改日期

修改日期将与文件长度使用一个空格分隔.

该日期是可选参数, 可以不传递该参数.

修改日期将使用一个八进制数, 并从 1970-1-1(Universal Coordinated Time (GMT)) 开始的秒数计算时间. 日期为 0 则表示修改日期未知, 可使用文件的接收日期.

标准格式的选择可消除不同时区传送之间的模糊含义.

文件模式

文件模式将与修改日期使用一个空格分隔, 文件模式会使用一个八进制字串保存. 除非该文件来自于 Unix 系统, 文件模式可设为 0,rz(1) 将会检查 0x8000 位的文件模式, 它可表示 Unix 常规文件的类型, 也可确认是否传递给另一个 Unix 系统, 或是要进行转换.

串行号

串行号将与文件模式使用一个空格分隔, 而串行号将使用一个八进制字串保存, 程序可忽略该串行号, 或将其设为 0, 接收器对于该域的使用也是可选的.

[1] 域不可以被跳过

文件号
即是指后续需传送的剩余文件, 该域将使用空格与之前的域进行分割, 可使用十进制数, 并包含在当前文件中, 该域可被检查, 因此不正确的值将导致数据的丢失, 接收器可选用该域.

字节号
即是指后续需传送的字节, 该域将使用空格与之前的域进行分割, 可使用十进制数, 并包含在当前文件中, 该域可被检查, 因此不正确的值将导致数据的丢失, 接收器可选用该域.

文件类型
文件类型可使用空格分隔, 并使用十进制数表示, 当前的定义值为:
0: 表示顺序文件 -无特殊类型
1: 其他定义的类型
接收器可选用该域.

文件信息将使用一个 null 结尾, 如果只需要发送路径名, 路径名需要使用 2 个 null 结尾, 文件信息包的长度, 和后续的 2 个 null 字符, 不能超过 1024 个字节, 一个典型的长度应小于 64 字节.

第 14 章　性能

14.1　兼容性

大量的测试已证明 ZMODEM 兼容于卫星环路, 包交换网络, 微型计算机系统, 微控制器, 固定和错误修正的缓冲 modem 在 75-19200bps 速率,ZMODEM 可节省逆向通道的带宽, 并允许 modem 在两个方向上动态分配带宽, 已达到最优化的速度.

14.2　速率

在两台 PC-XT 计算机之间, 并基于家用 Telenet 链接, 在 1200 波特率下,SuperKermit 可提供的 72 字符/秒,YMODEM-k 可提供 85 字符/秒, 而 ZMODEM 可提供 113 字符/秒,XMODEM 无法测试, 但它肯定会更慢.

最近的测试是在 IBM PC(4.7 mHz V20) 上下载巨大的二进制文件, 并运行在 YAMK 16.30, 使用 32 bit CRC 的查表运算, 可在直连的 19200bps 下获得 1870 字符/秒的速率.

对 TELEBIT TrailBlazer modem 的测试, 在大文件的传送中可获取 1400 字符的速率, 如果对文件进行压缩, 则可获得 2000 字符的速率.

14.3　错误恢复

有些测试可获得 ZMODEM 协议的错误恢复性能, 当使用 SCO SYS V Xenix 或 DOS 3.1 的 PC-AT 与使用 DOS 2.1 PC 直连的情况下可获取 9600bps, 或可使用无缓冲的 1200bps 的拨号 modem,ZMODEM 软件可在配置完 1024 字节的数据包尺寸后, 速率可提升到 2400bps 以上.

由于在正常的文件传送中不存在延时, 每个文件的握手比 YMODEM 更快, 只剩下程序用于更新日志文件的时间.

在文件传送中, 接收器将会使用一个短包来指示一个 CRC 错误, 而中断序列通常会在发送器完成下一个数据包之前被接收, 这将会减少数据包重传的时间丢失.

速率的下降将意味着通道延时的增加, 在发现错误时, 会有更多的数据包被丢弃.

