EC #: 111232

PHYSIO CONTROL REDMOND, WASHINGTON	**Engineering Change**		EC # 111232	Page 1 of 2
	Change Class: 4	Effectivity Code: 12	Priority: Normal	
			Incorp. Method: Regular	

Doc Type	Doc ID	Aff Rev	Targ Rev	Reason Code: New Document
PS-Product Spec	3011345	1	1	Life Cycle Status: Proto

Title: Specification, ZMODEM Protocol

Product Applicability: All,

NPD Project: (optional) N/A

Development Plan Required: Yes □ No ☒

Training Requirements: None

Change Description: New Document

Change Effectivity: Upon Release

Attachments List

Att Type	Att ID	Att Type	Att ID

Approvals

Function	Printed Name	Signature	Date
Originator	Ward Silver		11 Nov 97
SWQA	Eric Schnellman	Eric Schnellman	11/11/97
Released by	Jim Clifford		11-11-97

FrameMaker

EC #: 111232

PHYSIO CONTROL
REDMOND, WASHINGTON

Engineering Change Justification/Validation Notes

EC#: 111232

Page 2 of 2

Originator: Ward Silver

1. **Reason for/Justification of:**

New document.

2. **510(k) Evaluation**

	YES	NO
Does the 2000-316 procedure require a 510(k) evaluation for this change?		X

If yes, answer the following questions by checking the appropriate box.
Note: *If a LCSO is required, verify that the 510(k) impact question has been answered.*

Is a **final assembly or accessory** part number or dash number added or changed?		X
Does this change affect **product specifications** (PS doc type)?		X

If the answer to either of the two questions above is yes, then complete a 510(k) evaluation and document the results per the 510(k) evaluation Procedure 3006878. If all questions are answered "No" then no further documentation of the decision is required.

3. ***Validation/Qualification***: (this section not required for prototypes)

Check all that apply:

□ Design Validation □ Process Validation □ Component Qualification □ Other
□ Design Verification □ Test Validation □ Equipment Qualification □ ______________

Note: If a Report was used for any type checked above then enter its' number(s) below in V/V/Q Details and in Attachments.

List Specification Sources (e.g. PRS,HRS,SRS, Engineering Drawings, MTS, etc.):

List Requirements affected by this change (e.g. vibration, charge time, timing, leakage, EMI, RF susceptibility, etc.):

V/V/Q Details (Describe approach(s) and show that all requirements listed have been met):

Location of supporting test data:

Rev: 1

Doc ID: 3011345

Doc Type: PS

Title: Zmodem Interface Control Document

Rev	Creation Date	Description	Reference EC #	Drawn By	Reviewer #1	Reviewer #2
1	11/10/97	First Release	111232	ESS		

PHYSIO CONTROL

11811 Willows Road Northeast
PO Box 97006
Redmond, WA 98073-9706 USA

Elements:	Qty

Appendices:	Qty	Pages	Page Sequencing

Released By:	Release Date: 11-11-97	Life Cycle Status: Prototype

Title:

Zmodem Interface Control Document

Doc Type: PS	Doc ID: 3011345	Rev: 1	**Page 1 of 59**

FrameMaker

1 Introduction

1.1 Purpose

The purpose of this Interface Control Document is to define the Physio interpretation of the ZMODEM protocol used to exchange data between Physio-Control products.

1.2 Scope

This document defines the standard ZMODEM interface between Physio-Control products, it does not support any third party interfaces. This document is a complete description of all interaction between the two elements, sender and receiver, to allow the transfer of patient and related data.

The information contained in this specification is for Physio-Control internal use only.

1.3 Applicable Documents

1.3.1 Physio Documents

3004961, Communications Interface Control Document, Revision 7, 1997.

1.3.2 Other Documents

ZMODEM Inter Application File Transfer Protocol by Chuck Forsberg, Omen Technology Inc., Portland Oregon, 1990.

The Working Programmer's Guide to Serial Protocols, Tim Kientzle, Coriolis Group Books, 1995.

TIA/EIA-602-1992, Data Transmission Systems and Equipment - Serial Asynchronous Automatic Dialing and Control, Telecommunications Industry Association, Washington, D.C., 1992

EIA/TIA-232-E-1991, Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange, Electronic Industries Association, Washington D.C., July, 1991.

1.4 Acronyms and Abbreviations

These definitions are only applicable to this document.

Addams — Code name for Defib product in development, the LP12.

BPS, bps — Bits Per Second. A unit of measure for identifying the digital rate of data transmission.

Cart — A defibrillation or other device that sends data to a "host". This term comes from the SCP specification.

DME — Data Management Entity: Any device used to store and/or manage data from Physio-Control products. These are usually DT-3/DT-4 type devices.

Frame — A ZMODEM frame consists of a header and 0 or more data subpackets.

Grapevine — AKA receiver [host device]

Header — All ZMODEM headers contain a type byte, format byte, **ZPAD**s, **ZDLE**, header information, and CRC.

Host A data management unit or other device receiving data from a “cart”, AKA Grapevine. This term comes from the SCP-ECG specification.

MODEM ‘Modulator/Demodulator’. In this document's usage, it refers to an interface between an RS-232 serial data stream with an analog telephone system.

RLE Run Length Encoding

Subpacket A ZMODEM subpacket is a stream of 0 to 1024 data bytes terminated by a flag sequence and CRC.

XMODEM Refers to the original 1977 file transfer etiquette introduced by Ward Christensen's MODEM2 program. It's also called the MODEM or MODEM2 protocol. Some who are unaware of MODEM7's unusual batch file mode call it MODEM7. Other aliases include "CP/M Users's Group" and “TERM II FTP 3". This protocol is supported by most communications programs because it is easy to implement.

ZMODEM A serial communications file transfer protocol.

1.5 Definitions

ASCII Mnemonic	Full ASCII name	ASCII Character	ASCII keystroke (Control code)
NUL	Null	00h	<ctrl-@>
SOH	Start of Heading	01h	<ctrl-A>
STX	Start of Text	02h	<ctrl-B>
ETX	End of Text	03h	<ctrl-C>
EOT	End of Transmission	04h	<ctrl-D>
ENQ	Enquire	05h	<ctrl-E>
ACK	Acknowledge	06h	<ctrl-F>
BEL	Bell	07h	<ctrl-G>
TAB	Tab	09h	<ctrl-I>
LF	Line Feed	0ah	<ctrl-J>
CR	Carriage Return	0dh	<ctrl-M>
DC1	XON	11h	<ctrl-Q>
NAK	Negative Acknowledge	15h	<ctrl-U>
CAN	Cancel	18h	<ctrl-X>
ESC	Escape	1Bh	ESC key

2 Overview

Standardized communication between Physio-Control devices can be described as several distinct communications modes. Therapy devices will store data for on-line or later transfer to Data Management Entities (DME). When transmission of the stored data is required, a connection between the device and DME will be established and data transfer completed. The methods of the connection process using ZMODEM as a core protocol are defined within this document.

This document is limited to a presentation of the Physio implementation of the ZMODEM protocol and is not intended to be a descriptive tutorial. It is assumed that the reader is familiar with the concepts presented in chapters 1 to 5 of Kientzle (see section 1.3.2, Other Documents). In addition, it is highly recommended that the reader also review chapters 8 and 18 of Kientzle which specifically discuss ZMODEM.

2.1 Background

Physio uses ZMODEM as a core protocol for data communications functions. While ZMODEM is widely used, it is not strictly-controlled, with several "flavors" in use and a weak defining document from its inventors at Omen Technologies. Furthermore, there are no test packages available to determine the quality of any particular implementation

Given that information on ZMODEM is not from a single-source, our specification will not be able to conform to the usual model where all the information is contained in one (or very few) documents. Rather, we will use an "aggregate" model, where information is distributed among several authoritative sources, with supplemental information added to make a coherent whole.

Purpose of developing this aggregate specification:

1) To provide guidelines for future product development
2) Provide a "ruler" for measuring system performance during and after development
3) Capture lessons-learned and implementation knowledge from product to product

In order to accomplish this, there must be a common understanding of what ZMODEM is, what it does well and poorly, and how to test it.

2.2 Structure of this Document

The ZMODEM specification will consist of the following sections:

Introduction and Overview.
Global ZMODEM transmit and receive requirements.
Global ZTERM requirements.
Implementation specific requirements.
A standard set of tests using Procomm's ASPECT.
Examples and other appendices.

2.2.1 Global ZMODEM Requirements Section

This section describes the Physio implementation of ZMODEM which consists of:

1. Excerpts from the Omen document,
2. Requirements and their rationale,
3. in-depth discussion of important features and concepts, and
4. supplemental diagrams or examples to illustrate key points.

2.2.1.1 Who's on First

The multiple sources of information caused us to develop a system to keep it all straight:

1. Omen is the designer and provides the documentation basis because of the availability of the documentation in computer form.
2. Kientzle's implementation is the core of the Physio implementation.
3. The numbered requirements are what is expected of the implementer which may or may not match either of the above.

2.2.1.2 Commentary Formats

This document uses a number of specially formatted paragraphs to convey information. They are:

OMEN Omen text always appears as this paragraph does.

Commentary: Important commentary about a requirement or other aspect of this document uses paragraphs such as this.

Rationale: Rationale about a requirement uses paragraphs such as this.

2.2.1.3 Requirement Format

Requirements that must be met use the special format shown below:

2.2.1.3/1 *Example requirement.*
STATE MACHINE REQUIREMENT.

Where:

The first part of the requirement is the parent section number, in the case of the sample requirement, 2.2.1.3.

The second part is the number of the requirement within the section, 1 in the example above.

The body of the requirement.

A notation as to whether the requirement is part of the basic state machine and can be found on one of the state machine diagrams, or the requirement is more generic or used in too many state transitions to be factored into the diagrams without them becoming too complex to understand.

2.2.2 Global ZTERM Requirements Section

This describes specific ZTERM requirements not covered in the Communications ICD, 3004961.

2.2.3 Tests

The standard tests using Procomm's ASPECT to validate a ZMODEM implementation.

2.2.4 Implementation-Specific Requirements Section

An implementation section will consist of a three-part chapter for each Physio product that implements any subset of ZMODEM. The three parts are

(a) exceptions and extensions to the standard section information,
(b) design description information, and
(c) specific test implementations including modifications to the standard test scripts.

The exceptions/extensions section will be a list of deviations from this document and/or from standard/ reference information. Rationales for deviations must be presented.

Design descriptions may be simply copied from the product software design description, but expanded discussion of the data communications functions is desired.

References to the test scripts and protocols used to certify the satisfactory behavior of the particular ZMODEM implementation should be contained in the final part, along with any rationale for deviations from the standard tests.

The point of having a per-product section is to provide a long-term knowledge pool for ZMODEM implementation. This is important both for new products and for legacy product support.

2.2.5 Definitions

To use this specification for the ZMODEM protocol requires the understanding of the operation of several different concurrent processes. These are analogous to the levels of the OSI model (see Section 3, Interface -- OSI/ISO Model for Physio-Control). It is important to the accurate understanding of this document that the following terms are used consistently and appropriately:

Connection - This refers to the process by which a communication channel is established, maintained, and terminated. It is activated at the request of a user or by an automated device function. There are four states of this process:

Not Connected - The idle state of the connection process. No channel functions are available to any other process.

Attempting Connection- The connection process is requesting or creating resources to support communications functions. No functions (except status reporting) are available to any other process.

Connected - A channel is considered to be connected when all required communications functions are available for use by other processes.

Disconnecting - The connection process is terminating or releasing resources used to support communications functions for the channel. This state can be entered as a result of successful function completion or unilaterally as a result of the loss of resources to support a required function.

Connection Manager -This refers to the supervisory process for the connection. The connection manager interacts with the session manager process (see below) in order to determine resource needs and status information.

Session - In the context of this document, a session includes all ZMODEM activity following the establishment of the channel through the final **ZFIN/ZFIN/"OO"** sequence.

Session Manager -The supervisory process for the session. The session manager controls the operations of ZMODEM and interacts with the connection manager and shell processes.

Mode - In the context of this document, "mode" refers to the subset of ZMODEM protocol rules referred to as "ZTERM" (see section 5, Physio-Control ZTERM Mode). The entry into ZTERM mode does not terminate or preempt the ZMODEM session. Entry into and exit from ZTERM mode is controlled by the session manager.

Shell - In the context of this document, "shell" refers to the process for which information is exchanged using the ZTERM mode of ZMODEM. The shell process communicates with the session manager in order to invoke ZMODEM and ZTERM activities.

2.2.6 Constants & Conventions

The Omen source material uses octal in most cases, the form being 0xxx. Other sources use hexadecimal, the form being 0xhh. Omen octal values have a hex form placed in parentheses after them. The original was left so that if the author made a conversion error, the information would not be lost.

Frequently used constants (such as carriage-return, **CR**) or headers with a unique name (such as a **ZFILE** header) will be listed in boldface type.

2.3 Implementation Notes

The Physio implementation of ZMODEM does NOT support **ZSINIT, ZCRC, ZCHALLENGE**, or **ZFREECNT.**

The File Type field is not used in Physio **ZFILE** headers.

2.3.1 Device Capabilities

In the **ZRINIT/ZFILE** exchange, the receiver [host device] and sender [cart device] negotiate what functions are supported. (see section 4.4.2 - ZRINIT and section 4.4.4 - ZFILE).

(a) All Physio receiving devices will set the CANFDX and CANOVIO bits - all others will be cleared.

(b) All Physio transmitting devices will set the ZFILE byte ZF0 to ZCBIN = 0x01 (Binary transfer mode) and byte ZF1 to ZMCLOB = 0x04 (*Unconditional overwrite*). Bytes ZF2 and ZF3 will be cleared.

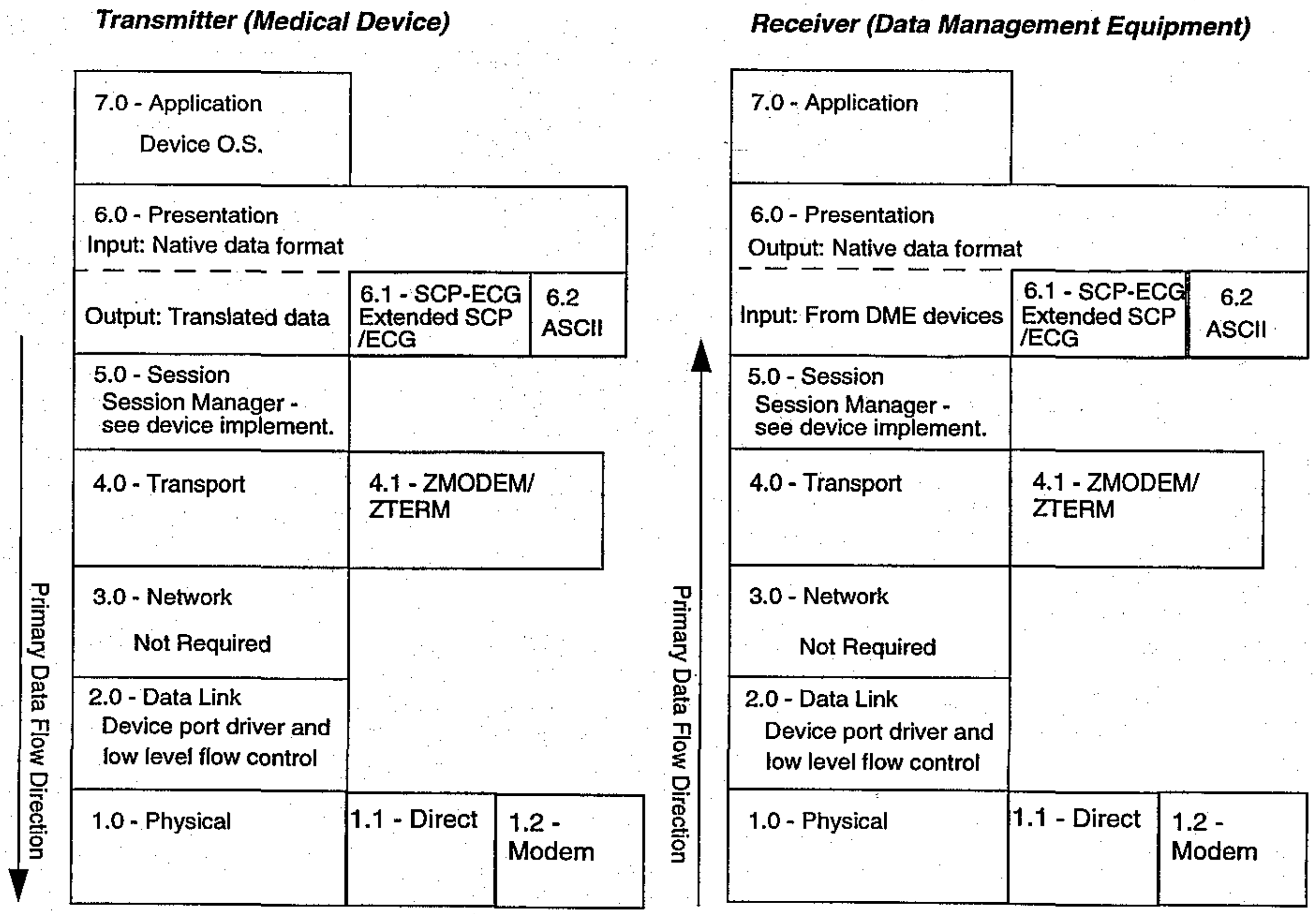
2.3.2 ZTERM Capabilities

For ZTERM-based transfer of shell commands, the sender and receiver roles reverse as commands are exchanged. All Physio devices will be bound by the device capabilities stated in the preceding paragraph, regardless of the command send/receive status.

