# X/YModem

## 1.巴别塔

一座“YMODEM巴别塔”降临到了微计算社区，带来了困惑、沮丧、电话费膨胀和工时浪费。不幸的是，我（查克·福斯伯格）是造成这一混乱的部分原因。

作为20世纪80年代早期batch和1kxmodem扩展的作者，我假设本文档早期版本的读者将在他们的编程技能和计算环境允许的情况下实现尽可能多的YMODEM协议。这证明了一个相当幼稚的假设，因为受竞争压力驱使的程序员将尽可能少地实现YMODEM。有些人已经采用了YMODEM的任何部分呼吁他们，将他们应用到MODEM7批处理、Telink、XMODEM或其他什么，并调用结果YMODEM。

Jeff Garbers（相声软件包开发总监）说了这么多：“有了公共领域的协议，任何想和协议打交道的人都可以继续。”

包含从YMODEM.DOC派生的更改示例的文档增加了混淆。在一个实例中，一些自封的历史重写器将YMODEM.DOC图1中的标题从“1024字节数据包”更改为“YMODEM/CRC文件传输协议”。该文档中显示的XMODEM和YMODEM示例都不正确。

为了结束这种混乱，我们必须“非常清楚”地说明YMODEM代表什么，正如沃德·克里斯滕森在1985年创造的术语中所定义的那样。

对于阅读、理解和尊重沃德对YMODEM定义的大多数人，我对由此带来的不便表示歉意。

## 1.1定义

ARC是一个程序，它将一个或多个文件压缩到一个归档文件中，并从这些归档文件中提取文件。

XMODEM引用了由Ward Christensen的1977 MODEM.ASM程序引入的文件传输规范，为远程CP/M（RCPM）系统对MODEM.ASM的一种改编。它也被称为MODEM或MODEM2协议。一些不知道MODEM7不寻常的批处理文件模式的人称它为MODEM7。其他别名包括“CP/M用户组”和“TERM II FTP 3”。XMODEM之所以流行，一部分是因为它与众不同，另一部分是因为媒体对它感兴趣

通过“XMODEM”命令访问的公告板和RCPM系统。该协议因其通用性、简单性和合理的性能而受到所有严肃通信程序的支持。

**XMODEM/CRC**将XMODEM的1字节校验和替换为2字节循环冗余校验（CRC-16），提供了现代的错误检测保护。

**XMODEM-1k**是指具有1024字节数据块的XMODEM/CRC协议。

**YMODEM**指的是XMODEM/CRC（可选1k块）协议，具有如下所述的批传输。

**YMODEM-g**是指下面描述的流式YMODEM变体。

**True YMODEM（TM）**为了整理巴别塔的YMODEM，Omen Technology将术语True YMODEM（TM）注册为商标，以表示本文档中描述的完整YMODEM协议，包括路径名、长度，修改日期在0块中传输。请联系奥曼科技公司，以证明程序符合YMODEM（TM）的要求。