一个长时间的噪声突发, 会影响到发送器和接收器, 当发送器对于中断序列的接收受到影响后, 会使发送器进入静默, 直到接收器超时 10s. 如果环路延时超过接收器的 10s 超时, 对于错误的恢复将非常困难.

噪声对于发送器的影响可忽略, 因为这是一个公网的问题. 噪声伪造的 XOFF 字符会停止发送器, 除非接收器超时, 并发送了一个中断序列, 并包含一个 XON.

总而言之 ZMODEM 的性能与 X.PC 和 SuperKermit 相当, 短促的突发噪声只会使少量数据被重新发送, 长突发噪声 (比如脉冲拨号噪声) 会导致一个超时错误, 并数据流进行恢复.

第 15 章　包交换网络

流控制可用于消息打印和方向设定, 文件流传送协议. 一个不透明的流控制不被 XMODEM 和 YMODEM 传送所支持,XMODEM 和 YMODEM 协议只能用于 8bit 的透明传送.

最好的流控制 (X.25 或硬件 CTS 无效时) 不会" 吃掉" 所有的字符, 当 PAD 缓冲已满, XOFF 将发出, 当缓冲接近满载时, 将发送 XON, 否则网络既不会吃 XON 也不会吃 XOFF 字符.

在 Telenet 中, 可以设定 CCIT X3 5:1 和 12:0, 为了更好的速率, 可发送 64 个参数 (高级 ACK). 转发数据包可以是一个完整的 128byte, 或是在稳定延时之后.

With PC-Pursuit, it is sufficient to set parameter 5 to 1 at both ends after one is connected to the remote modem.

<ENTER>@<ENTER>
set 5:1<ENTER>
rst? 5:1<ENTER>
cont<ENTER>

Unfortunately, many PADs do not accept the "rst?" command.

For YMODEM, PAD buffering should guarantee that a minimum of 1040 characters can be sent in a burst without loss of data or generation of flow control characters. Failure to provide this buffering will generate excessive retries with YMODEM.

表 15.1: Network and Flow Control Compatibility

Connectivity	**Interactive**	**XMODEM**	**WXMODEM**	**SUPERKERMIT**	**ZMODEM**
Direct Connect	YES	YES	YES	YES	YES
Network, no FC	no	YES	(4)	(6)	YES (1)
Net, transparent FC	YES	YES	YES	YES	YES
Net, non-trans. FC	YES	no	no (5)	YES	YES
Network, 7 bit	YES	no	no	YES (2)	YES (3)

(1) ZMODEM can optimize window size or burst length for fastest transfers.
(2) Parity bits must be encoded, slowing binary transfers.
(3) Natural protocol extension possible for encoding data to 7 bits.
(4) Small WXMODEM window size may may allow operation.
(5) Some flow control codes are not escaped in WXMODEM.
(6) Kermit window size must be reduced to avoid buffer overrun.

第 16 章　性能对照表

”Round Trip Delay Time” includes the time for the last byte in a packet to propagate through the operating systems and network to the receiver, plus the time for the receiver’s response to that packet to propagate back to the sender.

The figures shown below are calculated for round trip delay times of 40 milliseconds and 5 seconds. Shift registers in the two computers and a pair of 212 modems generate a round trip delay time on the order of 40 milliseconds. Operation with busy timesharing computers and networks can easily generate round trip delays of five seconds. Because the round trip delays cause visible interruptions of data transfer when using XMODEM protocol, the subjective effect of these delays is greatly exaggerated, especially when the user is paying for connect time.

A 102400 byte binary file with randomly distributed codes is sent at 1200 bps 8 data bits, 1 stop bit. The calculations assume no transmission errors. For each of the protocols, only the per file functions are considered. Processor and I/O overhead are not included. YM-k refers to YMODEM with 1024 byte data packets. YM-g refers to the YMODEM ”g” option. ZMODEM uses 256 byte data subpackets for this example. SuperKermit uses maximum standard packet size, 8 bit transparent transmission, no run length compression. The 4 block WXMODEM window is too small to span the 5 second delay in this example; the resulting thoughput degradation is ignored.