3 Interface -- OSI/ISO Model for Physio-Control

The model used to bound the implementation of ZMODEM is the OSI/ISO Model. Below is the graphical representation of this model. In this model, the term 'Transmitting Device' describes a device which contains stored patient data to transfer, and 'Receiving Device' is the destination point for the data.

OSI 7 Layer Network Model


Transmitter (Medical Device)
7.0 - Application
Device O.S.
6.0 - Presentation
Input: Native data format
Output: Translated data
6.1 - SCP-ECG Extended SCP /ECG
6.2 ASCII
5.0 - Session
Session Manager - see device implement.
4.0 - Transport
4.1 - ZMODEM/ ZTERM
3.0 - Network
Not Required
2.0 - Data Link
Device port driver and low level flow control
1.0 - Physical
1.1 - Direct
1.2 - Modem
Primary Data Flow Direction
Receiver (Data Management Equipment)
7.0 - Application
6.0 - Presentation
Output: Native data format
Input: From DME devices
6.1 - SCP-ECG Extended SCP /ECG
6.2 ASCII
5.0 - Session
Session Manager - see device implement.
4.0 - Transport
4.1 - ZMODEM/ ZTERM
3.0 - Network
Not Required
2.0 - Data Link
Device port driver and low level flow control
1.0 - Physical
1.1 - Direct
1.2 - Modem
Primary Data Flow Direction

4 Layer 4.0 - Transport Layer Protocol Description

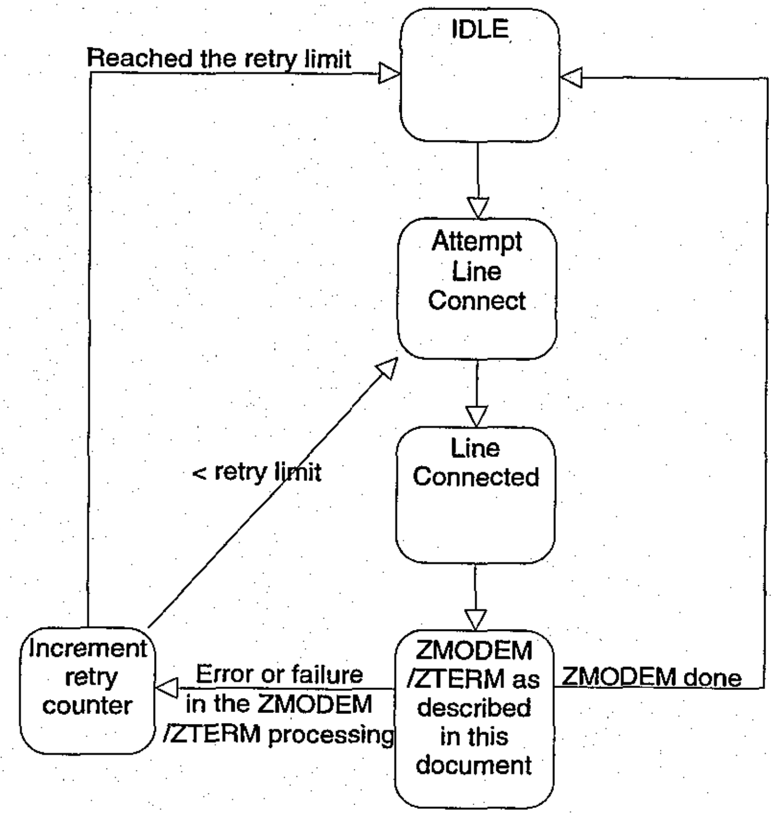
A transport layer protocol is responsible for end to end integrity of data. No check of data validity is performed, this is the responsibility of the presentation and application layers.

ZMODEM actually spans Level 4 and Level 2. The packet to packet integrity data check is a Level 2 function of ZMODEM, while the overall integrity of files and resumption of an interrupted file transmission is a Level 4 function.

4.1 ZMODEM Overview

Before starting the description of ZMODEM in this chapter and ZTERM in the next, the following diagram shows the context of this document in relation to the rest of the data communications process.

SCOPE



4.1.1 Description

The ZMODEM file transfer protocol is a bidirectional protocol that can be initiated from either the sender [cart device],e.g, LP12 or LP500; or the receiver [host device], e.g., Grapevine. It treats data entities to be transferred as files which are divided into packets for transmission. Error detection is applied to each packet.

ZMODEM can transmit file data in variable packet lengths. The initial packet length is selected via the protocol configuration parameters and then the length is dynamically modified during a transmission based on the transmission characteristic of the connection. ZMODEM decomposes the file into variable length blocks and adds headers to the transmitted data packets.

Header packets include command and control information, controlling the initiation and conduct of the session. Data packets are sent following an exchange of headers which define the data to be transferred and the type of exchange that will be performed. ZMODEM has several modes for the exchange of data from taking advantage of benign communications channels to reduce protocol overhead to performing acknowledgment of each data packet when the error rate of transmissions are high.

ZMODEM has two distinct processes; receive and transmit. At initial contact, the sender [cart device] and receiver [host device] exchange header packets to determine the parameters acceptable to each and to define the file to be transferred. Once the initial sequence is complete, data exchange begins under the control of the transmitter. A batch mode for file transfer is also supported.

OMEN With equivalent binary (efficient) and hex (application friendly) frame headers, the sending program can send a hex "invitation to receive" sequence to activate the receiver [host device] without crashing the remote application with unexpected control characters. ZMODEM's ability to accept options from both the sender [cart device] and receiver [host device] further simplifies application development.

OMEN Standard ZMODEM requires an 8 bit transfer medium. ZMODEM escapes network control characters to allow operation with packet switched networks. In general, ZMODEM operates over any path that supports XMODEM, and over many that don't.

OMEN To support full streaming,[1] the transmission path should either assert flow control or pass full speed transmission without loss of data. Otherwise the ZMODEM sender [cart device] must manage the window size.

4.1.2 Throughput

OMEN All file transfer protocols make trade-offs between throughput, reliability, universality, and complexity according to the technology and knowledge base available to their designers.

OMEN In the design of ZMODEM, three applications deserve special attention.

OMEN 1. Network applications with significant delays (relative to character transmission time) and low error rate.

OMEN 2. Timesharing and buffered modem applications with significant delays and throughput that is quickly degraded by reverse channel traffic. ZMODEM's economy of reverse channel bandwidth allows modems that dynamically partition bandwidth between the two directions to operate at optimal speeds. Special ZMODEM features allow simple, efficient implementation on a wide variety of timesharing hosts.

OMEN 3. Direct modem to modem communications with high error rate.

OMEN Unlike Sliding Windows Kermit, ZMODEM is not optimized for optimum throughput when error rate and delays are both high. This trade-off markedly reduces code complexity and memory requirements. ZMODEM generally provides faster error recovery than network compatible XMODEM implementations.

ZMODEM can accommodate network and timesharing system delays by continuously transmitting data (streaming mode) unless the receiver [host device] interrupts the sender [cart device] to request retransmission of garbled data.

4.1.3 Special Sequences

ZMODEM has special strings or sequences which are defined here and used later in this section.

4.1.3.1 Cancel Sequence

The Cancel sequence consists of ten **CAN** characters and ten backspace characters. A null character may be appended to the end of the string to allow the use of string-based programming functions. ZMODEM only requires five Cancel characters, the other five are "insurance".

Trailing backspace characters are normally added to attempt to erase the effects of the **CAN** characters if they are received by a command interpreter.

[1] With XOFF and XON, or out of band flow control such as X.25 or CTS.

4.1.3.2 Attention Sequence

The receiver [host device] may send the **Attn** sequence where it detects an error and needs to interrupt the sending program. The default **Attn** string value is empty (no Attn sequence). The receiver [host device] resets **Attn** to the empty default before each transfer session.

The sender [cart device] specifies the Attn sequence in its optional **ZSINIT** frame. The **Attn** string is terminated with a null.

Two meta-characters perform special functions in the **Attn** sequence:

0335 (0xdd), Send a break signal
0336 (0xde), Pause one second

ZSINIT and the **ATTN** sequence are not used in the Physio implementation.

4.2 ZMODEM BASICS

All software references in this section are to Omen/Forsberg's code.

4.2.1 Link Escape Encoding

OMEN ZMODEM achieves data transparency by extending the 8 bit character set (256 codes) with escape sequences based on the ZMODEM data link escape character **ZDLE.**[1]

Rationale: Link Escape coding is performed by inserting the **ZDLE** character immediately preceding the character to be link escaped.

OMEN Link Escape coding permits variable length data subpackets without the overhead of a separate byte count. It allows the beginning of frames to be detected without special timing techniques, facilitating rapid error recovery.

OMEN Link Escape coding does add some overhead. The worst case, a file consisting entirely of escaped characters, would incur a 50% overhead.

OMEN The **ZDLE** character is special. **ZDLE** represents a control sequence of some sort. If a **ZDLE** character appears in binary data, it is prefixed with **ZDLE**, then sent as **ZDLEE**. Note: **ZDLEE** is defined in section 6.1, ZMODEM.H - Manifest constants for ZMODEM file transfer protocol.

OMEN The value for **ZDLE** is octal 030 (ASCII **CAN**). This particular value was chosen to allow a string of 5 consecutive CAN characters to abort a ZMODEM session, compatible with YMODEM session abort.

OMEN Since **CAN** is not used in normal terminal operations, interactive applications and communications programs can monitor the data flow for **ZDLE**.

4.2.1/1 When a **ZDLE** character appears in a binary data field to be transmitted, it shall be prefixed with a **ZDLE** character, converting it to the **ZDLEE** character string.
NON-STATE MACHINE REQUIREMENT

Rationale: Because the **ZDLE** character can appear anywhere in a binary data field, it must be converted to the **ZDLEE** string to prevent the ZMODEM protocol from treating the isolated **ZDLE** character as the start of a command/control sequence.

Further references in this document to **ZDLE** characters will not apply to the second character of the **ZDLEE** string.

[1] This and other constants are defined in 6.1, ZMODEM.H - Manifest constants for ZMODEM file transfer protocol. Please note that constants with a leading 0 are octal constants in C.

4.2.1/2 Receiving 5 consecutive **ZDLE** characters (equivalent to the ASCII **CAN** character) shall cause the current ZMODEM session to be aborted.
NON-STATE MACHINE REQUIREMENT

Rationale: This abort sequence is compatible with the YMODEM session abort command.

OMEN The receiving program decodes any sequence of **ZDLE** followed by a byte with bit 6 set and bit 5 reset (upper case letter, either parity) to the equivalent control character by inverting bit 6. This allows the transmitter to escape any control character that cannot be sent by the communications medium. In addition, the receiver recognizes link escapes for 0177 and 0377 should these characters need to be escaped.

OMEN ZMODEM software link escapes **ZDLE**, 020, 0220, 021, 0221, 023, and 0223. If preceded by 0100 or 0300 (@), 015 and 0215 are also link escaped to protect the Telenet command escape CR-@-CR. The receiver ignores 021, 0221, 023, and 0223 characters in the data stream.

OMEN The ZMODEM routines in zm.c accept an option to link escape all control characters, to allow operation with less transparent networks. This option can be given to either the sending or receiving program.

4.2.1/3 A character with bit 6 set and bit 5 reset, following a **ZDLE**, shall be converted by the receiver [host device] to a single character with bit 6 reset and all other bits unchanged.
NON-STATE MACHINE REQUIREMENT

Rationale: A link-escaped 7-bit upper-case letter character of any parity is thereby converted to a control character, allowing the sender [cart device] to send control characters from system to system that would otherwise be intercepted as commands by the communications medium.

4.2.1/4 If link-escaped, the octal (hex) characters:

0020 (0x10) and 0220 (0x90),
0021 (0x11) and 0221 (0x91),
0023 (0x13) and 0223 (ox93),
0177 (0x7f) and 0377 (oxff),

shall be converted on receipt to a single character with bit 6 reset and all other bits unchanged.
NON-STATE MACHINE REQUIREMENT

Rationale: These characters are sometimes used as control characters and need to be link-escaped as described in Requirement 4.2.1/1.

4.2.1/5 The octal (hex) characters 0015 (0x0d) and 0215 (0x4d), the ASCII **CR,** shall be link-escaped if they are preceded by either a 0100 (0x40) or 0300 (oxc0), the ASCII @ character.
NON-STATE MACHINE REQUIREMENT

Rationale: The Telenet communications system uses the sequence **'CR-@-CR'** as an escape sequence for commands. By link-escaping the second **CR** in the string, this protects the Telenet link.

4.2.1/6 The option to link-escape all ASCII control characters (those characters with bits 5 and 6 reset) shall be implemented in both transmit and receive modes.
NON-STATE MACHINE REQUIREMENT

Rationale: Some communications links may interpret control characters whenever used. By link-escaping all control characters, these links are protected. Control characters are first converted to upper-case letters by setting bit 5.

OMEN The ZMODEM routines in zm.c accept an option to link escape all control characters, to allow operation with less transparent networks. This option can be given to either the sending or receiving program.

4.2.2 Header

OMEN All ZMODEM frames begin with a header which may be sent in binary or HEX form. ZMODEM uses a single routine to recognize binary and hex headers.

OMEN All ZMODEM header packets will consist of a Start-of-Frame (ZPAD,ZPAD,ZDLE) followed by a Frame-Format constant, the Frame-Type, four parameter bytes, and a 16-bit or 32-bit CRC, as specified by the frame format constant. *The basic format is as follows:*

4.2.2/1 All ZMODEM headers shall contain 2 **ZPAD**s, a **ZDLE**, a format byte, a type byte, 4 parameter bytes, and a CRC as shown in table 1 below.
NON-STATE MACHINE REQUIREMENT

Commentary: This format description only applies prior to encoding or after decoding.

4.2.2/2 Extra **ZPAD** characters received at the start of a header shall be ignored.
NON-STATE MACHINE REQUIREMENT

Commentary: Kientzle's implementation allows up to 25000 ZPADs in this circumstance.

Table 1 Basic Format of a ZMODEM Hex *Header* Frame

Start of Frame			Format	Type	Parameters (*Note 3, 4*)				CRC	
ZPAD	ZPAD	ZDLE	*(Note 1)*	*(Note 2)*	*Z3*	*Z2*	*Z1*	*Z0*	CRCH	CRCL

Note 1 - The format byte is either **ZBIN** or **ZHEX.**
Note 2 - The header type byte is as defined in 4.4, FRAME TYPES
Note 3 - All parameters that are unused or uninitialized are to be set to a value of zero.
Note 4 - Parameters Z3...Z0 consist of:

ZFx[1] = ZMODEM Flags which are transmitted in the parameter section of a header
ZPx = ZMODEM file Position which are also transmitted in the parameter section of a header
The order of transmission of ZFx and ZPx are different. The orders are as follows:

OMEN Either form of the header contains the same raw information, a type byte and four bytes of data indicating flags and/or numeric quantities depending on the frame type:

OMEN TYPE: frame type
OMEN F3: Flags most significant byte
OMEN F0: Flags least significant byte
OMEN P0: file Position least significant
OMEN P3: file Position most significant
OMEN TYPE F3 F2 F1 F0
OMEN or ----------------------
OMEN TYPE P0 P1 P2 P3

OMEN Figure 1. Order of Bytes in Header

Table 2 The ordering of ZFx and ZPx Parameters

Start of Frame			Format	Type	Parameters				CRC	
-	-	-	-	-	ZF3	ZF2	ZF1	ZF0	CRCH	CRCL
-	-	-	-	-	ZP0	ZP1	ZP2	ZP3	CRCH	CRCL

[1] x = 0,1,2, or 3 as in ZF0, ZF1... ZP0, ZP1...

4.2.2.1 16 Bit CRC Binary Header

Example of a ZMODEM *Binary* Header Packet

Start of Frame			Format	Type	Parameters				CRC	
ZPAD	ZPAD	ZDLE	ZBIN	ZFILE	ZF3	ZF2	ZF1	ZF0	High	Low
‘*’	‘*’	0x018	‘A’	0X004	0x00	0x00	0x00	0x00	0x89	0x06

Data packets following 16-bit CRC headers will use 16-bit CRC's as described in section 4.5.1, ZMODEM 16-Bit CRC polynomial. Data packets following 32-bit headers will use 32-bit CRC's.

OMEN A binary header is sent by the sending program to the receiving program. **ZDLE** encoding accommodates XON/XOFF flow control.

OMEN The frame type byte is ZDLE encoded.

OMEN The four position/flags bytes are **ZDLE** encoded.

OMEN A 16 bit CRC of the frame type and position/flag bytes is **ZDLE** encoded.

OMEN 0 or more binary data subpackets with 16 bit CRC will follow depending on the frame type.