ZMODEM在一个新的协议中使用了熟悉的XMODEM/CRC和YMODEM技术，该协议提供了适合当代数据通信的可靠性、吞吐量、文件管理和用户便利性。

与ARC类似，ZOO是一个将一个或多个文件压缩成“ZOO归档”的程序。ZOO支持许多不同的操作系统，包括Unix和VMS。

## 2.2最低要求

所有声称支持YMODEM的程序必须满足以下最低要求：

+发送程序应发送块0中的路径名（文件名）。

+路径名应为空终止ASCII字符串，如下所述。

对于那些懒得阅读整个文档的人：

+除非特别要求，否则只发送文件名部分。

+未发送驱动器号。

+文件名中不区分大小写字母的系统应仅以小写形式发送路径名。

+除非显式重写，否则接收程序应使用此路径名作为接收的文件名。

+当接收程序接收到此块并成功打开输出文件时，它应使用ACK字符确认此块，然后继续进行正常的XMODEM文件传输，从接收方发送的“C”或NAK开始。

+发送程序应使用CRC-16来响应“C”路径名nak，否则使用8位校验和。

+接收程序必须在接收的每个文件中接受128和1024字节块的任意混合。发送程序可以在1024和128字节块之间任意切换。

+发送程序不得更改未确认块的长度。

+在每个文件的末尾，发送程序最多应发送十次EOT，直到收到ACK字符为止（这是XMODEM规范的一部分）

+传输会话的结束应由空（空）路径名表示，此路径名块应与其他路径名块一样得到确认。

不满足所有这些要求的程序与YMODEM不兼容，不应描述为支持YMODEM。

满足这些最低要求并不能保证在压力下进行可靠的文件传输。需要特别注意XMODEM的单字符监控消息，这些消息很容易被传输错误损坏。

## 3.为什么选择YMODEM？

自从五年前的发展以来，Ward-Christensen调制解调器协议已经使各种各样的计算机系统能够交换数据，几乎没有一个通信程序不支持这个协议。

计算机、调制解调器和网络的进步揭示了原始协议中的一些弱点：

+当与分时系统、分组交换网络、卫星电路和缓冲（纠错）调制解调器一起使用时，短块长度导致吞吐量下降。

+8位算术校验和和其他方面允许线路损伤干扰可靠、准确的传输。

+每个命令只能发送一个文件。文件名必须给出两次，先给发送程序，然后再给接收程序。

+传输的文件可以累积多达127个无关字节。

+文件的修改日期已丢失。

多年来已经制定了许多其他协议，但迄今为止没有一个协议取代XMODEM：

+由于缺乏公共领域文档和示例程序，Blast、Relay等专有协议与其供应商的财富紧密相连。

+复杂性阻碍了BISYNC、SDLC、HDLC、X.25和X.PC协议的广泛应用。

+性能上的折衷和复杂性限制了Kermit协议的流行，Kermit协议是为允许在不利于XMODEM的环境中进行文件传输而开发的。

XMODEM协议扩展和YMODEM批处理解决了其中一些弱点，同时保持了XMODEM的大部分简单性。

YMODEM由公共域程序YAM（CP/M）、YAM（CP/M-86）、YAM（CCPM-86）、IMP（CP/M）、KMD（CP/M）、rz/sz（Unix、Xenix、VMS、Berkeley Unix、Venix、Xenix、Coherent、IDRIS、Regulus）支持。商业实现包括镜像和专业YAM。[1]支持这些扩展的通信程序自1981年起就开始使用。

下面描述的1k块长度（XMODEM-1k）可以与YMODEM批处理协议一起使用，或者与XMODEM/CRC协议相同的单文件传输一起使用，除了支持1k块的最小更改之外。

另一个扩展是YMODEM-g协议。当与端到端纠错介质（如X.PC和纠错调制解调器）一起使用时，YMODEM-g提供最大吞吐量的批量传输，包括TeleBit、美国机器人公司、Hayes、电子保险库、数据竞赛和其他公司提供的9600 bps单元。

为了完成这本书，编辑版本的沃德克里斯滕森的原始协议文件和约翰伯恩斯的CRC-16文件包括作为参考。

对MODEM或MODEM7协议的引用已更改为XMODEM以适应当地语言。在澳大利亚，它被恰当地称为Christensen协议。

## 3.1开创者说

#：130940 S0/Communications 25-Apr-85 18:38:47

Sb: my protocol

Fm: Ward Christensen 76703,302

To: all

请注意，文章确实正确地引用了我的一些短语，如“not robust”等。

其实这是一个快速hack出来的，非常没有计划（像我做的一切），只是满足个人与“其他人”通讯的需要的协议。

只是在“事实”上，它完成于1977年8月，此时我把它发布在公共领域，使它成为标准。

我想是时候把它记录下来了（人们打电话给我，说“我的产品想要包含它——有什么可参考吗”，或者“我正在写一篇关于它的论文，我在参考文献中能放了什么”），并且建议对其进行“增量扩展”，它可能“完全”采用Chuck Forsberg的YAM协议的形式。他用C为CP/M编写了YAM，并将其放在公共域中，为Unix编写了一个批处理协议，叫做rb（批量接收）和sb（批量发送），其基本上是在XMODEM的基础上，带有（1）0包，包含文件名日期时间和大小；（2）可选的1K块大小（3）CRC-16。

他做了一些聪明的编程来检测错误的ACK或EOT，但大体上还是保持不变。

那些建议我对协议进行重大修改的人，比如“全双工”、“多个未完成的块”、“多个目的地”等等，都不明白协议令人难以置信的简单性是它在尽可能多的机器和程序中幸存至今的原因之一！

想想77年左右的PC-NET团队——他们有一个协议，但“极其复杂”，因为它试图做到“所有人都能拥有一切”。比如，在7位系统上发送二进制文件，等等。我不是那么“仁慈”。我有一个8位UART，所以“我的协议是一个8位协议”，我只想对那些被7位限制阻碍的人说“对不起”。

**块大小：**Chuck Forsberg为我的协议创建了一个扩展，称为YAM，它也通过他的UNIX公共域程序rb和sb得到支持。他们巧妙地发送了一个“Block 0”，其中包含文件名、日期、时间和大小。不幸的是，它的UNIX风格有点奇怪——八进制数等等。但是，这是一个很好的方法来克服MODEM7引入的笨拙的“echo the chars of the name”。此外，chuck使用CRC-16和可选的1K块。因此，Block 0、1K和CRC，使它成为一个“非常灵活的新协议”，与我自己的协议没有太大的不同。

另外，还有一个吸引人的名字——YMODEM，这对一些人来说意味着它是“XMODEM之后的下一个东西”，而对另一些人来说则意味着它是Y（am）调制解调器协议 。我不想强调太多-出于担心其他制造商可能会认为这是一个“竞争性”协议，而不是一个“独立的”协议。查克目前正在出售一个增强版的CP/M-80 C程序YAM，称之为专业版Yam，它是为PC而写的——我现在正在使用它。非常丝滑！32K捕获缓冲区、脚本、滚动、以前捕获的文本搜索，以及用于几乎所有内容的内置命令——目录（按各个方向排序）、XMODEM、YMODEM、KERMIT和ASCII文件上载/下载，等等。你可以对它进行编程，使其在大多数任何系统中都能“正常工作”。

## 4.XMODEM协议增强

本章讨论对Ward Christensen 1982年XMODEM协议描述文档的协议扩展。

原始文档建议用户在重试10次后被询问是继续尝试还是中止。大多数程序不再询问操作员是否希望继续重试。几乎所有可纠正的错误都会在前几次重新传输中得到纠正。如果线路太差，10次尝试都不够，存在一个未被发现的错误的重大危险。如果连接是那么坏，最好重拨一个更好的连接，或邮寄软盘。