For comparison, a straight ”dump” of the file contents with no file management or error checking takes 853 seconds.

表 16.1: Protocol Overhead Information
(102400 byte binary file, 5 Second Round Trip)

Protocol	**XMODEM**	**YM-k**	**YM-g**	**ZMODEM**	**SKermit**	**WXMODEM**
Protocol Round Trips	804	104	5	5	5	4
Trip Time at 40ms	32s	4s	0	0	0	0
Trip Time at 5s	4020s	520s	25s	25s	25	20
Overhead Characters	4803	603	503	3600	38280	8000
Line Turnarounds	1602	204	5	5	2560	1602
Transfer Time at 0s	893s	858s	857s	883s	1172s	916s
Transfer Time at 40ms	925s	862s	857s	883s	1172s	916s
Transfer Time at 5s	5766s	1378s	882s	918s	1197s	936s

图 16.1: Transmission Time Comparison (102400 byte binary file, 5 Second Round Trip)
********** XMODEM
********** YMODEM-K
********* SuperKermit (Sliding Windows)
****** ZMODEM 16kb Segmented Streaming
****** ZMODEM Full Streaming
****** YMODEM-G

表 16.2: Local Timesharing Computer Download Performance

Command	**Protocol**	**Time/HD**	**Time/FD**	**Throughput**	**Efficiency**

kermit -x	Kermit	1:49	2:03	327	34%
sz -Xa phones.t	XMODEM	1:20	1:44	343	36%
sz -a phones.t	ZMODEM	:39	:48	915	95%

Times were measured downloading a 35721 character text file at 9600 bps, from Santa Cruz SysV 2.1.2 Xenix on a 9 mHz IBM PC-AT to DOS 2.1 on an IBM PC. Xenix was in multiuser mode but otherwise idle. Transfer times to PC hard disk and floppy disk destinations are shown.

C-Kermit 4.2(030) used server mode and file compression, sending to Pro-YAM 15.52 using 0 delay and a "get phones.t" command.

Crosstalk XVI 3.6 used XMODEM 8 bit checksum (CRC not available) and an "ESC rx phones.t" command. The Crosstalk time does not include the time needed to enter the extra commands not needed by Kermit and ZMODEM.

Professional-YAM used ZMODEM AutoDownload. ZMODEM times included a security challenge to the sending program.

表 16.3: File Transfer Speeds

Prot	**file**	**bytes**	**bps**	**ch/sec**	**Notes**
X	jancol.c	18237	2400	53	Tymnet PTL 5/3/87
X	source.xxx	6143	2400	56	Source PTL 5/29/87
X	jancol.c	18237	2400	64	Tymnet PTL
XN	tsrmaker.arc	25088	1200	94	GEnie PTL
B/ovth	emaibm.arc	51200	1200	101	CIS PTL MNP
UUCP	74 files, each>	7000	1200	102	(Average)
ZM	jancol.c	18237	1200	112	DataPac(604-687-7144)
X/ovth	emaibm.arc	51200	1200	114	CIS PTL MNP
ZM	emaibm.arc	51200	1200	114	CIS PTL MNP
ZM	frombyte87.txt	62506	1200	117	BIX
SK	source.xxx	6143	2400	170	Source PTL 5/29/87
ZM	jancol.c	18237	2400	221	Tymnet PTL upl/dl
B/ovth	destro.gif	33613	2400	223	CIS/PTL 9-12-87
ZM	jancol.c	18237	2400	224	Tymnet PTL
ZM	tp40kerm.arc	112640	2400	224	BIX 6/88
ZM	readme.lis	9466	2400	231	BIX 6/88
ZM	jancol.c	18237	2400	226/218	TeleGodzilla upl
ZM	jancol.c	18237	2400	226	Tymnet PTL 5/3/87
ZM	zmodem.ov	35855	2400	227	CIS PTL node
C	jancol.c	18237	2400	229	Tymnet PTL 5/3/87
ZM	jancol.c	18237	2400	229/221	TeleGodzilla
ZM	zmodem.ov	35855	2400	229	CIS PTL node upl
ZM	jancol.c	18237	2400	232	CIS PTL node
QB	gifeof.arc	32187	2400	232	CIS PTL node
ZM	pcpbbs.txt	38423	2400	534	MNP Level 5
ZM	mbox	473104	9600	948/942	TeleGodzilla upl
ZM	zmodem.arc	318826	14k	1357/1345	TeleGodzilla
ZM	mbox	473104	14k	1367/1356	TeleGodzilla upl
ZM	c2.doc	218823	38k	3473	Xenix 386 TK upl
ZM	mbox -a	511893	38k	3860	386 Xenix 2.2 Beta
ZM	c.doc	218823	57k	5611	AT Clone & 386