4.2.2.2 32 Bit CRC Binary Header

OMEN A "32 bit CRC" Binary header is similar to a Binary Header, except the **ZBIN** (A) character is replaced by a **ZBIN32** (C) character, and four characters of CRC are sent.

Commentary: This is not supported by Physio-Control.

4.2.2.3 Hex Header

Example of a ZMODEM *Hex* Header Packet

Start of Frame			For-mat	Type	Parameters				CRC		ASCII Trailer		
ZPAD	ZPAD	ZDLE	ZHEX	ZRQINIT	ZF3	ZF2	ZF1	ZF0	High	Low	CR	LF	Xon*
0X052	0X052	0x018	'B'	0x00	0x00	0x00	0x00	ox00	0x00	0x00	0x0D	0x8A	0x11

*Xon is not sent for the headers of type ZACK and ZFIN.

OMEN The receiver sends responses in hex headers. The sender also uses hex headers when they are not followed by binary data subpackets.

OMEN Hex encoding protects the reverse channel from random control characters. The hex header receiving routine ignores parity; except to detect the presence of a 7-bit transmission path.

OMEN Use of Kermit-style encoding for control and paritied characters was considered and rejected because of increased possibility of interacting with some timesharing systems' line edit functions. Use of HEX headers from the receiving program allows control characters to be used to interrupt the sender when errors are detected. A HEX header may be used in place of a binary header wherever convenient. If a data subpacket follows a HEX header, it is protected with CRC-16.

OMEN A hex header begins with the sequence **ZPAD, ZPAD, ZDLE, ZHEX.** The zgethdr routine synchronizes with the **ZPAD-ZDLE** sequence. The extra **ZPAD** character allows the sending program to scan for a **ZPAD** character, to detect a header (indicating an error condition) and then call zgethdr to receive the header.

OMEN The following are definitions, not requirements.

OMEN The type byte, the four position/flag bytes, and the 16 bit CRC thereof are sent in hex using the character set 01234567890abcdef. Upper case hex digits are not allowed; they false trigger XMODEM and YMODEM programs.

OMEN A carriage return and line feed are sent with HEX headers. The receive routine expects to see at least one of these characters, two if the first is CR.

OMEN The receiver carefully checks hex headers for violations of this format. Since the constraints and redundancy of this hex encoding detect many patterns of errors, especially missing characters, a hex header with 32 bit CRC has not been defined.

OMEN The CR/LF aids debugging from printouts, and helps overcome certain operating system related problems. The LF is sent with 8th bit set to aid the receiver in checking for 7 bit paths.

OMEN An XON character is appended to all HEX headers except **ZACK** and **ZFIN**.

OMEN The XON releases the sender from spurious XOFF flow control characters generated by line noise, a common occurrence. XON is not sent after **ZACK** headers to protect flow control in streaming situations. XON is not sent after a **ZFIN** header to allow clean session cleanup. The receiver may or may not receive the XON character depending on the flow control arrangements.

4.2.2.3/1 When parsing a hex header, the receiver [host device] shall ignore the state of the parity bit.
NON-STATE MACHINE REQUIREMENT

Rationale: 7-bit communications channels use bit 7 for byte parity. When a hex header is used, bit 7 has no meaning with respect to ZMODEM functions and should be ignored.

4.2.2.3/2 For the type byte, the four position/flag bytes, and the 16-bit CRC, which are sent in hex, only the following character set shall be used: 0123456789abcdef.
NON-STATE MACHINE REQUIREMENT

4.2.3 Binary Data Subpackets

Since some characters may be link escaped, there wasn't any point wasting bytes to fill out a fixed packet length or to specify a variable packet length. In ZMODEM, the size of data subpackets is denoted by ending each subpacket with an escape sequence (0x18, 'K') similar to BISYNC and HDLC.

Data frames consist of a header followed by one or more data subpackets. In the absence of transmission errors, an entire file can be sent in one data frame.

Structure of the ZMODEM Data Sub-packet

A ZMODEM data sub-packet following a ZCOMMAND

Start of Frame			For-mat	TYPE	Parameters				CRC	
ZPAD	ZPAD	ZDLE	ZBIN	ZCOMMAND	ZF3	ZF2	ZF1	ZF0	High	Low

The Data sub-packet

Data	Frame-End		CRC	
0-1024 Bytes	ZDLE	ZCRCW	High	Low

OMEN Binary data subpackets immediately follow the associated binary header packet. A binary data subpacket contains 0 to 1024 bytes of data. Recommended length values are 256 bytes below 2400 bps, 512 at 2400 bps, and 1024 above 4800 bps or when the data link is determined to be relatively error free.[1]

OMEN No padding is used with binary data subpackets. The data bytes are **ZDLE** encoded and transmitted. A **ZDLE** and frame end are then sent, followed by two or four **ZDLE** encoded CRC bytes. The CRC accumulates the data bytes and frame end, not including CAN.

[1] Strategies for adjusting the subpacket length for optimal results based on real time error rates are still evolving. Shorter subpackets speed error detection but increase protocol overhead slightly.

4.3 ZMODEM Operation

4.3.1 Recommended Time-Outs

The requirements in this section apply to all subsequent state diagrams.

OMEN As with the XMODEM recommendation, ZMODEM timing is receiver driven. The transmitter should not time out at all, except to abort the program if no headers are received for an extended period of time, say one minute.[1]

OMEN It goes without saying that both the transmitter and receiver should check carrier detect or equivalent signals often enough to properly exit in the event of a disconnect. Keep in mind that in ZMODEM's most efficient mode of operation, no response is expected from the receiver during the entire file transfer except for error correction!

4.3.1/1 A header response timer shall measure the time between the completion of the transmission of the final byte of a header packet and the receipt of the first byte of a response.
NON-STATE MACHINE REQUIREMENT

Commentary: Software and operating system latencies are not included in the above requirement.

4.3.1/2 If the header response timer expires before a required response to a valid and appropriate header is received, the header shall be retransmitted; except during file transfer.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

4.3.1/3 If the header response timer expires following the tenth consecutive retransmission of a header, the device attempting to send the header shall abort the session by sending **ZCAN**, section 4.1.3.1, Cancel Sequence.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

Commentary: The requirement above requires ten consecutive time-out-driven retries to abort the session. If a response header is received between retries, the re-try counter is reset.

While Omen does not specify how to abort a session, Kientzle simply quits the session without sending anything. The assumption is that since there was no response to the header, then the receiver is not receiving anything and sending an abort message is useless, but harmless.

The ZMODEM session may be terminated without sending anything or by sending a **CAN** sequence. The communications manager may try to send the record again, which may or may not require termination and re-establishment of the physical connection depending on the circumstances of the implementation.

4.3.1/4 If during a file transfer, the data line is quiescent for >= 10 seconds, the receiver shall attempt to restart the transfer using the ZRPOS header.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

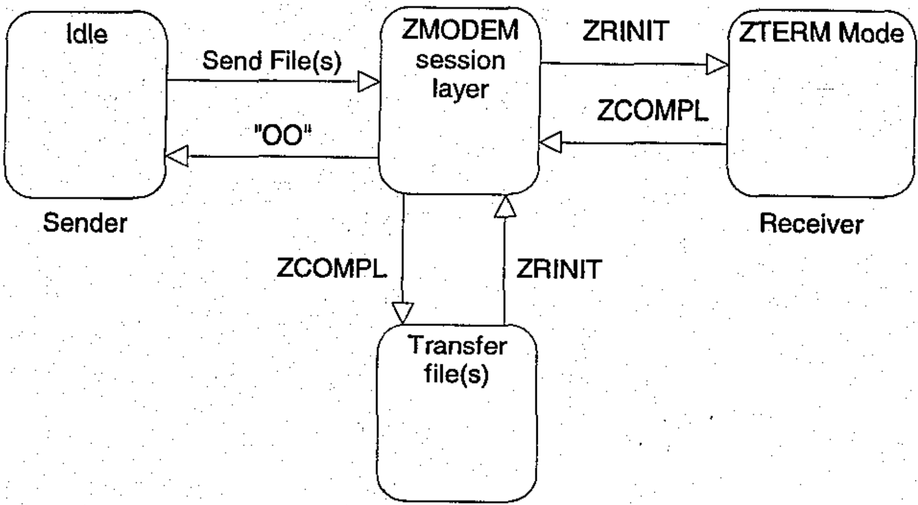
Rationale: Attempting to restart the transfer with a header places the session under the packet retry rules. An unsuccessful restart will result in session termination per 4.3.1/4.

[1] Special considerations apply when sending commands.

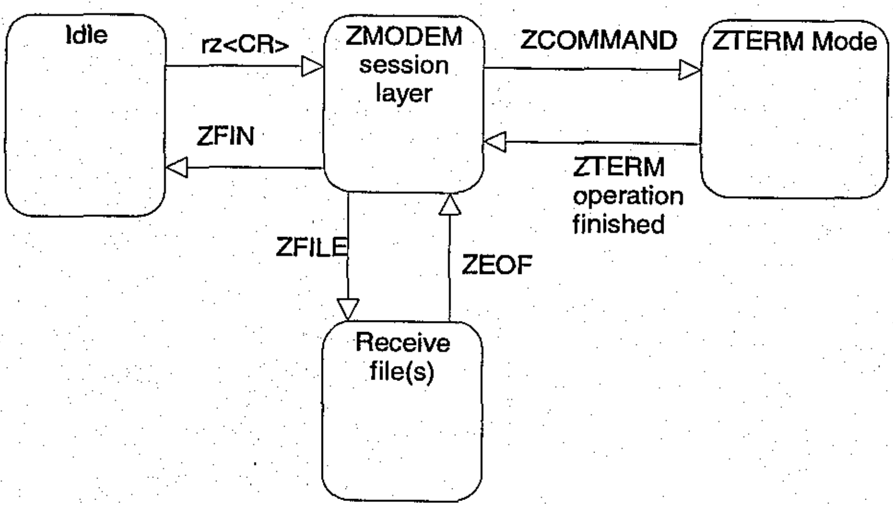
4.3.2 Session Start-up

The following state diagram shows the basic functions performed by the sender and receiver:

ZMODEM/ZTERM Interface - Sender



ZMODEM/ZTERM Interface - Receiver



4.3.2.1 Transmit ‘rz’<cr>, and ZRQINIT

OMEN To start a ZMODEM file transfer session, the sending program is called with the names of the desired file(s) and option(s).

OMEN The sending program may send the string "rz\cr" to invoke the receiving program from a possible command mode. The "rz" followed by carriage return activates a ZMODEM receive program or command if it were not already active. (note: the “rz\cr” command is used to start a ZMODEM receiving program on a Unix system, hence the concept of a receiver not being active, ZMODEM tries to start on Unix systems by talking to the command shell. Ed.)

OMEN The sender may then display a message intended for human consumption, such as a list of the files requested, etc.

OMEN Then the sender sends a **ZRQINIT** header. The **ZRQINIT** header causes a previously started receive program to send its **ZRINIT** header without delay.

OMEN RZ is a command to start ZMODEM receive program. It is sent in case the receiver is in terminal mode.

OMEN In an interactive or conversational mode, the receiving application may monitor the data stream for **ZDLE**. The following characters may be scanned for "B00" indicating a **ZRQINIT** header, a command to download a command or data.

OMEN The sending program awaits a command from the receiving program to start file transfers. If a "C", "G", or NAK is received, an XMODEM or YMODEM file transfer is indicated, and file transfer(s) use the YMODEM protocol. Allowing this fall back to XMODEM or YMODEM compromises reliability and is not recommended as a default. Note: With ZMODEM and YMODEM, the sending program provides the file name, but not with XMODEM.

OMEN In case of garbled data, the sending program can repeat the invitation to receive a number of times until a session starts.

OMEN When the ZMODEM receive program starts, it immediately sends a **ZRINIT** header to initiate ZMODEM file transfers, or a **ZCHALLENGE** header to verify the sending program. The receive program resends its header at response time (default 10 second) intervals for a suitable period of time (40 seconds total) before falling back to YMODEM protocol.

OMEN If the receiving program receives a **ZRQINIT** header, it sends the **ZRINIT** header. If the sending program receives the **ZCHALLENGE** header, it places the data in ZP0...ZP3 in an answering **ZACK** header.

OMEN If the receiving program receives a **ZRINIT** header, it is an echo indicating that the sending program is not operational.

OMEN Eventually the sending program correctly receives the **ZRINIT** header.

OMEN The sender may then send an optional **ZSINIT** frame to define the receiving program's **Attn** sequence, or to specify complete control character escaping. If the receiver specifies the same or higher level of escaping, the ZSINIT frame need not be sent unless an Attn sequence is needed.

OMEN If the **ZSINIT** header specifies ESCCTL or ESC8, a HEX header is used, and the receiver activates the specified ESC modes before reading the following data subpacket.

OMEN The receiver sends a **ZACK** header in response, containing either the serial number of the receiving program, or 0.

4.3.2.1/1 To initiate a ZMODEM session, the transmitting device shall send 'rz' (ASCII characters) followed by an ASCII **CR**, followed by a **ZRQINIT** header.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

Commentary: This is defined as the ZMODEM initiation sequence.

Commentary: Should the initiation sequence go unanswered, retransmissions of the **ZRQINIT** header do not require a preceding 'rz<**CR**>'.

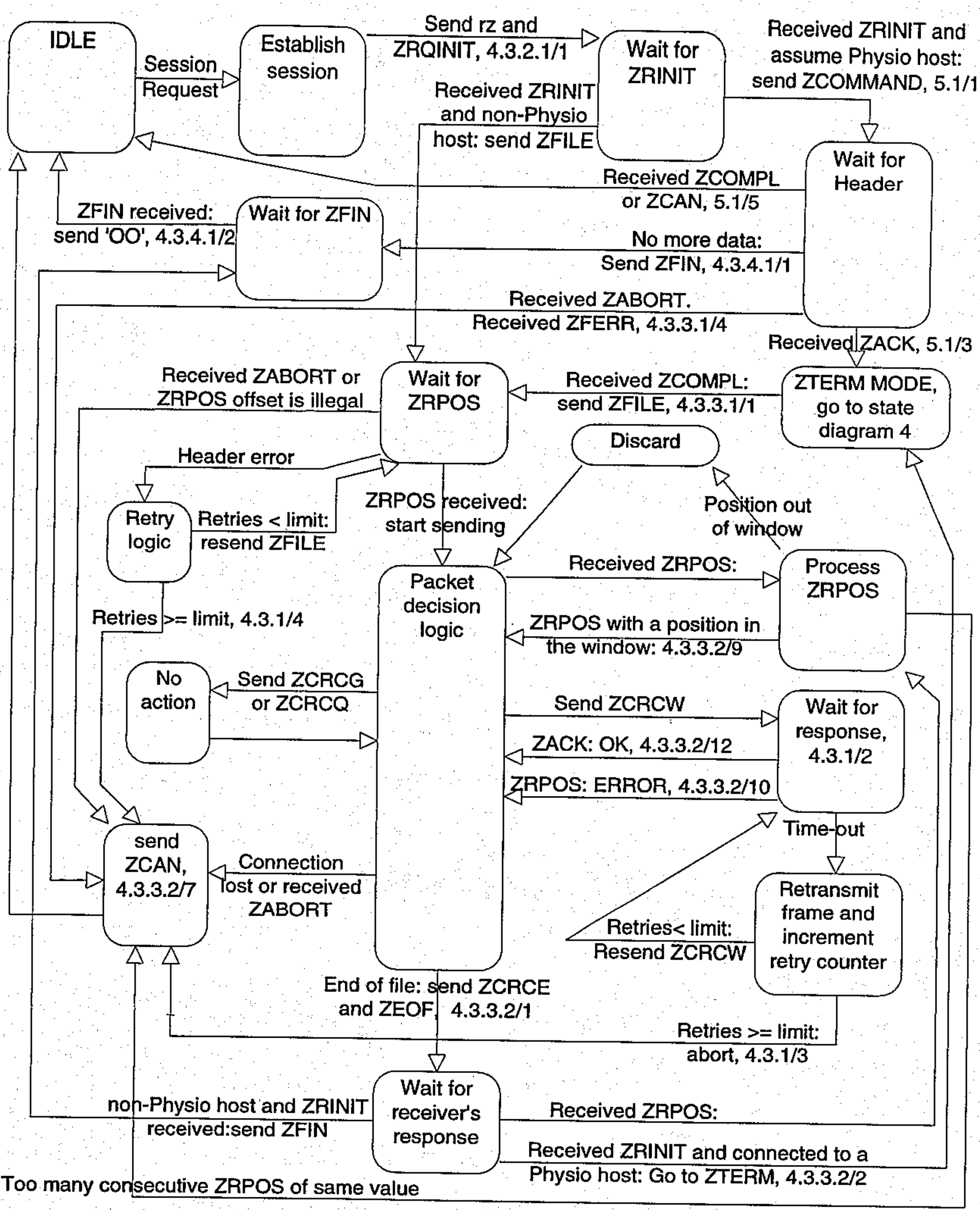
4.3.2.1/2 Following receipt of the ZMODEM initiation sequence, the receiver [host device] will respond with a **ZRINIT** header.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

4.3.2.1/3 The receiver's retry mechanics for establishing the ZMODEM session shall be the same as defined in section 4.3.1, Recommended Time-Outs.
NON-STATE MACHINE REQUIREMENT.