## 4.1快速中止

YAM和专业版-YAM X/YMODEM例程可以识别两个连续的CAN（0x18）字符序列作为传输中止命令，而不会出现调制解调器错误（溢出、帧错误等）。当等待块的开始或等待已发送块的确认时，会识别此序列。 对两个连续CAN字符的检查可减少行命中中止的传输数。YAM在中止XMODEM、YMODEM或ZMODEM协议文件预传输时发送八个CAN字符。Pro-YAM接着发送八个backspaces，用于从远程键盘输入缓冲区中删除CAN字符，以防远程主机已经准备中止传输，并且正在等待键盘命令。

## 4.2 CRC-16选项

XMODEM协议使用可选的两字符CRC-16，而不是原始协议和大多数商业实现所使用的一字节算术校验和。CRC-16保证检测所有单位和双位错误、所有具有奇数个错误位的错误、所有长度小于等于16的突发错误，所有17位错误突发的99.9969%，以及所有可能的较长错误突发的99.9984%。相比之下，双位错误或9位或更多的突发错误可以偷偷通过XMODEM协议的算术校验和。

XMODEM/CRC协议与XMODEM协议类似，只是在请求第一个块时，接收方通过发送C（0x43）而不是NAK来指定CRC-16。发送两字节的CRC来代替一字节的算术校验和。

YAM发送c使能r命令在单文件接收中启用CRC-16，相当于MODEM7系列程序中的原始实现。这仍然是默认值，因为许多商业通信程序和公告板系统仍然不支持CRC-16，特别是那些用Basic或Pascal编写的程序。

如果发送方和接收方都报告传输成功，那么带有CRC的XMODEM协议是准确的。该协议在分时系统中存在由于缓冲区过载而丢失的字符时是健壮的。

接收程序生成的单字符ACK/NAK响应很好地适应分速调制解调器，它的反向信道限制在主信道速度的10%或更低。

XMODEM和YMODEM是半双工协议，它们不试图同时在两个方向上传输信息和控制信号，这避免了当用户试图使用全双工异步文件传输协议（如Blast）时报告的缓冲区溢出问题。

专业版YAM为XMODEM的错误检测和恢复添加了一些专有的逻辑增强功能，这些兼容的增强功能消除了其他程序在不太理想的条件下使用XMODEM协议时产生的大部分错误文件传输。

## 4.3 XMODEM-1k 1024字节块

使用YMODEM从Unix下载吞吐量令人失望，导致1982年开发了1024字节块。1024字节块将分时系统、调制解调器和分组交换网络的延迟对吞吐量的影响降低了87.5%，此外还减少了XMODEM 3%的开销（块号、CRC等）。

某些环境不能接受1024字节的突发数据，包括某些网络和小型计算机端口。所以较长的数据块长度应该是一个可选项。

使用1024字节块的选项在发送程序的命令行或选择菜单上表达出来（见下面的“KMD/IMP YMODEM例外”）。1024字节块提高了许多应用程序的吞吐量。

STX（02）代替传输块开始处的SOH（01），以通知接收方是较长的块长度。传输的块包含1024字节的数据。接收方应该能够接受任何128和1024字节块的混合。块编号（在块的第二和第三字节中）每个块加一，而不考虑块长度。

如果发送方没有收到当前块的有效ACK，则不能在128到1024字节的块长度之间更改。如果不遵守此限制，传输错误将无法检测到。

如果正在使用1024字节块，则文件可能会“增长”到1024字节的下一个倍数。如果分配粒度为1k或更大，这不会浪费磁盘空间。对于YMODEM批量传输，在文件名块中传输的可选文件长度允许接收方丢弃填充，保留确切的文件长度和内容。

1024字节块可用于批处理文件传输或单文件传输。CRC-16应与k选项一起使用，以保持电话线上的数据完整性。如果程序希望执行此建议，则应取消传输，如果接收方请求校验和而不是CRC-16，则发出一条信息性诊断消息。

在任何情况下，发送程序都不能使用CRC-16，除非接收方命令CRC-16。

例1.XMODEM-1k区块

SENDER RECEIVER

"sx -k foo.bar"

"foo.bar open x.x minutes"

C

STX 01 FE Data[1024] CRC CRC

ACK

STX 02 FD Data[1024] CRC CRC

ACK

STX 03 FC Data[1000] CPMEOF[24] CRC CRC

ACK

EOT

ACK

例2.1024和128字节混合块

SENDER RECEIVER

"sx -k foo.bar"

"foo.bar open x.x minutes"

C

STX 01 FE Data[1024] CRC CRC

ACK

STX 02 FD Data[1024] CRC CRC

ACK

SOH 03 FC Data[128] CRC CRC

ACK

SOH 04 FB Data[100] CPMEOF[28] CRC CRC

ACK

EOT

ACK

## 5.批量文件传输

YMODEM批处理协议是XMODEM/CRC协议的一个扩展，它允许用一个命令传输0个或多个文件（如果请求的文件都不可访问，则可以发送零个文件）。YMODEM批处理协议的设计方法是使用普通程序以分层方式发送和接收XMODEM块类似于分组交换方法。