Times are for downloads unless noted. Where two speeds are noted, the faster speed is reported by the receiver because its transfer time calculation excludes the security check and transaction log file processing. The TeleGodzilla computer is a 4.77 mHz IBM PC with a 10 MB hard disk. The 386 computer uses an Intel motherboard at 18

mHz 1ws. The AT Clone (QIC) runs at 8 mHz 0ws.

Abbreviations:
B Compuserve B Protocol
QB Compuserve Quick-B/B+ Protocol
B/ovth CIS B with Omen Technology OverThruster(TM)
C Capture DC2/DC4 (no protocol)
K Kermit
MNP Microcom MNP error correcting SX/1200 modem
PTL Portland Oregon network node
SK Sliding Window Kermit (SuperKermit) w=15
X XMODEM
XN XMODEM protocol implemented in network modes
X/ovth XMODEM, Omen Technology OverThruster(TM)
ZM ZMODEM

表 16.4: Protocol Checklist

Item	XMODEM	YMDM-k	YMDM-g	ZMODEM	SK	Etc.
IN SERVICE	1977	1982	1985	1986	1985	?
VENDORS	??	??	>20	>20	??	1
HOST AVAILABILITY						
Compuserve	YES	-	-	YES	-	-
BIX	YES	-	-	YES	YES	-
Portal			YES	-	-	SOON
The Source	YES	-	-	-	YES	-
USER FEATURES						
User Friendly	-	-	-	YES	(10)	-
Commands/batch	2*N	2	2	1	1(1)	
Commands/file	2	0	0	0	0	
Command Download	-	-	-	YES	YES(6)	-
Menu Compatible	-	-	-	YES	-	-
Crash Recovery	-	-	-	YES	-	??
File Management	-	-	-	YES	-	some
Security Check	-	-	-	YES	-	-
COMPATIBILITY						
Dynamic Files	YES	-	-	YES	YES	?
Packet SW NETS	-	-	-	YES	YES	?
7 bit PS NETS	-	-	-	(3)	YES	?
Old Mainframes	-	-	-	(3)	YES	?
ATTRIBUTES						
Reliability(5)	fair	fair(5)	none	BEST	good	?
Streaming	-	-	YES	YES	YES	
Overhead(2)	7%	1%	1%	4%(8)	30%	
Faithful Xfers	-	YES(7)	YES(7)	YES	YES	?
Preserve Date	-	YES	YES	YES	-	?
COMPLEXITY						
No-Wait Sample	-	-	-	opt	REQD	REQD
Ring Buffers	-	-	-	opt	REQD	REQD
Complexity	LOW(5)	LOW(5)	LOW	MED	HIGH	?
EXTENSIONS						
Server Operation	-	-	-	YES(4)	YES	?
Multiple Threads	-	-	-	future	-	-

Etc: Relay, BLAST, DART

NOTES:

(1) Server mode or Omen Technology Kermit AutoDownload
(2) Character count, binary file, transparent channel
(3) Future enhancement provided for
(4) AutoDownload operation
(5) Omen Technology's Cybernetic Data Recovery(TM) improves XMODEM and YMODEM reliability with complex proprietary logic.
(6) Server commands only
(7) Only with True YMODEM(TM)
(8) More then 3% from protected network control characters
(9) With Segmented Streaming
(10) With Pro-YAM extensions

第 17 章　将来的扩展

- Compatibility with 7 bit networks
- Server/Link Level operation: An END-TO-END error corrected program to program session is required for financial and other sensitive applications.
- Multiple independent threads
- Bidirectional transfers (STEREO ZMODEM)
- Encryption
- Compression
- File Comparison
- Selective transfer within a file (e.g., modified segments of a database file)
- Selective Retransmission for error correction

第 18 章　版本演变

10-14-88 Pascal source code now available in Phil Burn's PibTerm v4.2. 6-24-88 An exception to the previously unconditional ZCBIN override: a ZCRESUM specified by the receiver need not be overridden by the sender's ZCBIN.

11-18-87 Editorial improvements

10-27-87 Optional fields added for number of files remaining to be sent and total number of bytes remaining to be sent.

07-31-1987 The receiver should ignore a ZEOF with an offset that does not match the current file length. The previous action of responding with ZRPOS caused transfers to fail if a CRC error occurred immediately before end of file, because two retransmission requests were being sent for each error. This has been observed under exceptional conditions, such as data transmission at speeds greater than the receiving computer's interrupt response capabilitiy or gross misapplication of flow control.

Discussion of the Tx backchannel garbage count and ZCRCW after error ZRPOS was added. Many revisions for clarity.

07-09-87 Corrected XMODEM's development date, incorrectly stated as 1979 instead of the actual August 1977. More performance data was added.

05-30-87 Added ZMNEW and ZMSKNOLOC

05-14-87 Window management, ZACK zshhdr XON removed, control character escaping, ZMSPARS changed to ZXPARS, editorial changes.

04-13-87 The ZMODEM file transfer protocol's public domain status is emphasized.

04-04-87: minor editorial changes, added conditionals for overview version.

03-15-87: 32 bit CRC added.

12-19-86: 0 Length ZCRCW data subpacket sent in response to ZPAD or ZDELE detected on reverse channel has been changed to ZCRCE. The reverse channel is now checked for activity before sending each ZDATA header.

11-08-86: Minor changes for clarity.

10-2-86: ZCNL definition expanded.

9-11-86: ZMPROT file management option added.

8-20-86: More performance data included.

8-4-86: ASCII DLE (Ctrl-P, 020) now escaped; compatible with previous versions. More document revisions for clarity.

7-15-86: This document was extensively edited to improve clarity and correct small errors. The definition of the ZMNEW management option was modified, and the ZMDIFF management option was added. The cancel sequence

was changed from two to five CAN characters after spurious two character cancel sequences were detected.

第 19 章　其他信息

Please contact Omen Technology for troff source files and typeset copies of this document.

19.1 TeleGodzilla BBS

More information may be obtained by calling the TeleGodzilla bulletin board at 503-621-3746. TeleGodzilla supports 19200 (Telebit PEP), 2400 and 1200 bps callers with automatic speed recognition.

Relevant files include YZMODEM.ZOO, YAMDEMO.ZOO, YAMHELP.ZOO, ZCOMMEXE.ARC, ZCOMMDOC.ARC, ZCOMMHLP.ARC.

Useful commands for TeleGodzilla include ”menu”, ”dir”, ”sx file (XMODEM)”, ”kermit sb file ...”, and ”sz file ...”.