4.3.3 File Transmission

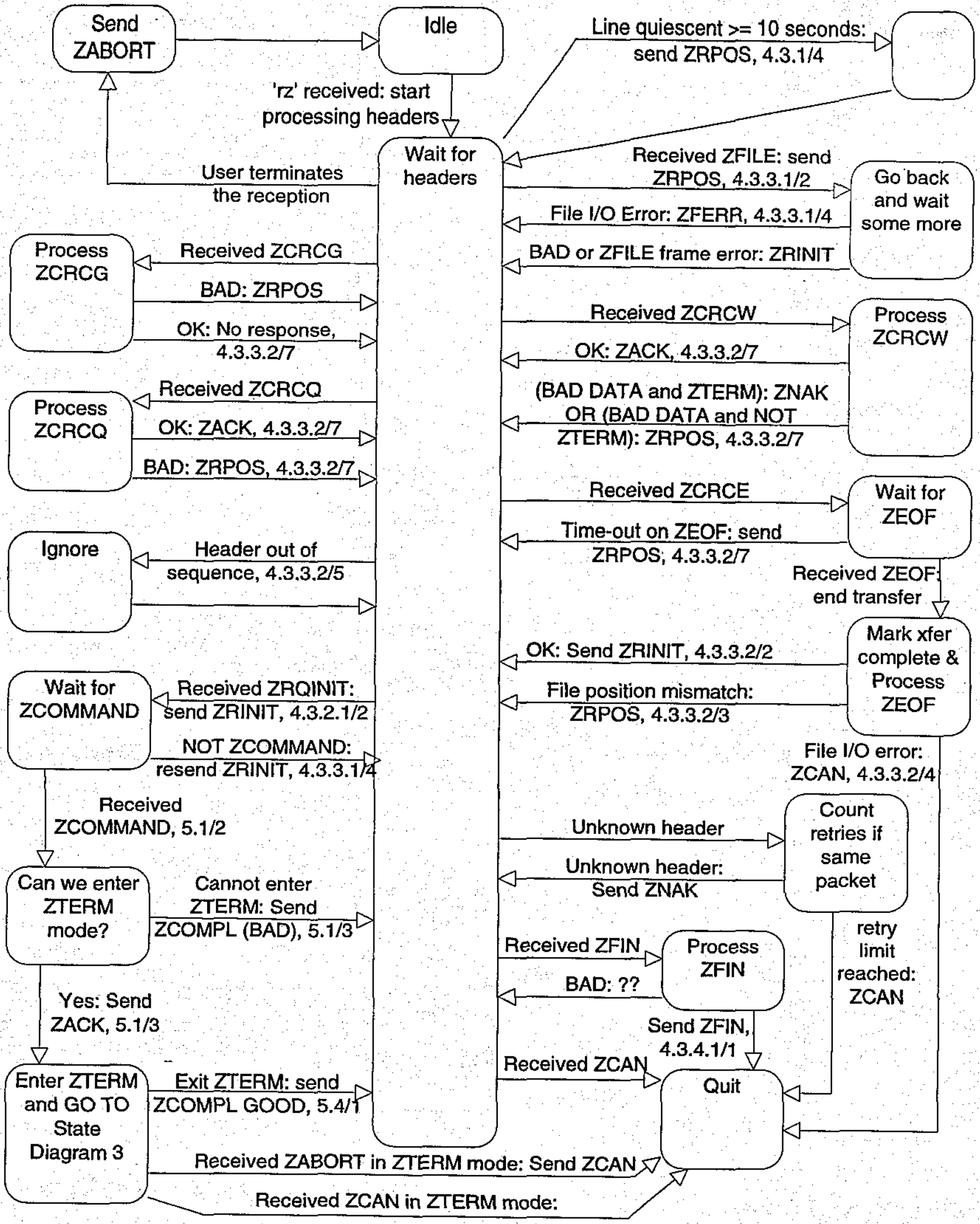
The following two state diagrams show the operation of the ZMODEM state machine, first from the sender's point of view and then from the receiver.

ZMODEM State Diagram 1, SENDER


IDLE
Session Request
Establish session
Send rz and ZRQINIT, 4.3.2.1/1
Wait for ZRINIT
Received ZRINIT and assume Physio host: send ZCOMMAND, 5.1/1
Received ZRINIT and non-Physio host: send ZFILE
Wait for Header
Received ZCOMPL or ZCAN, 5.1/5
ZFIN received: send 'OO', 4.3.4.1/2
Wait for ZFIN
No more data: Send ZFIN, 4.3.4.1/1
Received ZABORT. Received ZFERR, 4.3.3.1/4
Received ZACK, 5.1/3
Received ZABORT or ZRPOS offset is illegal
Wait for ZRPOS
Received ZCOMPL: send ZFILE, 4.3.3.1/1
ZTERM MODE, go to state diagram 4
Discard
Header error
Retry logic
Retries < limit: resend ZFILE
ZRPOS received: start sending
Position out of window
Packet decision logic
Received ZRPOS:
Process ZRPOS
Retries >= limit, 4.3.1/4
ZRPOS with a position in the window: 4.3.3.2/9
No action
Send ZCRCG or ZCRCQ
Send ZCRCW
Wait for response, 4.3.1/2
ZACK: OK, 4.3.3.2/12
ZRPOS: ERROR, 4.3.3.2/10
Time-out
send ZCAN, 4.3.3.2/7
Connection lost or received ZABORT
Retransmit frame and increment retry counter
Retries< limit: Resend ZCRCW
End of file: send ZCRCE and ZEOF, 4.3.3.2/1
Retries >= limit: abort, 4.3.1/3
Wait for receiver's response
non-Physio host and ZRINIT received:send ZFIN
Received ZRPOS:
Received ZRINIT and connected to a Physio host: Go to ZTERM, 4.3.3.2/2
Too many consecutive ZRPOS of same value

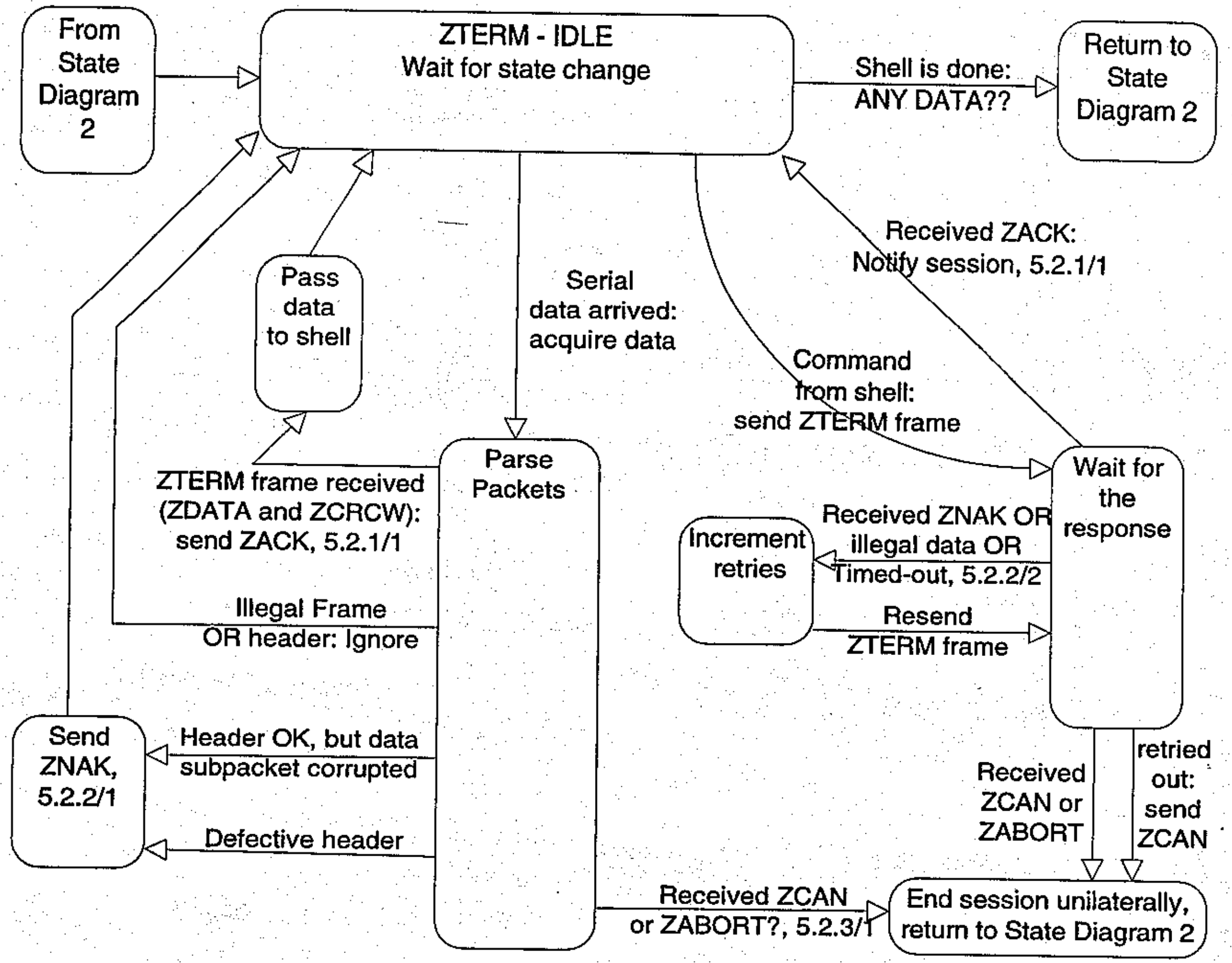
The requirements in section,4.3.1, Recommended Time-Outs apply to all states in this diagram.

ZMODEM State Diagram 2, RECEIVER


Send ZABORT
Idle
Line quiescent >= 10 seconds: send ZRPOS, 4.3.1/4
'rz' received: start processing headers
Wait for headers
User terminates the reception
Received ZFILE: send ZRPOS, 4.3.3.1/2
Go back and wait some more
File I/O Error: ZFERR, 4.3.3.1/4
BAD or ZFILE frame error: ZRINIT
Process ZCRCG
Received ZCRCG
BAD: ZRPOS
OK: No response, 4.3.3.2/7
Received ZCRCW
Process ZCRCW
OK: ZACK, 4.3.3.2/7
(BAD DATA and ZTERM): ZNAK OR (BAD DATA and NOT ZTERM): ZRPOS, 4.3.3.2/7
Process ZCRCQ
Received ZCRCQ
OK: ZACK, 4.3.3.2/7
BAD: ZRPOS, 4.3.3.2/7
Received ZCRCE
Wait for ZEOF
Time-out on ZEOF: send ZRPOS; 4.3.3.2/7
Ignore
Header out of sequence, 4.3.3.2/5
Received ZEOF: end transfer
Mark xfer complete & Process ZEOF
OK: Send ZRINIT, 4.3.3.2/2
File position mismatch: ZRPOS, 4.3.3.2/3
Wait for ZCOMMAND
Received ZRQINIT: send ZRINIT, 4.3.2.1/2
NOT ZCOMMAND: resend ZRINIT, 4.3.3.1/4
File I/O error: ZCAN, 4.3.3.2/4
Received ZCOMMAND, 5.1/2
Unknown header
Count retries if same packet
Unknown header: Send ZNAK
Can we enter ZTERM mode?
Cannot enter ZTERM: Send ZCOMPL (BAD), 5.1/3
retry limit reached: ZCAN
Received ZFIN
Process ZFIN
BAD: ??
Yes: Send ZACK, 5.1/3
Send ZFIN, 4.3.4.1/1
Received ZCAN
Quit
Enter ZTERM and GO TO State Diagram 3
Exit ZTERM: send ZCOMPL GOOD, 5.4/1
Received ZABORT in ZTERM mode: Send ZCAN
Received ZCAN in ZTERM mode:

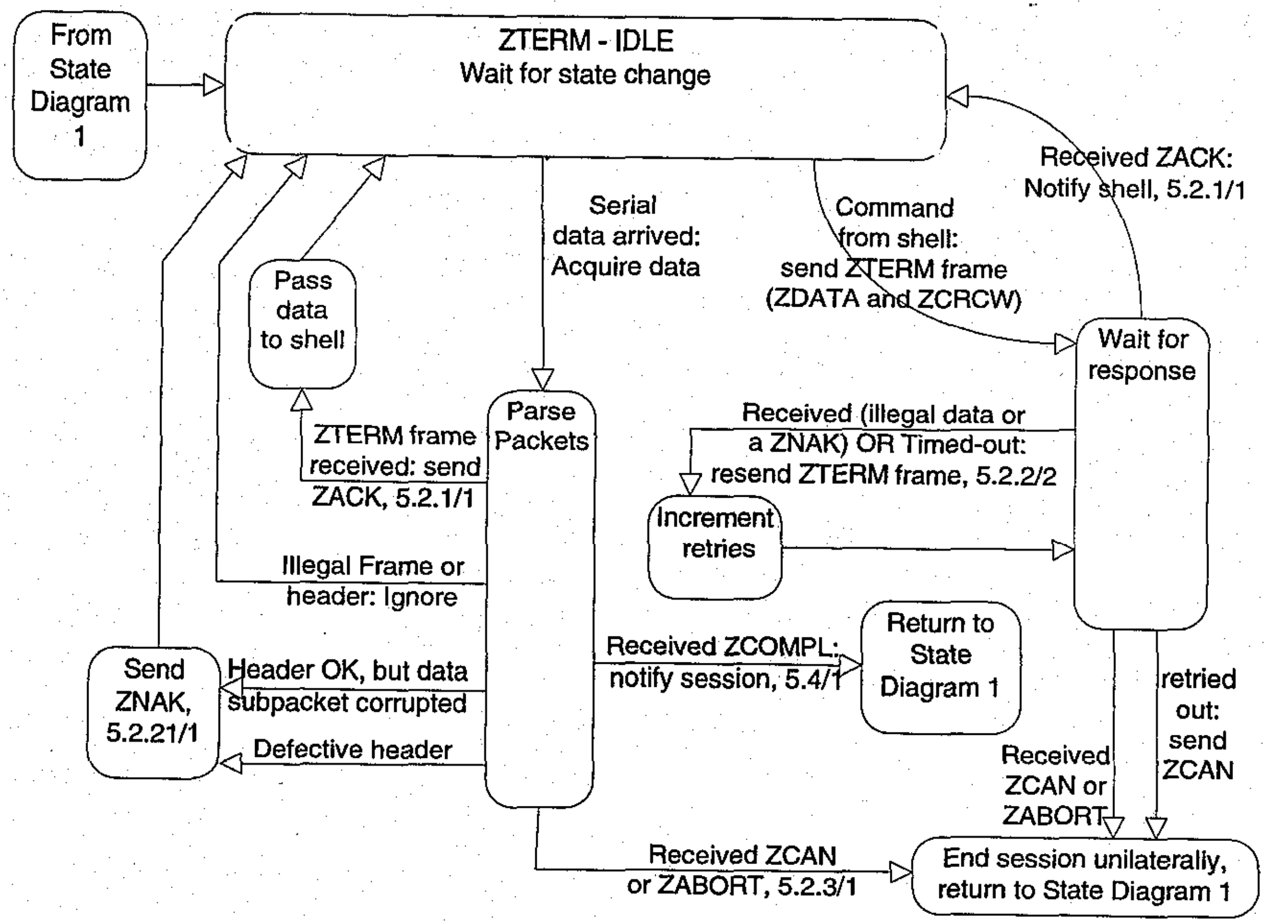
The requirements in section,4.3.1, Recommended Time-Outs apply to all states in this diagram.

ZMODEM State Diagram 3, ZTERM Receiver Implementation


From State Diagram 2
ZTERM - IDLE
Wait for state change
Shell is done:
ANY DATA??
Return to State Diagram 2
Received ZACK:
Notify session, 5.2.1/1
Pass data to shell
Serial data arrived: acquire data
Command from shell: send ZTERM frame
Parse Packets
Wait for the response
ZTERM frame received (ZDATA and ZCRCW): send ZACK, 5.2.1/1
Received ZNAK OR illegal data OR Timed-out, 5.2.2/2
Increment retries
Resend ZTERM frame
Illegal Frame OR header: Ignore
Send ZNAK, 5.2.2/1
Header OK, but data subpacket corrupted
Defective header
Received ZCAN or ZABORT
retried out: send ZCAN
Received ZCAN or ZABORT?, 5.2.3/1
End session unilaterally, return to State Diagram 2

The requirements in section,4.3.1, Recommended Time-Outs apply to all states in this diagram.

ZMODEM State Diagram 4, ZTERM Transmitter Implementation


From State Diagram 1
ZTERM - IDLE
Wait for state change
Received ZACK: Notify shell, 5.2.1/1
Serial data arrived: Acquire data
Command from shell: send ZTERM frame (ZDATA and ZCRCW)
Pass data to shell
Wait for response
Parse Packets
ZTERM frame received: send ZACK, 5.2.1/1
Received (illegal data or a ZNAK) OR Timed-out: resend ZTERM frame, 5.2.2/2
Increment retries
Illegal Frame or header: Ignore
Return to State Diagram 1
Received ZCOMPL: notify session, 5.4/1
Send ZNAK, 5.2.21/1
Header OK, but data subpacket corrupted
Defective header
retried out: send ZCAN
Received ZCAN or ZABORT
Received ZCAN or ZABORT, 5.2.3/1
End session unilaterally, return to State Diagram 1

The requirements in section,4.3.1, Recommended Time-Outs apply to all states in this diagram.