当MODEM7中已经存在一个批处理协议时，为什么有必要设计一个新的批处理协议？MODEM7使用的批处理文件模式是不合适的，因为它不允许传输完整的路径名、文件长度、文件日期或其他属性信息。这样一个限制性的设计，只是在考虑CP/M的情况下仓促实施，不允许扩展到当前的个人计算领域，如Unix、DOS和面向对象系统。此外，MODEM7批处理文件模式有点容易受到传输损害。

与单个文件传输的情况一样，接收方通过发送“C”字符（用于CRC-16）来启动批处理文件传输。

发送方打开第一个文件并发送带有以下信息的块编号0。

批量传输只需要路径名（文件名）部分。

要保持向上兼容性，块0中所有未使用的字节必须设置为null。

路径名路径名（通常是文件名）作为以null结尾的ASCII字符串发送。这是面向句柄的MSDOS(TM)函数和C 库(fopen)函数使用的文件名格式。

汇编语言示例如下：DB 'foo.bar',0

路径名中不能包含空格。通常仅传输文件名干（无目录前缀），除非发件人已选择YAM的f选项发送完整路径名。不发送源驱动器（A: B:等等）。

文件名注意事项：

* 除非发送系统支持大写/小写文件名，否则文件名将强制使用小写。这对于以大写和小写形式存储文件名的系统（如Unix）的用户来说是一种方便。
* 接收者应该容纳小写和大写的文件名。
* 在不同操作系统之间传输文件时，文件名必须为发送方和接收方操作系统所接受。

如果包含目录，则用反斜号分割(/)， 例如，“subdir/foo”是可接受的，“subdir\foo”不是。

长度——文件长度和每个后续字段都是可选的（不能跳过字段）。长度字段以十进制字符串形式存储在块中，用于计算文件中的数据字节数。文件长度不包括任何CPMEOF（^Z）或用于填充最后一个块的其他垃圾字符。

如果正在传输的文件在传输过程中不断增长，则长度字段应至少设置为最终预期的文件长度，否则不发送。

接收方存储指定数量的字符，丢弃发送方为填充最后一个块而添加的任何填充。

修改日期mod Date是可选的，可以发送文件名和长度，而不需要发送mod Date。

如果发送了修改日期，则用一个空格将修改日期与文件长度分隔开。

mod date以八进制数字形式发送，表示文件内容上次更改的时间，从1970年1月1日世界协调时间（GMT）开始以秒为单位。日期为0表示修改日期未知，应保留为收到文件的日期。

选择这种标准格式是为了消除因不同时区之间的传输而产生的歧义。

模式——如果文件模式被发送，一个空格将文件模式与修改日期分开。文件模式存储为八进制字符串。除非文件来自Unix系统，文件模式设置为0。rb检查文件模式中的0x8000位，该位表示Unix类型的常规文件。假定设置了0x8000位的文件是从另一个使用相同文件惯例的Unix（或类似）系统发送的。这些文件不会以任何方式进行翻译。

序列号——如果发送序列号，则序列号与文件模式之间用一个空格隔开。发送程序的序列号存储为八进制字符串。没有序列号的程序应省略此字段，或将其设置为0。接收方使用此字段是可选的。

其他字段YMODEM被设计成允许像上面那样添加额外的标题字段，而不会与旧的YMODEM程序产生兼容性问题。如果特殊应用要求需要其他字段，请联系Omen的技术人员。

块的其余部分被设置为空值。这对于保持向上兼容性至关重要。如果block 0可能超过128字节（可能使用Unix 4.2 BSD扩展文件名），则应如上所述将该块作为1K块发送。

如果接收到文件名块时出现CRC或其他错误，则会请求重新传输。在接收到文件名块后，如果写打开成功，则会确认该文件。如果无法打开文件进行写操作，则接收方会使用上述CAN字符取消传输。

然后，接收方根据标准XMODEM/CRC协议，用“C”字符发起文件内容的传输。

在传输并确认文件内容和XMODEM EOT之后，接收方再次请求下一个路径名。

传输空路径名将终止批处理文件传输。

请注意，不传输任何文件不一定是错误的。如果无法打开发件人请求的任何文件进行读取，则可能是错误的。

大多数情况下，YMODEM默认接收方请求CRC-16。

源代码文件RZSZ.ZOO中包含的Unix程序sz和rz有关于YMODEM批处理协议的其他问题的回答。

例3批量传输会话（1个文件）：

SENDER RECEIVER

"sb foo.\*<CR>"

"sending in batch mode etc."

C (command:rb)

SOH 00 FF foo.c NUL[123] CRC CRC

ACK

C

SOH 01 FE Data[128] CRC CRC

ACK

SOH 02 FC Data[128] CRC CRC

ACK

SOH 03 FB Data[100] CPMEOF[28] CRC CRC

ACK

EOT

NAK

EOT

ACK

C

SOH 00 FF NUL[128] CRC CRC

ACK

图7 YMODEM标题信息和功能

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| Program | Length | Date | Mode | S/N | 1k-Blk | YMODEM-g |

|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|Unix rz/sz | yes | yes | yes | no | yes | sb only |

|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|VMS rb/sb | yes | no | no | no | yes | no |

|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|Pro-YAM | yes | yes | no | yes | yes | yes |

|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|CP/M YAM | no | no | no | no | yes | no |

|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

|KMD/IMP | ? | no | no | no | yes | no |

|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|

## 5.1 KD/IMP YMODEM例外

KMD和IMP使用接收方发出的“CK”字符序列触发1024字节块的使用，作为向发送程序指定此选项的替代方法。

图4.批量传输会话（2个文件）：

SENDER RECEIVER

"sb foo.c baz.c<CR>"

"sending in batch mode etc."