19.2 Unix UUCP Access

UUCP sites can obtain the current version of this file with
uucp omen!/u/caf/public/zmodem.doc /tmp

A continually updated list of available files is stored in /usr/spool/uucppublic/FILES.
uucp omen! uucp/FILES /usr/spool/uucppublic

The following L.sys line allows UUCP to call site ”omen” via Omen’s bulletin board system ”TeleGodzilla”. TeleGodzilla is an instance of Omen Technology’s Professional-YAM in host operation, acting as a bulletin board and front ending a Xenix system.

In response to TeleGodzilla’s ”Name Please:” (e:–e:), uucico gives the Pro-YAM ”link” command as a user name. Telegodzilla then asks for a link password (d:). The password (Giznoid) controls access to the Xenix system connected to the IBM PC’s other serial port. Communications between Pro-YAM and Xenix use 9600 bps; YAM converts this to the caller’s speed.

Finally, the calling uucico sees the Xenix ”Login:” message (n:– n:), and logs in as ”uucp”. No password is used for the uucp account.

omen Any ACU 2400 1-503-621-3746 e:–e: link d: Giznoid n:–n: uucp

第 20 章　ZMODEM 开发

A copy of this document, a demonstration version of Professional-YAM, a flash-up tree structured help file and processor, are available in YZMODEM.ZOO on TeleGodzilla and other bulletin boards. This file must be unpacked with LOOZ.EXE, also available on TeleGodzilla. YZMODEM.ZOO may be distributed provided none of the files are deleted or modified without the written consent of Omen Technology.

TeleGodzilla and other bulletin boards also feature ZCOMM, a shareware communications program. ZCOMM includes Omen Technology's TurboLearn(TM) Script Writer, ZMODEM, Omen's highly acclaimed XMODEM and YMODEM protocol support, Sliding Windows Kermit, several traditional protocols, a powerful script language, and the most accurate VT100/102 emulation available in a usr supported program. The ZCOMM files include:

- **ZCOMMEXE.ARC** Executable files and beginner's telephone directory
- **ZCOMMDOC.ARC** "Universal Line Printer Edition" Manual
- **ZCOMMHLP.ARC** Tree structured Flash-UP help processor and database

C source code and manual pages for the Unix/Xenix rz and sz programs are available on TeleGodzilla in RZSZ.ZOO. This ZOO archive may be unpacked with LOOZ.EXE, also available on TeleGodzilla. Most Unix like systems are supported, including V7, Sys III, 4.x BSD, SYS V, Idris, Coherent, and Regulus.

RZSZ.ZOO includes a ZCOMM/Pro-YAM/PowerCom script ZUPL.T to upload the small (178 lines) YMODEM bootstrap program MINIRB.C without a file transfer protocol. MINIRB uses the Unix stty(1) program to set the required raw tty modes, and compiles without special flags on virtually all Unix and Xenix systems. ZUPL.T directs the Unix system to compile MINIRB, then uses it as a bootstrap to upload the rz/sz source and manual files.

Pascal source code for ZMODEM support is available in PibTerm v4.2 written by Phil Burns.

The PC-DOS EXEC-PC, QuickBBS, Opus and Nochange bulletin boards support ZMODEM. Integrated ZMODEM support for the Collie bulletin board program is planned. Most of the PC-DOS bulletin board programs that lack integrated ZMODEM support ZMODEM with external modules (DSZ, etc.).

The BinkleyTerm, Dutchie and D'Bridge email systems support ZMODEM as their primary protocol.

The IN-SYNCH PC-DOS Teleconferencing system uses ZMODEM.

The LAN modem sharing program Line Plus has announced ZMODEM support.

Many programs have added direct ZMODEM support, including Crosstalk Mark IV, and Telix 3.

Most other PC-DOS communications programs support external ZMODEM via Omen Technology's DSZ, including PibTerm, Qmodem SST and BOYAN.

The ZMDM communications program by Jwahar Bammi runs on Atari ST machines.

The Online! and A-Talk Gold programs for the Amiga support ZMODEM.

The Byte Information eXchange supports ZMODEM. The Compuserve Information Service has ported the Unix rz/sz ZMODEM programs to DECSYSTEM 20 assembler, and has announced future support for ZMODEM.