4.3.3.1 Transmit ZFILE Header and Data Subpackets

OMEN Section 8.2, *File Transmission*

OMEN The sender then sends a **ZFILE** header with ZMODEM Conversion, Management, and Transport options followed by a **ZCRCW** data subpacket containing the file name and possibly the file length, modification date, and other information identical to that used by YMODEM.

OMEN The receiver examines the file name, length, and date information provided by the sender in the context of the specified transfer options, the current state of its file system(s), and local security requirements. The receiving program should insure the pathname and options are compatible with its operating environment and local security requirements.

OMEN The receiver may respond with a **ZSKIP** header, which makes the sender proceed to the next file (if any) in the batch.

OMEN If the receiver has a file with the same name and length, it may respond with a **ZCRC** header with a byte count, which requires the sender to perform a 32 bit CRC on the specified number of bytes in the file and transmit the complement of the CRC in an answering **ZCRC** header.The CRC is initialized to 0xFFFFFFFF. A byte count of 0 implies the entire file. The receiver uses this information to determine whether to accept the file or skip it. This sequence may be triggered by the ZMCRC Management Option.

OMEN A **ZRPOS** header from the receiver initiates transmission of the file data starting at the offset in the file specified in the **ZRPOS** header. Normally the receiver specifies the data transfer to begin at offset 0 in the file.

OMEN If this information is received correctly by the receiver then it will be acknowledged by the receiver with a ZRPOS header which identifies the offset in the file where communications should take place.

OMEN If the receive process is unable to correctly receive the **ZFILE** header or the data subpacket it will either issue a **ZNAK** and wait for a good packet or issue a **CAN** sequence and terminate the transmission.

4.3.3.1/1 To initiate a file transfer after exiting ZTERM mode as described in section 5, Physio-Control ZTERM Mode, the sender [cart device] shall send a **ZFILE** header followed by a **ZCRCW** data packet.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

Commentary: The sender [cart device] begins any particular file transaction by offering a file to the receiver [host device]. The **ZFILE** header contains flags which guide the receiver [host device]. The data sub-packet contains information on the file itself: file name, file size, etc.

Commentary: This is defined as the file initiation sequence.

4.3.3.1/2 The receiver [host device] shall respond to the sender [cart device]'s file initiation sequence (**ZFILE**) with a **ZRPOS** header indicating the offset in bytes from the first byte of the file at which transmission of data shall begin.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

Commentary: At the beginning of file transfer, the offset contained in the **ZRPOS** header will be zero, indicating the start of the file.

4.3.3.1/3 After the receipt of a valid file initiation sequence, the receiver [host device] shall cause to be created a destination file to receive the sender [cart device] data as defined in the file initiation **ZFILE** header and data packet.
NON-STATE MACHINE REQUIREMENT.

4.3.3.1/4 If the receiver [host device] cannot create a destination file for the sender [cart device] data as described in 4.3.3.1/3, then it shall send a **ZFERR** header.
STATE MACHINE REQUIREMENT.

4.3.3.2 Transmit ZDATA header and Data Subpackets

OMEN The transmit process is now ready to inform the receiver that the data subpackets of the file to be transmitted will be coming next. This portion of the transmit process will send the contents of the transmitted file. The transmission of data subpackets will continue until:

OMEN 1. The transmitter has completed sending the selected file. In this case the last subpacket type will be ZCRCE (end subpacket). The receiver is waiting for a ZEOF header to complete the transaction.

OMEN 2. The transmitter needs some time to set up the next block of data for transmission. In this case the last subpacket type will be ZCRCW (wait subpacket). After a ZCRCW has been issued the transmitter must repeat the ZFILE header before any additional subpackets can be transmitted. In any case, the transmitter should wait for a ZACK from the ZCRCW before continuing with transmission.

OMEN The two other subpacket types are ZCRCG and ZCRCQ. The ZCRCQ data subpacket will cause the receiver to issue a ZACK each time one is received. The ZACK header contains the current file offset of the in-process file from the receivers perspective.

OMEN The ZCRCG allows the receiver to receive the subpackets the fastest with receiver responses of ZRPOS coming only if there is a subpacket transmission error.

OMEN A data subpacket terminated by **ZCRCG** and CRC does not elicit a response unless an error is detected; more data subpacket(s) follow immediately.

OMEN **ZCRCQ** data subpackets expect a **ZACK** response with the receiver's file offset if no error, otherwise a **ZRPOS** response with the last good file offset. Another data subpacket continues immediately. **ZCRCQ** subpackets are not used if the receiver does not indicate FDX ability with the **CANFDX** bit. (Note: Kientzle doesn't support this limitation, it will send regardless of the **CANFDX** bit.)

OMEN **ZCRCW** data subpackets expect a response before the next frame is sent. If the receiver does not indicate overlapped I/O capability with the CANOVIO bit, or sets a buffer size, the sender uses the **ZCRCW** to allow the receiver to write its buffer before sending more data.

OMEN A zero length data frame may be used as an idle subpacket to prevent the receiver from timing out in case data is not immediately available to the sender.

OMEN In the absence of a fatal error, the sender eventually encounters end of file. If the end of file is encountered within a frame, the frame is closed with a **ZCRCE** data subpacket which does not elicit a response except in case of error.

OMEN The sender sends a **ZEOF** header with the file ending offset equal to the number of characters in the file; as determined by the offset within the file at the time the sending program encountered end-of-file. The receiver compares this number with the number of characters received. If this comparison indicates the receiver has received all of the file, the receiver closes the file. If the file close was satisfactory, the receiver responds with **ZRINIT**. If the receiver has not received all the bytes of the file, the receiver ignores the **ZEOF** because a new **ZDATA** is coming. If the receiver cannot properly close the file, a **ZFERR** header is sent. (Note: Kientzle responds to improper byte count with a ZRPOS. Ed.)

OMEN After all files are processed, any further protocol errors should not prevent the sending program from returning with a success status.

OMEN The receiver [host device] compares the file position in the **ZDATA** header with the number of characters successfully received to the file. If they do not agree, a **ZRPOS** error response is generated to force the sender [cart device] start a new frame at the correct position within the file. If the **ZMSPARS** option is used, the receiver instead seeks to the position given in the **ZDATA** header.

4.3.3.2/1 The sender [cart device] shall send the final data packet from the file being transferred as a **ZCRCE** packet, which may contain 0 bytes of data, followed by a **ZEOF** header.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

Commentary: The proper responses to ZEOF are:

a) ZRINIT	everything's fine, next file please
b) ZRPOS	file position mismatch (recoverable)
c) ZFERR	file I/O (access/creation/closure) error (non-recoverable)

4.3.3.2/2 Upon receipt of a **ZEOF** as described in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file and compare the total number of file bytes received to that contained in the **ZEOF**. If the total bytes sent and received are the same, the receiver [host device] shall send a **ZRINIT** header containing receiver capability flags and receiver buffer limitations.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

4.3.3.2/3 Upon receipt of a **ZEOF** as described in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file and compare the total number of file bytes received to that contained in the **ZEOF**. If the total bytes sent and received are not the same the receiver [host device] shall send a **ZRPOS** header with the **ZPx** fields set to the first byte of data that the receiver wants retransmitted.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

4.3.3.2/4 Upon receipt of a **ZEOF** as described in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file. If this results in a file input/output error, the receiver [host device] shall send **ZCAN** (section 4.1.3.1, Cancel Sequence).
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

4.3.3.2/5 The receiver [host device] shall ignore session initiation sequence packets (see section 4.3.2.1, Transmit 'rz'<cr>, and ZRQINIT) received after a file-completion **ZEOF** packet and before a **ZRINIT** packet is sent by the receiver [host device].
STATE MACHINE REQUIREMENT

4.3.3.2/6 If a **ZFERR** header is received, the sender [cart device] shall send the cancel sequence.
STATE MACHINE REQUIREMENT.

Rationale: In the Physio implementation, there is no possible recovery mechanism from a receiver [host device] file-creation or access error. The **ZCRC** sequence is not used for the same reason.

Commentary: This will require modifying code. At present this would be considered an inappropriate header.

4.3.3.2/7 During the file transfer, the receiver [host device] shall respond to data packets as follows:

Packet	Normal Response	Abnormal Response	Note
ZCRCE	None	**ZRPOS**	
ZCRCG	None	**ZRPOS**	
ZCRCQ	**ZACK**	**ZRPOS**	- do not wait for **ZACK** to proceed
ZCRCW	**ZACK**	**ZRPOS**	- wait for **ZACK** before proceeding

An abnormal response is issued if there is a CRC error or the packet size is illegal (> 1024 bytes).

STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

Commentary: This does not apply to 4.3.3.1/2 "The receiver [host device] shall respond to the sender [cart device]'s file initiation sequence with a **ZRPOS** header indicating the offset in bytes from the first byte of the file at which transmission of data shall begin"

Commentary: The receiver [host device] responds to a properly formed **ZCRCQ** or **ZCRCW** packet with a **ZACK** packet containing the current file offset of the in-process file as calculated by the receiver [host device].

4.3.3.2/8 If, during the file transfer, the sender [cart device] receives a **ZRPOS**, then it shall begin sending data at the file offset specified in the **ZRPOS** header.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

4.3.3.2/9 The receiver [host device] shall respond to a properly formed **ZCOMMAND/ZCRCW** frame with a **ZACK** packet with **ZPx** = 0x00.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 1and 2.

Commentary: **ZPx**=0 is an exception to the general ZCRCW-ZACK relationship rule.

More information can be found in section 5.1, Entering ZTERM Mode.

4.3.4 Session Exit

4.3.4.1 Transmit ZFIN header 'The Shutdown Sequence'

OMEN The sender closes the session with a **ZFIN** header. The receiver acknowledges this with its own **ZFIN** header.

OMEN When the sender receives the acknowledging header, it sends two characters, "OO" (Over and Out) and exits to the operating system or application that invoked it. The receiver waits briefly for the "O" characters, then exits whether they were received or not.

4.3.4.1/1 The receiver [host device] shall respond to a **ZFIN** header with a **ZFIN** header and immediately terminate the session.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

4.3.4.1/2 When the sender [cart device] receives a **ZFIN** header in response to a **ZFIN** header, it shall send the string "OO" (ASCII upper-case 'O') and terminate the session without requiring additional responses. STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

Commentary: The requirement of the 'OO' (Over and Out) is kept because we may interface to a system some time in the future that may require this string.

4.4 FRAME TYPES

OMEN The numeric values for the values shown in boldface are given in zmodem.h. Unused bits and unused bytes in the header (ZP0...ZP3) are set to 0.

All ZMODEM frames begin with a header packet optionally followed by one or more data sub-packets. Physio implements fourteen ZMODEM header packets and four types of data sub-packets. **ZCAN**, counted as a header packet, is actually a pseudo-header returned in response to the cancel sequence.

4.4.1 ZRQINIT

OMEN Sent by the sending program, to trigger the receiving program to send its **ZRINIT** header. This avoids the aggravating start-up delay associated with XMODEM and Kermit transfers. The sending program may repeat the receive invitation (including **ZRQINIT**) if a response is not obtained at first.

4.4.1/1 ZF0 shall be set to **ZCOMMAND** if the program is attempting to send a command, 0 otherwise. NON-STATE MACHINE REQUIREMENT.

Type:	Parameters	Value:	Sent By:
ZRQINIT	Request Initialization packet	0x00	sender [cart device]
Format	**ZHEX**		
ZF0	ZCOMMAND (optional)	0x00 or 0x12	
ZF1	N/A	0x00	
ZF2	N/A	0x00	
ZF3	N/A	0x00	

4.4.2 ZRINIT

OMEN The **ZRINIT** must be sent by the receiving program.

Type:	Parameters	Value:	Sent By:
ZRINIT	Receiver Capabilities	0x01	receiver [host device]
Format	**ZHEX**		
ZF0		0x00	
ZF1		0x00	
ZP1		0x00	
ZP0		0x00	

OMEN ZF0 and ZF1 contain the bitwise OR of the receiver [host device] capability flags:

OMEN	CANFDX	0001 (0x01)	/* Rx can send and receive true FDX */
OMEN	CANOVIO	0002 (0x02)	/* Rx can receive data during disk I/O */
OMEN	CANBRK	0004 (0x04)	/* Rx can send a break signal */
OMEN	CANRLE	0010 (0x08)	/* Receiver can decode RLE */
OMEN	CANLZW	0020 (0x10)	/* Receiver can uncompress LZW */
OMEN	CANFC32	0040 (0x20)	/* Receiver can use 32 bit Frame Check */
OMEN	ESCCTL	0100 (0x40)	/* Receiver expects ctl chars to be escaped */
OMEN	ESC8	0200 (0x80)	/* Receiver expects 8th bit to be escaped */

OMEN ZP0 and ZP1 contain the size of the receiver's buffer in bytes, or 0 if nonstop I/O is allowed.

OMEN ZP0 and ZP1 contain the size of the receiver's buffer in bytes, or 0 if nonstop I/O is allowed.

Commentary: All of the flags defined above are contained in ZF0. No bits in ZF1 are currently defined in the **ZRINIT** header.

4.4.2/1 CANFC32 = 0 (16 bit CRCs) for Physio applications.

4.4.3 ZSINIT

OMEN The Sender sends flags followed by'a binary data subpacket terminated with **ZCRCW**.

OMEN The data subpacket contains the null terminated **Attn** sequence, maximum length 32 bytes including the terminating null.

Bit Masks for ZSINIT flags byte ZF0:

TESCCTL	0100 (0x40)	/* Transmitter expects ctl chars to be escaped */
TESC8	0200 (0x80)	/* Transmitter expects 8th bit to be escaped */

NOT IMPLEMENTED IN PHYSIO-CONTROL APPLICATIONS.

4.4.4 ZACK

OMEN Acknowledgment to a **ZSINIT** frame, **ZCHALLENGE** header, **ZCRCQ** or **ZCRCW** data subpacket. ZP0 to ZP3 contain file offset. The response to **ZCHALLENGE** contains the same 32 bit number received in the **ZCHALLENGE** header.

Type:	Parameters	Value:	Sent By:
ZACK	Acknowledge Packet	0x03	Both
Format	**ZHEX**		
ZP0		= File position for ZCRCQ/W, no meaning for ZTERM frames	
ZP1			
ZP2			
ZP3			

4.4.5 ZFILE

OMEN This frame denotes the beginning of a file transmission attempt. ZF0, ZF1, and ZF2 may contain options. **ZRWOVR**, if present, contain the sender's override rx window size/256. A value of 0 in each of these bytes implies no special treatment. Options specified to the receiver override options specified to the sender with the exception of **ZCBIN**. A **ZCBIN** from the sender overrides any other Conversion Option given to the receiver except **ZCRESUM**. A **ZCBIN** from the receiver overrides any other Conversion Option sent by the sender.

OMEN The following sections are an extract from the Omen document on ZMODEM.

4.4.5.1 ZF0: Conversion Option

OMEN If the receiver [host device] does not recognize the Conversion Option, an application dependent default conversion may apply.

OMEN **ZCBIN** "Binary" transfer - inhibit conversion unconditionally

OMEN **ZCNL** - Convert receiver [host device] end of line to local end of line convention. The supported end of line conventions are CR/LF (most ASCII based operating systems except Unix and Macintosh), and NL (Unix). Either of these two end of line conventions meet the permissible ASCII definitions for Carriage Return and Line Feed/New Line. Neither the ASCII code nor ZMODEM ZCNL encompass lines separated only by carriage returns. Other processing appropriate to ASCII text files and the local operating system may also be applied by the receiver [host device]. Filtering RUBOUT, NULL, Ctrl-Z, etc.

OMEN **ZCRECOV** - Recover/Resume interrupted file transfer. ZCREVOV is also useful for updating a remote copy of a file that grows without resending of old data. If the destination file exists and is no longer than the source, append to the destination file and start transfer at the offset corresponding to the receiver's end of file. This option does not apply if the source file is shorter. Files that have been converted (e.g., ZCNL) or subject to a single ended Transport Option cannot have their transfers recovered.

4.4.5.2 ZF1: Management Option

OMEN If the receiver [host device] does not recognize the Management Option, the file should be transferred normally.

OMEN The **ZMSKNOLOC** bit instructs the receiver [host device] to bypass the current file if the receiver [host device] does not have a file with the same name.

OMEN Five bits (defined by **ZMMASK**) define the following set of mutually exclusive management options.

OMEN **ZMNEWL** - Transfer file if destination file absent. Otherwise, transfer file overwriting destination if the source file is newer or longer.

OMEN **ZMCRC** - Compare the source and destination files. Transfer if file lengths or file polynomials differ.

OMEN **ZMAPND** - Append source file contents to the end of the existing destination file (if any).

OMEN **ZMCLOB** - Replace existing destination file (if any).

OMEN **ZMDIFF** - Transfer file if destination file absent. Otherwise, transfer file overwriting destination if files have different lengths or dates.

OMEN **ZMPROT** - Protect destination file by transferring file only if the destination file is absent.