C (command:rb)

SOH 00 FF foo.c NUL[123] CRC CRC

ACK

C

SOH 01 FE Data[128] CRC CRC

ACK

SOH 02 FC Data[128] CRC CRC

ACK

SOH 03 FB Data[100] CPMEOF[28] CRC CRC

ACK

EOT

NAK

EOT

ACK

C

SOH 00 FF baz.c NUL[123] CRC CRC

ACK

C

SOH 01 FB Data[100] CPMEOF[28] CRC CRC

ACK

EOT

NAK

EOT

ACK

C

SOH 00 FF NUL[128] CRC CRC

ACK

图5.批量传输会话-1k块：

SENDER RECEIVER

"sb -k foo.\*<CR>"

"sending in batch mode etc."

C (command:rb)

SOH 00 FF foo.c NUL[123] CRC CRC

ACK

C

STX 01 FD Data[1024] CRC CRC

ACK

SOH 02 FC Data[128] CRC CRC

ACK

SOH 03 FB Data[100] CPMEOF[28] CRC CRC

ACK

EOT

NAK

EOT

ACK

C

SOH 00 FF NUL[128] CRC CRC

ACK

图6. sz传输的YMODEM文件名块：

-rw-r--r-- 6347 Jun 17 1984 20:34 bbcsched.txt

00 0100FF62 62637363 6865642E 74787400 |...bbcsched.txt.|

10 36333437 20333331 34373432 35313320 |6347 3314742513 |

20 31303036 34340000 00000000 00000000 |100644..........|

30 00000000 00000000 00000000 00000000

40 00000000 00000000 00000000 00000000

50 00000000 00000000 00000000 00000000

60 00000000 00000000 00000000 00000000

70 00000000 00000000 00000000 00000000

80 000000CA 56

此双字符序列通常在直接通信的单进程微型计算机上工作良好，如果程序严格遵守本文中包含的所有XMODEM建议，使用差不多实现的XMODEM程序可能运行得不太好。分时系统和分组交换网络可以分离连续字符，使得这种方法不可靠。

如果操作环境不妨碍可靠的实现，发送程序可以检测CK序列。

KMD和IMP传输头块最后两个字节中的CP/M记录计数，而不是十进制的标准YMODEM文件长度。

## 6.YMODEM-g文件传输

发展中的技术正在使用非常专业的技术以更高的速度提供电话线数据传输。这些高速调制解调器以及会话协议（如X.PC）以大大增加的延迟时间为代价提供高速、几乎无差错的通信。

这种延迟时间与人与人之间的交互相比是适度的，但是它削弱了大多数纠错协议的吞吐量。

事实证明，在这种情况下，对YMODEM的g选项是有效的。g选项由接收方驱动，接收方通过发送g而不是C来启动批处理传输。当发送方识别到g时，它绕过通常等待每个发送的块的ACK，全速发送后续块，受XOFF/XON或介质施加的其他流控制。

发送方期望inital G启动特定文件的传输，还期望在每个文件的末尾发送EOT上的ACK。此同步允许接收方根据需要打开和关闭文件。

如果在YMODEM-g传输中检测到错误，接收方将使用多个CAN-abort序列中止传输。ZMODEM协议应在需要流吞吐量和错误恢复的应用程序中使用。

图8.YMODEM-g传输会话：

SENDER RECEIVER

"sb foo.\*<CR>"

"sending in batch mode etc..."

G (command:rb -g)

SOH 00 FF foo.c NUL[123] CRC CRC

G

SOH 01 FE Data[128] CRC CRC

STX 02 FD Data[1024] CRC CRC

SOH 03 FC Data[128] CRC CRC

SOH 04 FB Data[100] CPMEOF[28] CRC CRC

EOT

ACK

G

SOH 00 FF NUL[128] CRC CRC

## 7.XMODEM协议概述

1982年8月9日，沃德·克里斯滕森。

我将保留此文件的主副本。请通过CBBS/芝加哥电话（312）545-8086、CBBS/CPMUG（312）849-1132或语音电话（312）849-6279传达更改或建议。

## 7.1定义

<soh> 01H

<eot> 04H

<ack> 06H

<nak> 15H

<can> 18H

<C> 43H

## 7.2传输中级协议

异步，8个数据位，无奇偶校验，一个停止位。

该协议对传输的数据内容没有任何限制。在128字节的数据消息中找不到控制字符。绝对可以发送任何类型的数据-二进制、ASCII等。该协议还没有正式应用于仅传输ASCII（或解包十六进制）数据的7位环境，虽然它可以简单地通过两端都同意和协议依赖的数据与7F十六进制之前，验证它。我特别是指校验和，块数和他们的1-补充。