20.1 添加 ZMODEM 到 DOS 程序

DSZ is a small shareware program that supports XMODEM, YMODEM, and ZMODEM file transfers. DSZ is designed to be called from a bulletin board program or another communications program. It may be called as
dsz port 2 sz file1 file2
to send files, or as
dsz port 2 rz
to receive zero or more file(s), or as
dsz port 2 rz filea fileb

to receive two files, the first to filea and the second (if sent) to fileb. This form of dsz may be used to control the pathname of incoming file(s). In this example, if the sending program attempted to send a third file, the transfer would be terminated.

Dsz uses DOS stdout for messages (no direct CRT access), acquires the COMM port vectors with standard DOS calls, and restores the COMM port's interrupt vector and registers upon exit.

Further information on dsz may be found in dsz.doc and the ZCOMM or Pro-YAM user manuals.

第 21 章　YMODEM 开发

The Unix rz/sz programs support YMODEM as well as ZMODEM. Most Unix like systems are supported, including V7, Sys III, 4.2 BSD, SYS V, Idris, Coherent, and Regulus.

A version for VAX-VMS is available in VRBSB.SHQ, in the same directory.

Irv Hoff has added 1k packets and YMODEM transfers to the KMD and IMP series programs, which replace the XMODEM and MODEM7/MDM7xx series respectively. Overlays are available for a wide variety of CP/M systems.

Many other programs, including MEX-PLUS and Crosstalk Mark IV also support some of YMODEM's features.

Questions about YMODEM, the Professional-YAM communications program, and requests for evaluation copies may be directed to:

Chuck Forsberg
Omen Technology Inc
17505-V Sauvie Island Road
Portland Oregon 97231
VOICE: 503-621-3406 :VOICE
Modem (TeleGodzilla): 503-621-3746
Usenet: ...!tektronix!reed!omen!caf
Compuserve: 70007,2304
Source: TCE022

第 22 章　致谢

The High Reliability Software(TM), TurboLearn Script Writer(TM), Cybernetic Data Recovery(TM), AutoDownload(TM), Intelligent Crash Recovery(TM), Error Containment(TM), Full Time Capture(TM), True YMODEM(TM), OverThruster(TM), Password Guardian(TM), CryptoScript(TM), and TurboDial(TM) are Omen Technology trademarks.

ZMODEM was developed for the public domain under a Telenet contract. The ZMODEM protocol descriptions and the Unix rz/sz program source code are public domain. No licensing, trademark, or copyright restrictions apply to the use of the protocol, the Unix rz/sz source code and the ZMODEM name.

Encouragement and suggestions by Thomas Buck, Ward Christensen, Earl Hall, Irv Hoff, Stuart Mathison, and John Wales, are gratefully acknowledged. 32 bit CRC code courtesy Gary S. Brown.

第 23 章　相关文件

The following files may be useful while studying this document:

YMODEM.DOC Describes the XMODEM, XMODEM-1k, and YMODEM batch file transfer protocols. This file is available on TeleGodzilla as YMODEM.DQC.

zmodem.h Definitions for ZMODEM manifest constants

rz.c, sz.c, rbsb.c Unix source code for operating ZMODEM programs.

rz.1, sz.1 Manual pages for rz and sz (Troff sources).

zm.c Operating system independent low level ZMODEM subroutines.

minirb.c A YMODEM bootstrap program, 178 lines.

RZSZ.ZOO,rzsz.arc Contain the C source code and manual pages listed above, plus a ZCOMM script to upload minirb.c to a Unix or Xenix system, compile it, and use the program to upload the ZMODEM source files with error checking.

DSZ.ZOO,dsz.arc Contains DSZ.COM, a shareware X/Y/ZMODEM subprogram, DESQview ”pif” files for background operation in minimum memory, and DSZ.DOC.

ZCOMM*.ARC Archive files for ZCOMM, a powerful shareware communications program.