OMEN **ZMNEW** - Transfer file if destination file absent. Otherwise, transfer file overwriting destination if the source file is newer.

OMEN ZF2: Transport Option

OMEN If the receiver [host device] does not implement the particular transport option, the file is copied without conversion for later processing.

OMEN **ZTLZW** - Lempel-Ziv compression. Transmitted data will be identical to that produced by **compress 4.0** operating on a computer with VAX byte ordering, using 12 bit encoding.

OMEN **ZTRLE** - Run Length encoding. The transmitter will use RLE encoded data subpackets. The **ZTRLE** bit is used as a display flag, the actual encoding choice is indicated by the frame type.

OMEN A **ZCRCW** data subpacket follows with file name, file length, modification date, and other information described in a later chapter.

4.4.5.3 ZF3: Extended Options

OMEN The Extended Options are bit encoded.

OMEN **ZTSPARS -** Special processing for sparse files, or sender [cart device] managed selective retransmission. Each file segment is transmitted as a separate frame, where the frames are not necessarily contiguous. The sender [cart device] should end each segment with a ZCRCW data subpacket and process the expected ZACK to insure no data is lost. ZTSPARS cannot be used with ZCNL.

4.4.5.4 ZFILE Requirements

4.4.5.4/1 A **ZFILE** header with the **ZCBIN** bit set sent from the sender [cart device] to the receiver [host device] shall override any other Conversion Option given to the receiver [host device] except **ZCRESUM.** NON-STATE MACHINE REQUIREMENT.

Rationale: Recursive sessions are not support by Physio-Control.

Type:	Parameters	Value:	Sent By:
ZFILE	Initiate a file transfer	0x04	Both
Format	**ZBIN**		
ZF0	ZCBIN - Use binary transfer mode. No data conversion	0x01	Default = 0x01
	ZCNL - Convert end-of-line characters as defined by the local default	0x02	
	ZCRECOV - Recover from an interrupted file transfer	0x03	
	Undefined - do not set	0x04 - 0xFF	
ZF1	ZMNEWL - Overwrite destination file is longer of newer.	0x01	Default = 0x04
	ZMCRC - Transfer only if source and destination files have different CRC's	0x02	
	ZMAPND - Append source file to destinations file	0x03	
	ZMCLOB - Unconditionally overwrite the destination file	0x04	
	ZMNEW - Overwrite only if source is newer	0x05	
	ZMDIFF - Overwrite only if source has different length or later date	0x06	
	ZMPROT - Only transfer files which do not exist	0x07	
	Undefined - do not set	0x08 - 0xFF	
ZF2	ZTLZW - File being sent has been UNIX compressed	0x01	Default = 0x00
	ZCRYPT - Not implemented	0x02	
	ZTRLE- RLE. File being transferred has been compressed using	0x03	
	Undefined - do not set	0x04 - 0xFF	
ZF3	Undefined - do not set	0x01 - 0xFF	Default = 0x00

ZFILE ZCRCW data sub-packet	All data items are in ASCII text or ASCII numeral representation. i.e., a short unsigned integer consists of one to three ASCII characters such as "248" or "42". A value of zero is represented as "0".		
	Filename - For PHYSIO-CONTROL this will be 9 characters in length encoded as follows: xxxxyyzz<null> - where: xxxx = a four-character field such that the complete filename is unique within the data transfer session yy = Device type: "00" = LP500 "01" = LP12 "02" = LP300 "03" = LP11 zz = the ASCII encoded report type per CICD extensions to SCP-ECG section 1	Mandatory	
	Length - Estimated file length in bytes,"0" if unknown.	Mandatory	
	ASCII Space	Mandatory	
	Date - Date and time in UNIX format, mm/dd/yy, "0" indicates unknown date and time.	Mandatory	
	ASCII Space	Mandatory	
	Mode - UNIX mode bits. "0" for Physio systems.	Mandatory	
	ASCII Space	Mandatory	
	S/N - Serial number of the transmitting program, "0" for Physio systems	Mandatory	
	ASCII Space	Mandatory	
	# of Files - Number of files left to transmit, "0" if unknown.	Mandatory	
	ASCII Space	Mandatory	
	# of Bytes - Estimated total number of bytes left to transmit.	Mandatory	
	<NULL>	Mandatory	

4.4.6 ZSKIP

OMEN Sent by the receiver in response to **ZFILE**, makes the sender skip to the next file.

Type:	Parameters	Value:	Sent By:
ZSKIP	Receiver will not accept the requested file	0x05	Receiver
Format	**ZHEX**		

NOT IMPLEMENTED IN PHYSIO-CONTROL APPLICATIONS.

4.4.7 ZNAK

OMEN Indicates last header was garbled. (See also **ZRPOS**).

4.4.7/1 The receiver [host device] shall send a **ZNAK** if and only if there is an inappropriate header before a **ZFILE** or after a **ZEOF**.

Commentary: A **ZNAK** is handled differently in ZTERM mode.

Type:	Parameters	Value:	Sent By:
ZNAK	Last header was in error	0x06	Both
Format	**ZHEX**		

4.4.8 ZABORT

OMEN Sent by receiver to terminate batch file transfers when requested by the user. Sender responds with a **ZFIN** sequence.[1]

4.4.8/1 To terminate a ZMODEM session, the receiver [host device] shall send a **ZABORT** header or the cancel sequence in section 4.1.3.1, Cancel Sequence.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 1 and 2.

Rationale: The **ZABORT** header is used by the receiver [host device] to terminate a session on request from a supervisory protocol layer, usually in response to a user request to halt the session. The cancel sequence is used in case of serious errors.

Commentary: Note that senders do NOT send the **ZABORT** header.

4.4.8/2 If a **ZABORT** header is received, the sender [cart device] shall send the cancel sequence in section 4.1.3.1, Cancel Sequence.
NON-STATE MACHINE REQUIREMENT.

Type:	Parameters	Value:	Sent By:
ZABORT	Abort the transfer	0x07	receiver [host device]
Format	**ZHEX**		

[1] Or **ZCOMPL** in case of server mode.

4.4.9 ZFIN

OMEN Sent by sending program to terminate a ZMODEM session. Receiver responds with its own **ZFIN**.

Type:	Parameters	Value:	Sent By:
ZFIN	No more files to transmit	0x08	Both
Format	**ZHEX**		

4.4.10 ZRPOS

OMEN Sent by the receiver to force file transfer to resume at file offset given in ZP0...ZP3. Some sending programs restrict acceptable ZRPOS offsets. Offsets of transmitted subpackets are expected to be acceptable.

Type:	Parameters	Value:	Sent By:
ZRPOS	Begin sending data from the specified offset	0x09	receiver [host device]
Format	**ZHEX**		
ZP0	Offset (Low order word/Byte)		
ZP1	Offset (Low order word/High order Byte)		
ZP2	Offset (High order word / low order byte)		
ZP3	Offset (high order word / byte)		

4.4.11 ZDATA

OMEN Zdata uses ZP0...ZP3 to contain file offset.

Type:	Parameters	Value:	Sent By:
ZDATA	A data follows starting at the specified file offset	0x0A	Both
Format	**ZBIN**		
ZP0	Offset (Low order word/Byte)		
ZP1	Offset (Low order word/High order Byte)		
ZP2	Offset (High order word / low order byte)		
ZP3	Offset (high order word / byte)		

4.4.12 ZEOF

OMEN ZEOF is used by the sender to report the end of the File.

4.4.12/1 ZP0...ZP3 contain the total number of bytes sent.
NON-STATE MACHINE REQUIREMENT.

Type:	Parameters	Value:	Sent By:
ZEOF	All file data has been sent. EOF is located at the specified offset	0x0B	sender [cart device]
Format	**ZHEX**		
ZP0	Offset (Low order word/Byte)		
ZP1	Offset (Low order word/High order Byte)		
ZP2	Offset (High order word / low order byte)		
ZP3	Offset (high order word / byte)		

4.4.13 ZFERR

OMEN After receiving a ZFILE frame, but before the receiver sends ZRPOS, ZFERR indicates the output file cannot be opened because of an error condition (write protected, illegal pathname, etc.).

OMEN Otherwise, indicates an error in reading or writing file, protocol equivalent to ZABORT.

Type:	Parameters	Value:	Sent By:
ZFERR	A read/write error has occurred. Terminates session	0x0C	receiver [host device]
Format	**ZHEX**		

4.4.14 ZCRC

OMEN Request (receiver) and response (sender) for file polynomial. ZP0...ZP3 contain number of bytes (receiver) or file polynomial (sender). Offset of segment to checksum starts at byte 6 of header, or 0 (Available with variable length headers.) Number of bytes and offset must be multiples of 1024.

NOT IMPLEMENTED IN PHYSIO-CONTROL APPLICATIONS.

4.4.15 ZCHALLENGE

OMEN Request sender to echo a random number in ZP0...ZP3 in a ZACK frame. Sent by the receiving program to the sending program to verify that it is connected to an operating program, and was not activated by spurious data or a Trojan Horse message.

NOT IMPLEMENTED IN PHYSIO-CONTROL APPLICATIONS.

4.4.16 ZCOMPL

OMEN ZCOMPL indicates that the request is now complete.

Commentary: **ZCOMPL** is only sent to exit from ZTERM mode and is not used in our implementation as a response to any other command.

Type:	Parameters	Value:	Sent By:
ZCOMPL	Response to ZCOMMAND, indicating the command is complete. ZP0-3 contain the command status	0x0F	receiver [host device]
Format	**ZHEX**		

4.4.17 ZCAN

OMEN This is a pseudo frame type returned in response to a Session Abort sequence.

Type:	Parameters	Value:	Sent By:
Undefined	Undefined	0x10	N/A

4.4.18 ZCOMMAND

OMEN ZCOMMAND is sent in a binary frame. **ZF0** contains **0** or **ZCACK1** (see below).

OMEN A ZCRCW data subpacket follows, with the ASCII text command string terminated with a NULL character. If the command is intended to be executed by the operating system hosting the receiving program (e.g., "shell escape"), it must have "!" as the first character. Otherwise the command is meant to be executed by the application program which receives the command.

OMEN If the receiver detects an illegal or badly formed command, the receiver immediately responds with a ZCOMPL header with an error code [TBD] in ZP0...ZP3.

OMEN If ZF0 contained **ZCACK1,** the receiver immediately responds with a ZCOMPL header with 0 status.

OMEN Otherwise, the receiver responds with a ZCOMPL header when the operation is completed. The exit status of the completed command is stored in ZP0...ZP3. A 0 exit status implies nominal completion of the command.

OMEN Otherwise, the receiver responds with a ZCOMPL header when the operation is completed. The exit status of the completed command is stored in ZP0...ZP3. A 0 exit status implies nominal completion of the command.

OMEN If the command causes a file to be transmitted, the command sender will see a ZRQINIT frame from the other computer attempting to send data.

OMEN The sender examines ZF0 of the received ZRQINIT header to verify it is not an echo of its own ZRQINIT header. It is illegal for the sending program to command the receiving program to send a command.

OMEN If the receiver program does not implement command downloading, it may display the command to the standard error output, then return a ZCOMPL header.

Type:	Parameters	Value:	Sent By:
ZCOMMAND	Used to send a command to either an application program or command shell, contained within a ZCRCW data subpacket. For use at Physio-Control, this is used to enter ZTERM mode as described in section 5, Physio-Control ZTERM Mode. The immediate response is a ZACK.	0x12	sender [cart device]
Format	**ZBIN**		

4.5 ZMODEM CRC Definitions

The 16-Bit and 32-Bit ZMODEM CRC's have different calculations and transmission requirements. Both the 16-bit and 32-Bit CRC accumulate over the same data, but differently for header packets and data packets.

CRC's for header packets accumulates from the packet type to the end of the parameter bytes.

CRC of a header packet

Start of Frame			Format	TYPE	Parameters				CRC	
ZPAD	ZPAD	ZDLE	**ZHEX**	ZRQINIT	ZF3	ZF2	ZF1	ZF0	High	Low
				CRC accumulates these bytes						

CRC's for data sub-packets accumulates from the start of data to the end of the frame, not including the initial **ZDLE** of any link escape sequence:

CRC of a data sub-packet

Data	Frame-End		CRC	
0-1024 Bytes	ZDLE	Frame Type	High	Low
CRC accumulates data and frame bytes omitting **ZDLE**.				

4.5.1 ZMODEM 16-Bit CRC polynomial

4.5.1/1 The CRCs must be calculated according to this section.

The ZMODEM 16-BIT CRC is similar to the CCITT CRC-16 in that is based on the polynomial 0x1021 ($X^{16} + X^{12} + X^5 + 1$). However there are two specific differences:

1) The remainder is initialized to 0x0000 instead of 0xFFFF

2) The CRC calculations is made using a look-up table which is generated from the CCITT CRC function. The table is calculated as follows:

Each table index (0-255) represents the message polynomial

The entry at each table index is calculated as follows:

The remainder the modulo 2 division of:

$$\frac{x^{16} \times \langle Message - polynomial \rangle}{\langle x^{16} + x^{12} + x^{5} + 1 \rangle}$$

This calculation is repeated for each index (0-255) and the 16-bit result is stored in the table at the location of the index.

The CCITT-CRC is then calculated using a look-up and XOR function as follows:

accumulator = CRC_table[((accumulator >> 8) & 0xFF) ^ current_byte] ^ (accumulator << 8)

Although the tables are identical, the ZMODEM look-up function is not. The ZMODEM look-up function is as follows:

```
accumulator = CRC_table[((accumulator >> 8) & 0xFF)] ^ current_byte ^ (accumulator << 8)
```

This results in a different CRC value from the CCITT CRC-16

The ZMODEM CRC-16 is transmitted High Byte followed by Low Byte.

4.5.2 ZMODEM 32-Bit CRC polynomial

This algorithm (assumes modulo 2 math, Remainder initialized to -1.

Transmitted low word followed by the high word

The 32-Bit CRC is calculated as the ones compliment of the modulo 2 sum of:

The remainder of the modulo 2 division of:

$$\frac{x^{n} \times \langle x^{31} + x^{30} + x^{29} + \ldots x^{3} + x^{2} + x^{1} + 1\rangle}{\langle x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1\rangle}$$

n= number of bits in CRC area

and the remainder the modulo 2 division of:

$$\frac{x^{32} \times \langle \text{content of the data frame} \rangle}{\langle x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1\rangle}$$

5 Physio-Control ZTERM Mode

The standard file transfer protocol ZMODEM does not contain the ability to send bi-directional commands, such as those required by the SCP-ECG command shell. This prevents command shells from running under ZMODEM. To run command shells, Physio uses a special set of restrictions to the ZMODEM protocol syntax rules, called ZTERM mode (ZMODEM terminal emulation mode).

ZTERM is designed to handle bidirectional commands from a shell program/protocol. A shell manager (see section 2.3) acts as the supervisory process for the shell and communicates with the ZMODEM session manager.

To send a shell command, the shell manager requests that the ZMODEM session manager package the shell command in a ZTERM frame and send it. This is called a "shell transmission". Correspondingly, if a ZTERM frame is received in ZTERM mode, the ZMODEM session manager notifies the shell manager that a shell command string is available. This is called a "shell reception".

Once the shell manager determines that operations are complete, it notifies the session manager which cause a ZCOMPL(good) message to be sent and ZTERM mode is exited. This is noted as "shell operations complete" in the state diagrams.

5.1 Entering ZTERM Mode

The entry into the ZTERM mode is a multi-step process that requires two preparatory steps:

1) Establish a connection between the sending and receiving devices.

2) Initiate a ZMODEM session as described in section 4.3.2, Session Start-up, of this document.

ZTERM is entered by the sender [cart device] sending a **ZCOMMAND/ZCRCW** frame containing the string, "{phy::term}<CR>".

If the receiver [host device] does not accept the ZTERM initiation request, responding with any other header than **ZACK**, the sender [cart device] reverts to ordinary ZMODEM operation.

A **ZACK** response from the receiver [host device] accepts the ZTERM initiation, and causes all further exchanges between the devices to be controlled by the rules described in this section until the receiver [host device] sends a **ZCOMPL** header.

It is acceptable to both request entry to ZTERM mode and send a shell command within a single **ZCOMMAND/ZCRCW** frame. The receiver [host device] will follow the ZTERM entry acknowledgment with whatever response is required by the shell command, packaged in ZTERM frames.

5.1/1 Following the initial negotiation of the ZMODEM session in section 4.3.2, the sending device shall issue a **ZCOMMAND/ZCRCW** frame containing the following string: "{phy::term}<CR>" + Optional shell command string.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 1 and 2.

Commentary: The lack of a trailing null is a deviation from the Omen specification to match the LP500 implementation.

Commentary: This is the ZTERM mode entry request message.

5.1/2 If the response to a **ZRINIT** is not a **ZCOMMAND**, then the receiver [host device] shall resend the **ZRINIT**.

5.1/3 A Physio receiver [host device] shall recognize the string in requirement 5.1/1 and shall enter ZTERM mode while responding with a **ZACK**.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 1 and 2.

5.1/4 If the receiver [host device] does not recognize the string in requirement 5.1/1, it shall respond with a **ZCOMPL (BAD)**, with ZFx = 0xFF and fail to enter ZTERM mode.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 1.