希望保持CP/M文件结构兼容性，即允许在CP/M系统之间或从CP/M系统中调制ASCII 文件的用户应遵循以下数据格式：

* 使用ASCII制表符（09H）；每8个标签设置一次。
* CR/LF端接线路（0DH 0AH）
* 文件结尾由^Z，1AH表示。（一个或多个）
* 数据是可变长度的，即应被视为一个连续的数据字节流，被分成128字节块，纯粹用于传输。
* CP/M“特殊性”：如果数据恰好以128字节边界结束，即CR在127，LF在128，则包含^Z EOF字符的后续扇区是可选的，但是首选的。一些实用程序或用户程序在没有^Zs的情况下仍然不能处理EOF。
* 最后发送的块与其他块没有区别，即没有“短块”。

图9.XMODEM消息块级协议，传输的每个块看起来像：

<SOH><blk #><255-blk #><--128 data bytes--><cksum>

其中:

<SOH> = 01 Hex

<blk #> = 二进制数，从01开始递增1

<255-blk #> = blk的反码，作为补充验证

<cksum> = 仅数据字节之和。丢弃所有进位

## 7.3文件级协议

### 7.3.1通用发送方和接收方

所有错误都将重试10次。对于带有操作员（即不使用XMODEM）的版本，在10次错误后将询问操作员是“重试还是退出”。

某些版本的协议使用<can>、ASCII ^X来取消传输。这从未被作为一个标准采用，因为只有一个“abort”字符使得传输容易由于<ack><nak>或<soh>跳变成<can>并中止传输而被错误终止。

该协议可以被认为是“接收方驱动的”，也就是说，发送方不需要自动地重新发送，尽管在当前的实现中它是这样做的。

### 7.3.2接收程序注意事项

接收方有一个10秒的超时。每次超时时，它都会发送一个<nak>。接收方的第一个超时，即发送一个<nak>，向发射器发出启动信号。如果发送方准备就绪，接收方也可以立即发送一个<nak>。这将节省最初的10秒超时。但是，接收方必须每10秒继续超时一次，以防发送方未准备好。

一旦进入一个接收块，接收方进入一秒钟的每个字符和校验和超时。如果接收方想要因为一个块的任何原因（无效的头，接收数据超时）回复<nak>时，它必须等待接收行清除。请参阅“编程提示”的想法

同步——收到一个有效的块号时，它将是：1、预期的块号，在这种情况下一切正常；2、重复先前接收到的块。这应该被认为是正常的，并且只表明接收方<ack>出现故障，发送方重新发送；3、任何其他块号都表示严重的同步丢失，例如发送方出现类似于<ack>的线路故障的罕见情况。中止传输，发送<can>。

### 7.3.3发送程序注意事项

在等待传输开始时，发送方只有一个很长的超时时间，比如一分钟。在当前协议中，发送方在重试前有10秒的超时时间。我建议不要这样做，让协议完全由接收方驱动。这将与现有程序兼容。

当发送方没有更多的数据时，它发送一个<eot>，并等待一个<ack>，如果它没有得到一个<ack>，则重新发送该<eot>。同样，该协议可以由接收方驱动，发送方只有高级别的1分钟超时时间去中止。

这是数据流的一个示例，发送一个3块消息。它包括两个最常见的行命中-一个垃圾块和一个被垃圾的应答。<xx>表示校验和字节。

图10.包括错误恢复的数据流：

SENDER RECEIVER

times out after 10 seconds,

<--- <nak>

<soh> 01 FE -data- <xx> --->

<--- <ack>

<soh> 02 FD -data- xx ---> (data gets line hit)

<--- <nak>

<soh> 02 FD -data- xx --->

<--- <ack>

<soh> 03 FC -data- xx --->

(ack gets garbaged) <--- <ack>

<soh> 03 FC -data- xx ---> <ack>

<eot> --->

<--- <anything except ack>

<eot> --->

<--- <ack>

(finished)

## 7.4编程提示

应使用指定等待秒数的参数调用字符接收子例程。接收方应首先以10秒的时间调用它，回复<nak>后再试10次。

在接收到<soh>之后，接收方应该调用字符接收子例程，对消息的剩余部分和<cksum>进行1秒超时。由于它们是作为连续流发送的，因此超时意味着一个严重的故障，例如，导致看到127个字符而不是128个字符。

当接收者希望<nak>时，它应该调用一个“清除”子程序，等待行被清除。调用发送者在完成一个块的发送后立即在其UART缓冲区中抛弃任何字符，以确保没有误读的小差错。

最常见的技术是“PURGE”调用字符接收子程序，指定1秒超时（应调整这些时间，以便与分时系统一起使用），并循环回PURGE，直到出现超时。然后发送<nak>，确保另一端看到它。

您可能希望将John Mahr推荐的代码添加到您的字符接收例程中，以便在UART显示帧错误时设置错误标志，或溢出。这将有助于捕捉更多的小故障——其中最常见的是两个连续字节中字节高位的命中。<cksum>则显示正常，因为在1字节中计数产生的结果与在加00H+00H时加80H+80H的结果相同。

## 8.XMODEM/CRC概述

Original 1/13/85 by John Byrns -- CRC option.

调制解调器协议中使用的CRC是块检查的另一种形式，它提供比原始校验和更健壮的错误检测。安德鲁S。Tanenbaum在他的书《计算机网络》中说，调制解调器协议使用的CRC-CCITT将检测所有的单位和双位错误、所有的奇数位错误、所有长度为16或更少的突发错误、99.997%的17位错误突发以及99.998%的18位和更长的突发错误。（此可靠性数字具有误导性，因为XMODEM的关键监控功能不受此CRC保护）

用CRC替换校验和的调制解调器协议的改变是直接的。如果我们只做了这些，我们就不能在使用旧校验和协议的程序和使用新CRC协议的程序之间进行通信。添加了初始握手来解决此问题。握手允许具有CRC功能的接收程序确定发送程序是否支持CRC选项，如果支持，则将其切换到CRC模式。这种握手的设计使它能够在只实现原始协议的程序中正常工作。关于这种握手的描述见第10节。

图11消息块级协议，CRC模式，CRC模式下传输的每个数据块如下所示：

<SOH><blk #><255-blk #><--128 data bytes--><CRC hi><CRC lo>

其中:

<SOH> = 01 hex

<blk #> = binary number, starts at 01 increments by 1, and

wraps 0FFH to 00H (not to 01)

<255-blk #> = ones complement of blk #.

<CRC hi> = byte containing the 8 hi order coefficients of the CRC.

<CRC lo> = byte containing the 8 lo order coefficients of the CRC.

## 8.1 CRC计算

### 8.1.1 形式定义

计算16位CRC的形式定义消息位被认为是多项式的系数。此消息多项式首先乘以X^16，然后使用模2算法除以生成器多项式（X^16+X^12+X^5+1）。除法后的剩余部分是所需的CRC。由于调制解调器协议中的消息块是128字节或1024位，因此消息多项式的阶数为X^1023。消息块第一个字节的高阶位是消息多项式中X^1023的系数。消息块最后一个字节的低阶位是消息多项式中X^0的系数。

图12用C编写的CRC计算示例

以下XMODEM crc例程取自“rbsb.c”。有关用法，请参阅这些程序的源代码（包含在RZSZ.ZOO中）。此文件中还包含一个快速表驱动版本。

/\* update CRC \*/

unsigned short

updcrc(c, crc)

register c;

register unsigned crc;

{

register count;

for (count=8; --count>=0;) {

if (crc & 0x8000) {

crc <<= 1;

crc += (((c<<=1) & 0400) != 0);

crc ^= 0x1021;

}

else {

crc <<= 1;

crc += (((c<<=1) & 0400) != 0);

}

}

return crc;

}

## 8.2 CRC文件级协议更改

### 8.2.1发送方和接收方的通用协议

CRC选项文件级协议的唯一变化是初始握手，用于确定发送和接收程序是否都支持CRC模式。所有调制解调器程序都应支持校验和模式，以便与旧版本兼容。希望以CRC模式接收的接收程序通过发送<C>来代替初始<nak>来实现模式设置握手。如果发送程序支持CRC模式，它将识别<C>，并将自己设置为CRC模式，并通过发送第一个块来作出响应，如同已接收到<nak>。如果发送程序不支持CRC模式，它将根本不响应<C>。在接收方发送<C>后，它将等待最多3秒钟，等待启动第一个块的<soh>。如果它在3秒内收到<soh>，它将假定发送方支持CRC模式，并将在CRC模式下继续进行文件交换。如果在3秒内未收到<soh>，接收器将切换到校验和模式，发送<nak>，并在校验和模式下继续。如果接收方希望使用校验和模式，它应该发送一个初始的<nak>，并且发送程序应该响应原始调制解调器协议中定义的<nak>。在通过初始<C>或<nak>设置模式后，协议遵循原始调制解调器协议，并且无论使用校验和还是CRC都是相同的。

### 8.2.2接收程序注意事项

模式设置握手至少有4种情况会出错。

1.首字母<C>可能会被混淆或丢失。

2.首字母<soh>可能会被混淆。

3.初始<C>可以改变为<nak>。

4.想要在checksum中接收的接收方的初始<nak>可以改变为<C>。

如果接收方在第一次超时后发送第二个<C>，第一个问题就可以解决。这个过程可以重复几次。在发送<nak>并切换到校验和模式之前，不能重复太多次，否则没有CRC支持的发送程序可能会超时并中止。重复<C>也将解决第二个问题，如果发送程序通过响应来协作，就像接收到<nak>而不是忽略额外的<C>。

可以解决问题3和4，但可能不值得麻烦，因为它们很少发生。它们可以通过在大量连续<nak>之后在发送或接收程序中切换模式来固定。然而，这种解决方案可能会带来其他问题。

### 8.2.3发送程序注意事项

发送程序应以校验和模式启动。这将确保与只接收校验和的程序兼容。每当在第一个<nak>或<ack>之前接收到<C>时，发送程序应将自身设置为CRC模式，并像接收到<nak>一样响应。在收到第一个<ack>之前，发送方应该像响应<nak>一样响应额外的<C>。这将有助于接收程序在<soh>丢失或乱码时确定正确的模式。收到第一个<ack>后，发送程序应忽略<C>。