5.1/5 If a **ZCOMPL** or **ZCAN** (section 4.1.3.1, Cancel Sequence) is received by the sender [cart device] in place of the **ZACK** in response to the ZTERM entry request message as defined in requirement 5.1/1, the sender [cart device] shall abort the session.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 2.

Commentary: Warning, a non-Physio device that does not support ZTERM or command shell transactions may respond with a **ZCOMPL** or **ZCAN** (section 4.1.3.1, Cancel Sequence) instead of a **ZACK**, thus this requirement.

5.2 ZTERM Valid Headers

5.2/1 Within ZTERM mode, the only valid ZMODEM headers are **ZDATA, ZACK, ZNAK, ZABORT,** and **ZCOMPL**. Any other ZMODEM headers shall be ignored by the device receiving them.
NON-STATE MACHINE REQUIREMENT.

Commentary: "Ignored" means "discarded without error response (ZNAK)".

Commentary: Note that **ZABORT** may be sent by either device under ZTERM mode rules.

5.2.1 ZACK

5.2.1/1 **ZACK** is sent as a response to a valid ZTERM frame without errors.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 3 and 4.

5.2.2 ZNAK

5.2.2/1 **ZNAK** is sent as a response to a header or data packet with errors.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 3 and 4.

5.2.2/2 When a **ZNAK** is received as a response to a header or data packet, the software will retransmit the header or frame that caused the **ZNAK**.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 3 and 4.

5.2.3 ZABORT

5.2.3/1 **ZABORT** shall cause an exit of ZTERM mode and abort the ZMODEM session.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 3 and 4.

5.2.3/2 The software shall transmit a **ZABORT** when???.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 3 and 4.

ACTION Incomplete

5.2.4 ZCOMPL()

5.2.4/1 A **ZCOMPL** header is sent as a reply to a **ZCOMMAND**.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagrams 3 and 4.

Commentary: The ZMODEM defined purpose for the **ZCOMPL** header is to return a status complete message to a **ZCOMMAND.** Note that within Physio ZTERM mode, however, the return of the **ZCOMPL** header is used to indicate the successful execution of a shell command sequence and that ZTERM mode is to be exited.

5.3 Exchange of Commands

Once the ZTERM state has been entered, commands are exchanged by using the ZTERM frames defined below.

5.3.1 The ZCOMMAND/ZCRCW frame:

A **ZCRCW** data sub-packet following a **ZCOMMAND** - Example command shell entry:

Start of Frame			Format	TYPE	Parameters				CRC	
ZPAD	ZPAD	ZDLE	ZBIN	ZCOMMAND	0x00	0x00	0x00	0x00	High	Low

The Data sub-packet

Data	Frame-End		CRC	
"{phy::term}<CR>" + Optional command string	ZDLE	ZCRCW	High	Low

5.3.2 The structure of the ZTERM frame:

A ZTERM frame consists of a **ZCRCW** data sub-packet following a **ZDATA** header

Start of Frame			Format	TYPE	Parameters				CRC	
ZPAD	ZPAD	ZDLE	ZBIN	ZDATA	0	0	0	0	High	Low

The ZTERM Data sub-packet

Data	Frame-End		CRC	
0-1024 bytes of information	ZDLE	ZCRCW	High	Low

5.3.2/1 An **XON** character received at the end of the ZTERM frame shall be ignored.

5.4 Termination of ZTERM mode

5.4/1 To end a ZTERM session, the receiver [host device] shall send a **ZCOMPL** header.
STATE MACHINE REQUIREMENT, see ZMODEM State Diagram 4.

Commentary: In general, the labels are stimulus received from the other process. The exception is ZTERM mode in the receiver [host device], there is no outside stimulus to exit ZTERM, it exits of its own accord.

5.5 ZTERM Packet Acknowledge and Time-outs

5.5/1 Within ZTERM mode, retransmit and time-out rules shall be the same as for regular ZMODEM.

Rationale: Note that a retry failure or time-out will cause both ZTERM mode and ZMODEM to terminate.

6 HEADER and OTHER FILES

The following sections are copies of various source files pertinent to the documentation of ZMODEM.

6.1 ZMODEM.H - Manifest constants for ZMODEM file transfer protocol

```
/ *    Copyright 1993 Omen Technology Inc. All Rights Reserved
*/
#define ZPAD '*'              /* 052 Padding character begins frames */
#define ZDLE 030              /* Ctrl-X Zmodem escape - `ala BISYNC DLE */
#define ZDLEE (ZDLE^0100)     /* Escaped ZDLE as transmitted */
#define ZBIN 'A'              /* Binary frame indicator (CRC-16) */
#define ZHEX 'B'              /* HEX frame indicator */
#define ZBIN32 'C'            /* Binary frame with 32 bit FCS */
#define ZBINR32 'D'           /* RLE packed Binary frame with 32 bit FCS */
#define ZVBIN 'a'             /* Binary frame indicator (CRC-16) */
#define ZVHEX 'b'             /* HEX frame indicator */
#define ZVBIN32 'c'           /* Binary frame with 32 bit FCS */
#define ZVBINR32 'd'          /* RLE packed Binary frame with 32 bit FCS */
#define ZRESC 0176            /* RLE flag/escape character */
#define ZMAXHLEN 16           /* Max header information length  NEVER CHANGE */
#define ZMAXSPLEN 1024        /* Max subpacket length  NEVER CHANGE */

/* Frame types (see array "frametypes" in zm.c) */
#define ZRQINIT 0             /* Request receive init */
#define ZRINIT 1              /* Receive init */
#define ZSINIT 2              /* Send init sequence (optional) */
#define ZACK 3                /* ACK to above */
#define ZFILE 4               /* File name from sender */
#define ZSKIP 5               /* To sender: skip this file */
#define ZNAK 6                /* Last packet was garbled */
#define ZABORT 7              /* Abort batch transfers */
#define ZFIN 8                /* Finish session */
#define ZRPOS 9               /* Resume data trans at this position */
#define ZDATA 10              /* Data packet(s) follow */
#define ZEOF 11               /* End of file */
#define ZFERR 12              /* Fatal Read or Write error Detected */
#define ZCRC 13               /* Request for file CRC and response */
#define ZCHALLENGE 14         /* Receiver's Challenge */
#define ZCOMPL 15             /* Request is complete */
#define ZCAN 16               /* Other end canned session with CAN*5 */
#define ZFREECNT 17           /* Request for free bytes on filesystem */
#define ZCOMMAND 18           /* Command from sending program */
#define ZSTDERR 19            /* Output to standard error, data follows */

/* ZDLE sequences */
#define ZCRCE 'h'             /* CRC next, frame ends, header packet follows */
#define ZCRCG 'i'             /* CRC next, frame continues nonstop */
#define ZCRCQ 'j'             /* CRC next, frame continues, ZACK expected */
#define ZCRCW 'k'             /* CRC next, ZACK expected, end of frame */
#define ZRUB0 'l'             /* Translate to rubout 0177 */
#define ZRUB1 'm'             /* Translate to rubout 0377 */

/* zdlread return values (internal) */
/* -1 is general error, -2 is timeout */
#define GOTOR 0400
#define GOTCRCE (ZCRCE|GOTOR)/* ZDLE-ZCRCE received */
#define GOTCRCG (ZCRCG|GOTOR)/* ZDLE-ZCRCG received */
#define GOTCRCQ (ZCRCQ|GOTOR)/* ZDLE-ZCRCQ received */
#define GOTCRCW (ZCRCW|GOTOR)/* ZDLE-ZCRCW received */
#define GOTCAN (GOTOR|030)          /* CAN*5 seen */
```

```
/* Byte positions within header array */
#define ZF0             3      /* First flags byte */
#define ZF1             2
#define ZF2             1
#define ZF3             0
#define ZP0             0      /* Low order 8 bits of position */
#define ZP1             1
#define ZP2             2
#define ZP3             3      /* High order 8 bits of file position */

/* Bit Masks for ZRINIT flags byte ZF0 */
#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 CANRLE          010    /* Receiver can decode RLE */
#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 */

/* Bit Masks for ZRINIT flags byte ZF1 */
#define CANVHDR         01     /* Variable headers OK */

/* Parameters for ZSINIT frame */
#define ZATTNLEN 32          /* Max length of attention string */
#define ALTCOFF ZF1          /* Offset to alternate canit string, 0 if not used */
/* 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 */

/* Parameters for ZFILE frame */
/* Conversion options one of these in ZF0 */
#define ZCBIN           1      /* Binary transfer - inhibit conversion */
#define ZCNL            2      /* Convert NL to local end of line convention */
#define ZCRESUM         3      /* Resume interrupted file transfer */
/* Management include options, one of these ored in ZF1 */
#define ZMSKNOLOC       0200   /* Skip file if not present at rx */
/* Management options, one of these ored in ZF1 */
#define ZMMASK          037    /* Mask for the choices below */
#define ZMNEWL          1      /* Transfer if source newer or longer */
#define ZMCRC           2      /* Transfer if different file CRC or length */
#define ZMAPND          3      /* Append contents to existing file (if any) */
#define ZMCLOB          4      /* Replace existing file */
#define ZMNEW           5      /* Transfer if source newer */
                        /* Number 5 is alive ... */
#define ZMDIFF          6      /* Transfer if dates or lengths different */
#define ZMPROT          7      /* Protect destination file */
#define ZMCHNG          8      /* Change filename if destination exists */
/* Transport options, one of these in ZF2 */
#define ZTLZW           1      /* Lempel-Ziv compression */
#define ZTRLE           3      /* Run Length encoding */
/* Extended options for ZF3, bit encoded */
#define ZXSPARS         64     /* Encoding for sparse file operations */
#define ZCANVHDR        01     /* Variable headers OK */
/* Receiver window size override */
#define ZRWOVR 4             /* byte position for receive window override/256 */

/* Parameters for ZCOMMAND frame ZF0 (otherwise 0) */
#define ZCACK1          1      /* Acknowledge, then do command */
```

```
long rclhdr();

/* Globals used by ZMODEM functions */
extern Rxframeind;          /* ZBIN ZBIN32, or ZHEX type of frame */
extern Rxtype;              /* Type of header received */
extern Rxcount;             /* Count of data bytes received */
extern Rxtimeout;           /* Tenths of seconds to wait for something */
extern long Rxpos;          /* Received file position */
extern long Txpos;          /* Transmitted file position */
extern Txfcs32;             /* TURE means send binary frames with 32 bit FCS */
extern Crc32t;              /* Display flag indicating 32 bit CRC being sent */
extern Crc32;               /* Display flag indicating 32 bit CRC being received */
extern Znulls;              /* Number of nulls to send at beginning of ZDATA hdr */
extern char Attn[ZATTNLEN+1]:/* Attention string rx sends to tx on err */
extern char *Altcan;        /* Alternate canit string */

/* End of zmodem.h */
```

7 TESTS

UNDER CONSTRUCTION

Section	Requirement		Test Script Names
4.2.1, Link Escape Encoding	1	When a **ZDLE** character appears in a binary data field, it shall be prefixed with a **ZDLE** character, converting it to the **ZDLEE** character string.	TBD
	2	Receiving 5 consecutive **ZDLE** characters (equivalent to the ASCII **CAN** character) shall cause the current ZMODEM session to be aborted.	term1.was,term2.was,term3.was
	3	A character with bit 6 set and bit 5 reset, following a **ZDLE**, shall be converted by the receiver [host device] to a single character with bit 6 reset and all other bits unchanged.	TBD
	4	If link-escaped, the octal characters: 0020 (0x10) and 0220 (0x90), 0021 (0x11) and 0221 (0x91), 0023 (0x13) and 0223 (ox93), 0177 (0x7f) and 0377 (oxff), shall be converted on receipt to a single character with bit 6 reset and all other bits unchanged.	TBD
	5	The octal (hex) characters 0015 (0x0d) and 0215 (0x4d), the ASCII **CR,** shall be link-escaped if they are preceded by either a 0100 (0x40) or 0300 (oxc0), the ASCII @ character.	TBD
	6	The option to link-escape all ASCII control characters (those characters with bits 5 and 6 reset) shall be implemented in both transmit and receive modes.	TBD
4.2.2, Header	1	All ZMODEM headers shall contain 2 **ZPAD**s, a **ZDLE**, a format byte, a type byte, 4 parameter bytes, and a CRC as shown in table 1 below	zmdm.was
	2	One or two extra **ZPAD** characters received at the start of a header shall be ignored.	TBD
4.2.2.3, Hex Header	1	When parsing a hex header, the receiver [host device] shall ignore the state of the parity bit.	TBD
	2	For the type byte, the four position/flag bytes, and the 16 bit CRC, which are sent in hex, only the following character set shall be used: 0123456789abcdef.	zmdm.was

Section	Requirement		Test Script Names
4.3.1, Recommended Time-Outs	1	A header response timer shall measure the time between the completion of the transmission of the final byte of a header packet and the receipt of the first byte of a response.	TBD
	2	If the header response timer expires before a required response to a valid and appropriate header is received, the header shall be retransmitted; except during file transfer.	retry1.was,retry2.was
	3	If the header response timer expires following the tenth consecutive retransmission of a header, the device attempting to send the header shall be aborted by the **CAN**, section 4.1.3.1, Cancel Sequence.	TBD
	4	If during a file transfer, the data line is quiescent for >= 10 seconds, the receiver shall attempt to restart the transfer using the ZRPOS header.	TBD
4.3.2.1, Transmit 'rz'<cr>, and ZRQINIT	1	To initiate the first ZMODEM session, the transmitting device shall send 'rz' (ASCII characters) followed by an ASCII **CR**, followed by a **ZRQINIT** header.	zmdm.was
	2	Following receipt of the ZMODEM initiation sequence, the receiver [host device] will respond with a **ZRINIT** header.	TBD
	3	The receiver's retry mechanics for establishing the ZMODEM session shall be the same as defined in section 4.3.1, Recommended Time-Outs.	TBD
4.3.3.1, Transmit ZFILE Header and Data Subpackets	1	To initiate a file transfer after the end of the ZTERM session described in section 5, Physio-Control ZTERM Mode, the sender [cart device] shall send a **ZFILE** header followed by a **ZCRCW** data packet. This is defined as the file initiation sequence.	TBD
	2	The receiver [host device] shall respond to the transmitter's file initiation **ZFILE** and data packets with a **ZRPOS** header indicating the offset in bytes from the first byte of the file at which transmission of data shall begin.	TBD
	3	After the receipt of a valid file initiation sequence, the receiver [host device] shall cause to be created a destination file to receive the sender [cart device] data as defined in the file initiation **ZFILE** and data packets.	TBD
	4	If the receiver [host device] cannot create a destination file for the sender [cart device] data as described in 4.3.3.1/3, then it shall send a **ZFERR** header.	TBD
4.3.3.2, Transmit ZDATA header and Data Subpackets	1	The sender [cart device] shall send the final data packet from the file being transmitted as a **ZCRCE** packet, which may be 0 bytes, followed by a **ZEOF** header.	zmdm.was

Section	Requirement	Test Script Names
	2 Upon receipt of the **ZEOF** in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file and compare the total number of file bytes received to that contained in the **ZEOF**. If the total bytes sent and received are the same, the receiver [host device] shall send a **ZRINIT** header containing receiver capability flags and receiver buffer limitations.	TBD
	3 Upon receipt of the **ZEOF** in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file and compare the total number of file bytes received to that contained in the **ZEOF**. If the total bytes sent and received are not the same the receiver [host device] shall send a **ZRPOS** header up to the retry limit.	TBD
	4 Upon receipt of a **ZEOF** as described in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file. If this results in a file input/output error, the receiver [host device] shall send a **CAN** (section 4.1.3.1, Cancel Sequence).	TBD
	5 The receiver [host device] shall ignore session initiation sequence packets (see section 4.3.2.1, Transmit 'rz'<cr>, and ZRQINIT) received after a file-completion **ZEOF** packet and before a **ZRINIT** packet is sent by the receiver [host device].	TBD
	6 If a **ZFERR** header is received, the sender [cart device] shall send the cancel sequence.	zskip.was
	7 During the file transfer, the receiver [host device] shall respond to data packets as follows: Packet Normal ResponseAbnormal ResponseNote **ZCRCE** None **ZRPOS** **ZCRCG** None **ZRPOS** **ZCRCQ** **ZACK** **ZRPOS-** do not wait for **ZACK** to proceed **ZCRCW** **ZACK** **ZRPOS-** wait for **ZACK** before proceeding.	TBD
	8 If, during the file transfer, the sender [cart device] receives a **ZRPOS**, then it shall begin sending data at the file offset specified in the **ZRPOS** header.	TBD
	9 The receiver [host device] shall respond to a properly formed **ZCOMMAND/ZCRCW** frame with a **ZACK** packet with **ZPx** = 0x00.	TBD