## 8.3带CRC选项的数据流示例

这里是用于接收方请求以CRC模式传输但发送方不支持CRC选项的情况的数据流示例。此示例还包括各种传输错误。<xx>表示校验和字节。

图13.数据流：接收方有CRC选项，发送方没有

SENDER RECEIVER

<--- <C>

times out after 3 seconds,

<--- <C>

times out after 3 seconds,

<--- <C>

times out after 3 seconds,

<--- <C>

times out after 3 seconds,

<--- <nak>

<soh> 01 FE -data- <xx> --->

<--- <ack>

<soh> 02 FD -data- <xx> ---> (data gets line hit)

<--- <nak>

<soh> 02 FD -data- <xx> --->

<--- <ack>

<soh> 03 FC -data- <xx> --->

(ack gets garbaged) <--- <ack>

times out after 10 seconds,

<--- <nak>

<soh> 03 FC -data- <xx> --->

<--- <ack>

<eot> --->

<--- <ack>

以下是一个数据流示例，用于接收端以CRC模式请求传输，而发送端支持CRC选项的情况。此示例还包括各种传输错误。<xxxx>表示2个CRC字节。

图14.接收方和发送方都有CRC选项

SENDER RECEIVER

<--- <C>

<soh> 01 FE -data- <xxxx> --->

<--- <ack>

<soh> 02 FD -data- <xxxx> ---> (data gets line hit)

<--- <nak>

<soh> 02 FD -data- <xxxx> --->

<--- <ack>

<soh> 03 FC -data- <xxxx> --->

(ack gets garbaged) <--- <ack>

times out after 10 seconds,

<--- <nak>

<soh> 03 FC -data- <xxxx> --->

<--- <ack>

<eot> --->

<--- <ack>

## 9.更多信息

请联系Omen科技公司获取本文件的troff源文件和排版副本。

## 9.1 TeleGodzilla公告栏

有趣的文件包括RZSZ.ZOO（C源代码）、YZMODEM.ZOO（官方的XMODEM、YMODEM和ZMODEM协议描述）、ZCOMMEXE.ARC、ZCOMMDOC.ARC和ZCOMMHLP.ARC（PC-DOS共享程序，包括XMODEM、True YMODEM（TM）、ZMODEM、Kermit Sliding Windows、Telink、MODEM7 Batch、script language等）。

## 9.2 Unix UUCP访问

UUCP站点可以获取此文件的当前版本：uucp omen!/u/caf/public/ymodem.doc /tmp，可用文件的不断更新列表存储在/usr/spool/uucppublic/files中。使用uucp检索这些文件时，请记住，系统上的目标目录必须可由任何人写入，否则uucp传输将失败。

下面的L.sys行调用TeleGodzilla（主机操作中的Pro-YAM）。TeleGodzilla自动确定输入速度。

为了响应“Name Please:”uucico将Pro YAM“link”命令作为用户名。密码（Giznoid）控制对连接到IBM PC的其他串行端口的Xenix系统的访问。Pro YAM和Xenix之间的通信使用9600 bps；YAM将其转换为呼叫者的速度。

最后，调用uucico作为uucp登录。

预兆任何ACU 2400 1-503-621-3746 se:--se:链接ord:Giznoid in:--in:uucp

## 10.修订

6-18-88为清晰起见，进一步修订。在两个示例中更正了块编号。10-27-87为剩余要发送的文件数和剩余要发送的总字节数添加了可选字段。10-18-87流量控制讨论添加到1024字节的块描述中，为清晰起见，对每个用户的评论进行了小的修改。

8-03-87为清晰起见进行了修订。1987年5月31日强调了YMODEM的最低要求，并更新了有关访问文件的信息。1986年11月9日澄清了术语和一些小要点。1986年4月15日的版本澄清了一些关于CRC计算和页眉空格的问题。

## 11.1项目

ZCOMM是专业YAM的一个共享软件小兄弟，在TeleGodzilla和其他公告板系统上以ZCOMMEXE.ARC的形式提供。ZCOMM可以用来测试YMODEM和ZMODEM的实现。

支持YMODEM的Unix程序在RZSZ.ZOO的TeleGodzilla上可用。这个ZOO归档文件包含一个ZCOMM/Pro YAM/PowerCom脚本ZUPL.T，用于上载引导程序MINIRB.C，编译它，然后使用编译后的MINIRB上载其余文件。支持大多数类Unix系统，包括V7、Xenix、Sys III、4.2 BSD、Sys V、Idris、Coherent和Regulus。

VAX-VMS的版本在VRBSB.SHQ中提供。

Irv Hoff在KMD和IMP系列程序中增加了1k块和基本YMODEM批传输，分别取代了XMODEM和MODEM7/MDM7xx系列。覆盖层可用于各种CP/M系统。

有关专业YAM通信软件的问题，请联系：Chuck Forsberg Omen Technology Inc 17505-V Sauvie Island Road Portland Oregon 97231语音：503-621-3406语音调制解调器：503-621-3746速度：19200（Telebit PEP），24001200300 Usenet:。。。！泰克！里德！预兆！caf计算机服务：700072304精灵：caf

与ZMODEM和Kermit不同，XMODEM和YMODEM在可靠的高性能实现的道路上设置了障碍，表现为在行业领导者的XMODEM和YMODEM项目的压力下可靠性较差。Omen Technology为希望实现XMODEM、YMODEM和YMODEM的人提供咨询和其他服务，ZMODEM具有最先进的功能和可靠性。