Section	Requirement		Test Script Names
4.3.4.1, Transmit ZFIN header 'The Shutdown Sequence'	1	The receiver [host device] shall respond to a **ZFIN** header with a **ZFIN** header and immediately terminate the session.	TBD
	2	When the sender [cart device] receives a **ZFIN** header in response to a **ZFIN** header, it shall send the string "OO" (ASCII upper-case 'O') and terminate the session without requiring additional responses.	zmdm.was
4.4.1, ZRQINIT	1	ZF0 shall contain **ZCOMMAND** if the program is attempting to send a command, 0 otherwise.	TBD
4.4.2, ZRINIT	1	CANFC32 = 0 (16 bit CRCs) for Physio applications.	TBD
4.4.5.4, ZFILE Requirements	1	A **ZFILE** header with the **ZCBIN** bit set sent from the sender [cart device] to the receiver [host device] shall override any other Conversion Option given to the receiver [host device] except **ZCRESUM.**	TBD
4.4.7, ZNAK	1	The sender [cart device] shall send a **ZNAK** if and only if there is an inappropriate header before a **ZFILE** or after a **ZEOF.**	TBD
4.4.8, ZABORT	1	To terminate a ZMODEM session, the receiver [host device] shall send a **ZABORT** header or the cancel sequence in section 4.1.3.1, Cancel Sequence.	TBD
	2	If a **ZABORT** header is received, the sender [cart device] shall send the cancel sequence in section 4.1.3.1, Cancel Sequence.	term4.was
4.4.12, ZEOF	1	ZP0...ZP3 contain the total number of bytes sent.	TBD
4.5.1, ZMODEM 16-Bit CRC polynomial	1	The CRCs must be calculated according to this section.	TBD
5.1, Entering ZTERM Mode	1	Following the initial negotiation of the ZMODEM session in section 4.3.2, the sending device shall issue a **ZCOMMAND** packet with the following string: "{phy::term}<CR>" + Optional shell command string.	TBD
	2	If the response to a **ZRINIT** is not a **ZCOMMAND**, then the receiver [host device] shall resend the **ZRINIT.**	
	3	A Physio-Control receiver [host device] shall recognize the string in requirement 5.1/1 and shall enter ZTERM mode after responding with a **ZACK**.	TBD
	4	If the receiver [host device] does not recognize the string in requirement 5.1/1, it shall respond with a **ZCOMPL (BAD)**, with ZFx = 0xFF and fail to enter ZTERM mode.	TBD
	5	If a **ZCOMPL** or **ZCAN** (section 4.1.3.1, Cancel Sequence) is received by the sender [cart device] in place of the **ZACK** in response to the ZTERM entry request message as defined in requirement 5.1/1, the sender [cart device] shall abort the session.	TBD

Section	Requirement		Test Script Names
5.2, ZTERM Valid Headers	1	Within ZTERM mode, the only valid ZMODEM headers are **ZDATA, ZACK, ZNAK, ZABORT,** and **ZCOMPL**. Any other ZMODEM headers shall be ignored.	TBD
5.2.1, ZACK	1	**ZACK** is sent as a response to a valid ZTERM frame without errors.	TBD
5.2.2, ZNAK	1	**ZNAK** is sent as a response to a header or data packet with errors.	TBD
	2	When a **ZNAK** is received as a response to a header or data packet, the software will retransmit the header or frame that caused the **ZNAK.**	
5.2.3, ZABORT	1	**ZABORT** shall cause a termination of BOTH ZTERM mode and ZMODEM sessions.	TBD
	2	The software shall transmit a **ZABORT** when ???.	
5.2.4, ZCOMPL()	1	A **ZCOMPL** header is sent as a reply to a **ZCOMMAND.**	
5.3.2, The structure of the ZTERM frame:	2	An **XON** character received at the end of the ZTERM frame shall be ignored.	
5.4, Termination of ZTERM mode	1	To end a ZTERM session, the receiver [host device] shall send a **ZCOMPL** header.	TBD
5.5, ZTERM Packet Acknowledge and Time-outs	1	Within ZTERM mode, retransmit and time-out rules are the same as for regular ZMODEM.	TBD

A Implementation Sections

UNDER CONSTRUCTION

4.2.2/8 Conflicts with PCC implementation - 2 ZPAD characters are used in accordance with common industry practice.

Our existing documentation does not show the ZDLE characters preceding the frame-type, ZF/ZP, and CRC bytes. This is confusing and should be explicit.

Two ZPAD characters may be used at the beginning of both HEX and BINARY headers, conforming to common industry practice.

The Physio implementation of ZMODEM will not generate XON characters in any packet except HEX headers.

A.1 Legacy

A.1.1 Legacy Exceptions and Extensions

A.1.2 Legacy Design Description

A.1.3 Legacy Test Implementations

Including modifications to the standard test scripts.

A.2 LP500

ZRINIT sent by the DM unit should be set to:

CANFDX = 1
CANOVIO = 1
CANBRK = ?
CANCRY = ?
CANLZW = 0
CANFC32 = 0
ESCCTL = 0
ESC8 = 0

and ZF1 = 0.

ZSINIT sent by DT1/DT2:

TESCCTL = 1
TESCE8 = 0

and ZF1 = ZF2 = ZF3 = 0.

ZFILE sent by DT1/DT2:

```
ZCBIN = 1
ZCNL = 0
ZCRECOV = 0
ZMNEWL = don't care
ZMCRC = don't care
ZMAPND = 0
ZMCLOB = 1
ZMNEW = 0
ZMDIFF = 0
ZMPROT = 0
ZTLZW = 0
ZTRLE = 0
```

A.2.1 LP500 Exceptions and Extensions

From Drew, Kevin to: Greisen, Marc; White, Gus; wsilver@mailhost; Eric Schnellman; Greg Robinson; October 14, 1997 3:43PM

... the LP500 is setting the ZFILE flags with the following values.

```
#if 0
#define ZM_ZFILE_ZF0 0x01      /* Binary transfer mode. No data conversion */
#define ZM_ZFILE_ZF1 0x04      /* Unconditionally overwrite destination file */
#endif
#define ZM_ZFILE_ZF0 0x03      /* Binary transfer mode. No data conversion */
#define ZM_ZFILE_ZF1 0x02      /* Unconditionally overwrite destination file */
```

These constants were picked directly from the code and the comments on the second set (the one that is used) don't line up with the comments in the CICD/ZMODEM spec. Does this concern anyone?

The LP500, software versions 4.2 and earlier, sets the ZFILE bytes ZF0 and ZF1 to 0x02 and 0x03, respectively. This is an error related to debug values that were left in the production version.

A.2.2 LP500 Design Description

A.2.3 LP500 Test Implementations

Including modifications to the standard test scripts.

A.3 LP12

Doesn't implement Omen's or Keintzle's processing of **CANFDX.**

The first hex header, ZRQINIT, is used to identify the LP12 from other Physio-Control devices.

A.3.1 LP12 Exceptions and Extensions

A.3.2 LP12 Design Description

A.3.3 LP12 Test Implementations

Including modifications to the standard test scripts.

A.4 Grapevine

The string "ATEV01" is used to identify the LP500 from other Physio-Control devices.

A.4.1 Grapevine Exceptions and Extensions

A.4.2 Grapevine Design Description

A.4.3 Grapevine Test Implementations

Including modifications to the standard test scripts.

A.5 Trace Matrix

Section	Requirement		LP12	Grape-vine
4.2.1, Link Escape Encoding	1	When a **ZDLE** character appears in a binary data field, it shall be prefixed with a **ZDLE** character, converting it to the **ZDLEE** character string.	Y	Y
	2	Receiving 5 consecutive **ZDLE** characters (equivalent to the ASCII **CAN** character) shall cause the current ZMODEM session to be aborted.	Y	Y
	3	A character with bit 6 set and bit 5 reset, following a **ZDLE**, shall be converted by the receiver [host device] to a single character with bit 6 reset and all other bits unchanged.	TBD	Y
	4	If link-escaped, the octal characters: 0020 (0x10) and 0220 (0x90), 0021 (0x11) and 0221 (0x91), 0023 (0x13) and 0223 (ox93), 0177 (0x7f) and 0377 (oxff), shall be converted on receipt to a single character with bit 6 reset and all other bits unchanged.	TBD	Y
	5	The octal (hex) characters 0015 (0x0d) and 0215 (0x4d), the ASCII **CR,** shall be link-escaped if they are preceded by either a 0100 (0x40) or 0300 (oxc0), the ASCII @ character.	TBD	Y
	6	The option to link-escape all ASCII control characters (those characters with bits 5 and 6 reset) shall be implemented in both transmit and receive modes.	Y	Y
4.2.2, Header	1	All ZMODEM headers shall contain 2 **ZPAD**s, a **ZDLE,** a format byte, a type byte, 4 parameter bytes, and a CRC as shown in table 1 below.	TBD	Y
	2	One or two extra **ZPAD** characters received at the start of a header shall be ignored.	Y	Y
4.2.2.3, Hex Header	1	When parsing a hex header, the receiver [host device] shall ignore the state of the parity bit.	Y	Y
	2	For the type byte, the four position/flag bytes, and the 16 bit CRC, which are sent in hex, only the following character set shall be used: 0123456789abcdef.	TBD	Y

Section	Requirement		LP12	Grape-vine
4.3.1, Recommended Time-Outs	3	A header response timer shall measure the time between the completion of the transmission of the final byte of a header packet and the receipt of the first byte of a response.	TBD	Y
	4	If the header response timer expires before a required response to a valid and appropriate header is received, the header shall be retransmitted; except during file transfer.	TBD	Y
	5	If the header response timer expires following the tenth consecutive retransmission of a header, the device attempting to send the header shall be aborted by the **CAN**, section 4.1.3.1, Cancel Sequence.	TBD	Y
	6	If during a file transfer, the data line is quiescent for >= 10 seconds, the receiver shall attempt to restart the transfer using the ZRPOS header.	TBD	TBD
4.3.2.1, Transmit 'rz'<cr>, and ZRQINIT	1	To initiate the first ZMODEM session, the transmitting device shall send 'rz' (ASCII characters) followed by an ASCII **CR**, followed by a **ZRQINIT** header.	Y	TBD
	2	Following receipt of the ZMODEM initiation sequence, the receiver [host device] will respond with a **ZRINIT** header.	TBD	Y
	3	The receiver's retry mechanics for establishing the ZMODEM session shall be the same as defined in section 4.3.1, Recommended Time-Outs.	TBD	Y
4.3.3.1, Transmit ZFILE Header and Data Sub-packets	1	To initiate a file transfer after the end of the ZTERM session described in section 5, Physio-Control ZTERM Mode, the sender [cart device] shall send a **ZFILE** header followed by a **ZCRCW** data packet. This is defined as the file initiation sequence.	Y	TBD
	2	The receiver [host device] shall respond to the transmitter's file initiation **ZFILE** and data packets with a **ZRPOS** header indicating the offset in bytes from the first byte of the file at which transmission of data shall begin.	TBD	Y
	3	After the receipt of a valid file initiation sequence, the receiver [host device] shall cause to be created a destination file to receive the sender [cart device] data as defined in the file initiation **ZFILE** and data packets.	TBD	Y
	4	If the receiver [host device] cannot create a destination file for the sender [cart device] data as described in 4.3.3.1/3, then it shall send a **ZFERR** header.	TBD	TBD

Section	Requirement	LP12	Grape-vine
4.3.3.2, Transmit ZDATA header and Data Sub-packets	1 The sender [cart device] shall send the final data packet from the file being transmitted as a **ZCRCE** packet, which may be 0 bytes, followed by a **ZEOF** header.	Y	TBD
	2 Upon receipt of the **ZEOF** in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file and compare the total number of file bytes received to that contained in the **ZEOF**. If the total bytes sent and received are the same, the receiver [host device] shall send a **ZRINIT** header containing receiver capability flags and receiver buffer limitations.	TBD	TBD
	3 Upon receipt of the **ZEOF** in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file and compare the total number of file bytes received to that contained in the **ZEOF**. If the total bytes sent and received are not the same the receiver [host device] shall send a **ZRPOS** header up to the retry limit.	TBD	TBD
	4 Upon receipt of a **ZEOF** as described in 4.3.3.2/1, the receiver [host device] shall terminate the process by which data is added to the transmitted file. If this results in a file input/output error, the receiver [host device] shall send a **CAN** (section 4.1.3.1, Cancel Sequence).	TBD	TBD
	5 The receiver [host device] shall ignore session initiation sequence packets (see section 4.3.2.1, Transmit 'rz'<cr>, and ZRQINIT) received after a file-completion **ZEOF** packet and before a **ZRINIT** packet is sent by the receiver [host device].	TBD	TBD
	6 If a **ZFERR** header is received, the sender [cart device] shall send the cancel sequence.	Y	TBD
	7 During the file transfer, the receiver [host device] shall respond to data packets as follows: Packet Normal ResponseAbnormal Response Note **ZCRCE** None **ZRPOS** **ZCRCG** None **ZRPOS** **ZCRCQ** **ZACK** **ZRPOS-** do not wait for **ZACK** to proceed **ZCRCW** **ZACK** **ZRPOS-** wait for **ZACK** before proceeding	TBD	Y
	8 If, during the file transfer, the sender [cart device] receives a **ZRPOS**, then it shall begin sending data at the file offset specified in the **ZRPOS** header.	Y	TBD
	9 The receiver [host device] shall respond to a properly formed **ZCOMMAND/ZCRCW** frame with a **ZACK** packet with **ZPx** = 0x00.	TBD	Y

Section	Requirement		LP12	Grape-vine
4.3.4.1, Transmit ZFIN header 'The Shutdown Sequence'	1	The receiver [host device] shall respond to a **ZFIN** header with a **ZFIN** header and immediately terminate the session.	TBD	Y
	2	When the sender [cart device] receives a **ZFIN** header in response to a **ZFIN** header, it shall send the string "OO" (ASCII upper-case 'O') and terminate the session without requiring additional responses.	Y	Y
4.4.1, ZRQINIT	1	ZF0 shall contain **ZCOMMAND** if the program is attempting to send a command, 0 otherwise.	Y	?
4.4.2, ZRINIT	1	CANFC32 = 0 (16 bit CRCs) for Physio applications.		
4.4.5.4, ZFILE Requirements	1	A **ZFILE** header with the **ZCBIN** bit set sent from the sender [cart device] to the receiver [host device] shall override any other Conversion Option given to the receiver [host device] except **ZCRESUM.**	TBD	Y
4.4.7, ZNAK	1	The sender [cart device] shall send a **ZNAK** if and only if there is an inappropriate header before a **ZFILE** or after a **ZEOF.**	Y	TBD
4.4.8, ZABORT	1	To terminate a ZMODEM session, the receiver [host device] shall send a **ZABORT** header or the cancel sequence in section 4.1.3.1, Cancel Sequence.	TBD	Y
	2	If a **ZABORT** header is received, the sender [cart device] shall send the cancel sequence in section 4.1.3.1, Cancel Sequence.	Y	TBD
4.4.12, ZEOF	1	ZP0...ZP3 contain the total number of bytes sent.	Y	Y
4.5.1, ZMODEM 16-Bit CRC polynomial	1	The CRCs must be calculated according to this section.	TBD	TBD
5.1, Entering ZTERM Mode	1	Following the initial negotiation of the ZMODEM session in section 4.3.2, the sending device shall issue a **ZCOMMAND** packet with the following string: "{phy::term}<CR>" + Optional shell command string.	Y	TBD
	2	If the response to a **ZRINIT** is not a **ZCOMMAND,** then the receiver [host device] shall resend the **ZRINIT.**		

Section	Requirement		LP12	Grape-vine
	3	A Physio-Control receiver [host device] shall recognize the string in requirement 5.1/1 and shall enter ZTERM mode after responding with a **ZACK.**	TBD	Y
	4	If the receiver [host device] does not recognize the string in requirement 5.1/1, it shall respond with a **ZCOMPL (BAD)**, with ZFx = 0xFF and fail to enter ZTERM mode.	Y	TBD
	5	If a **ZCOMPL** or **ZCAN** (section 4.1.3.1, Cancel Sequence) is received by the sender [cart device] in place of the **ZACK** in response to the ZTERM entry request message as defined in requirement 5.1/1, the sender [cart device] shall abort the session.	TBD	TBD
5.2, ZTERM Valid Headers	1	Within ZTERM mode, the only valid ZMODEM headers are **ZDATA, ZACK, ZNAK, ZABORT,** and **ZCOMPL.** Any other ZMODEM headers shall be ignored.	Y	Y
5.2.1, ZACK	1	**ZACK** is sent as a response to a valid ZTERM frame without errors.	Y	Y
5.2.2, ZNAK	1	**ZNAK** is sent as a response to a header or data packet with errors.	Y	Y
	2	When a **ZNAK** is received as a response to a header or data packet, the software will retransmit the header or frame that caused the **ZNAK.**		
5.2.3, ZABORT	1	**ZABORT** shall cause a termination of BOTH ZTERM mode and ZMODEM sessions.	Y	Y
	2	The software shall transmit a **ZABORT** when ???.		
5.2.4, ZCOMPL()	1	A **ZCOMPL** header is sent as a reply to a **ZCOMMAND.**		
5.3.2, The structure of the ZTERM frame:	1	An **XON** character received at the end of the ZTERM frame shall be ignored.	Y	Y
5.4, Termination of ZTERM mode	1	To end a ZTERM session, the receiver [host device] shall send a **ZCOMPL** header.	Y	Y
5.5, ZTERM Packet Acknowledge and Time-outs	1	Within ZTERM mode, retransmit and time-out rules are the same as for regular ZMODEM.	Y	Y