** ZMP Documentation **

1. Introduction.

ZMP is a communications / file transfer program for CP/M which performs Xmodem, Xmodem-1k (often erroneously called Ymodem), true Ymodem and Zmodem file transfer protocols. Although tested with Z80DOS, ZRDOS and CP/M 2.2, there seems to be no reason why it shouldn't work with CP/M 3 as well. The only requirements are a Z80 processor (sorry about that!), a computer running CP/M in one of its various guises, with at least 45k of TPA (but the more the better!), and a modem.

When you try to pack this many features into one program, you end up with a pretty large file. The big problem occurs when file transfers are attempted: unless you have at least 4-8k of buffer size, you might as well use xmodem protocol. The approach taken in ZMP is to use overlays for various functions, accepting the time taken to load these from disk. Thus performance will vary depending on your disk setup: if you have a hard disk and a fast processor you will likely not notice the difference. If, on the other hand, you are running a Commodore 128 with CP/M on a 1571 drive, there is no physical reason why ZMP won't work, but you might consider investing in a book to read while the overlays load. (A suitable book to read might be a computer catalogue!).

The curious amongst you may notice that the beginning of the ZMPX.COM file has the magic 'Z3ENV' string, but don't let this fool you into thinking that you don't need to add terminal characteristics into the overlay if you have a ZCPR3 system. It has proved possible to persuade this particular C compiler to access ZCPR3's environment descriptor, but not for ZMP. Yet. Perhaps later. In the meantime, the startup code is there for it.

In order to produce a program which would work with most CP/M systems, the Zmodem protocol performed by ZMP is fairly simple. The transmit section uses 'Full Streaming with Reverse Interrupt', as Chuck Forsberg calls it in his description of the Zmodem protocol. The receive section uses 'Segmented Streaming'. This means that, if your system can do serial I/O and disk I/O at the same time, ZMP does not take advantage of the faster transfer rate which this capability provides. Since, however, I can't write and listen at the same time, and neither can my computer, and neither can the vast majority of CP/M computers, it seemed the best approach to take. Segmented Streaming means that the receive program tells the transmit program how big its buffer is. The transmit program then sends just that much data, then waits for an acknowledge from the receiver. We have encountered some Zmodem programs which send too much data in this case: errors will appear if this happens, but the protocol should recover and the file will be received intact (we hope!).

The string which ZMP passes to the receiving program to interrupt in case of errors is likewise simple. Basically it causes the receiving program to send a control-C character and then wait for one second. The receiver will then send its ZRPOS string, by which time ZMP, as the transmitter, should be ready to receive it.

2. Customization.

ZMP must be customized to suit your system. This involves overlaying the uninstalled copy of ZMP.COM (contained in this library as ZMPX.COM) with a user-written installation overlay. Some hints on writing this are given below; there is a blank overlay file in this library, or you may be able to obtain one for your computer from the same place you got this library. This value is set at 0145 hex, and should stay there permanently. See the notes in this document, and also the ZMP-OVL.UPD file, for more details on how to set up your overlay.

Once the installation overlay is written, assemble it with M80 (or SLR or whatever), use RELHEX to create a .HEX file, and use MLOAD to overlay it over the ZMPX.COM file to produce your very own ZMP.COM.

3. Operation.

The following files must be on the same disk and user area, which must be specified in your customization overlay:

ZMCONFIG.OVR	-- the configuration overlay
ZMINIT.OVR	-- the initialisation overlay
ZMTERM.OVR	-- the terminal overlay
ZMXFER.OVR	-- the file transfer overlay
ZMP.HLP	-- the help file (recommended).

Start the program with ZMP. The screen should clear, then a title message is printed, then ZMP enters terminal mode. You may type escape-H for help at this point. When you first run the program, the first thing you need to do is to enter the configuration overlay (type escape-C) and set all the defaults and others to suit your system as required. If you don't know whether to change something or not, it's probably better to leave it alone. When you exit the configuration program, answer 'Y' to the 'Make changes permanent' question. ZMP.FON and ZMP.CFG will be produced on your disk, in the same drive/user area as the above files.

Operation of the program is controlled by escape sequences entered in terminal mode. Escape-H gives you a list of options. Most of these are self-explanatory, but they will be summarized here.

B - Send Break to modem.

If your overlay has been set up to send a break command to your UART, this command will perform this function. (Some remote systems may require a break sent to them to interrupt Zmodem transfers).

C - Configure system.

This function is designed to set system defaults, save phone numbers etc. Changing baud rates etc. for a particular call are better handled with the escape-L option. If you answer Y or y to the 'make changes permanent?' question, the new configuration will be saved in a ZMP.CFG file, and the phone numbers in a ZMP.FON file for later use. The .CFG file is read, if it exists, when ZMP is first started.

D - Get disk Directory.

Gets a directory of the current drive and user area. Change to another with the escape-F command, described below. The directory will be sorted and will include file sizes unless you have so many files on the current drive/user area that there is insufficient memory available to sort them. In this case an unsorted directory, without file sizes, will be printed.

F - Disk/File operations.

This command is used to change the current drive/user area, to reset a disk in the current drive, and to view, erase, print or rename files. There are also options to give a directory of the current drive/user area, and to supply a new filename for the capture file. This filename may specify a different drive/user area than the current one. If capture mode is on, the status line printed when terminal mode is entered will state the capture file name.

H - Get Help.

Prints the ZMP.HLP file. You may then either type CR to return to terminal mode, or enter the required function key.

I - Initiate phone call.

Reads the ZMP.FON file, if any, and prints it. You have four seconds after typing ESC I, during which you may enter the letter corresponding to the required number, in which case ZMP will dial it without printing a list. Otherwise the full phone list will be printed, and you will be asked which number you want. Enter the identifying letter of the number you wish to call. You can also enter numbers not in the list. Multiple numbers can be called by entering them separated with commas. Dialing will then commence, and will continue until one of the numbers answers, or until you abort the process with the escape key. Note that the baud rate used for the call is that in the .FON file for that number, or the current one if a new number is entered. This function works best if the strings in the .CFG file have been set up for your modem, and the initialization string sent to the modem sets it into verbose mode for status messages (ATV1).

K - Display Keyboard macros.

The configuration option allows up to ten macro keys to be defined, and these are recalled by typing escape followed by the numbers 0 to 9. This function prints the current assignments.

L - Change Line settings.

This function allows temporary changes of baud rate, stop bits, and data bits. There is also an option to operate terminal mode in full duplex (locally typed characters are sent but not displayed on the screen), half duplex (locally typed characters are sent and also displayed on the screen), or echo mode (as for half duplex, but received characters are echoed to the remote system as well as being displayed). Don't have two computers talking to each other in echo mode unless you're bored. There are also options to allow/disallow control characters above CR to be displayed in terminal mode, to strip the parity bit in terminal mode, and to re-initialize the currently selected UART and modem at the current baud rate.

Like IMP, ZMP uses location 003C hex to store a value corresponding to the present baud rate. Then, if you leave ZMP and later re-enter, it checks location 003C. If it contains a legal value, then that baud rate is set for you, otherwise it sets the default baud rate as selected in the .CFG file.

You will also find in this menu an option to enable word wrap in terminal mode, and provision to switch to an alternate port. The word wrap function is rather simplistic, but it does work. It functions only with characters typed at the keyboard, and is designed to wrap on 80-character lines.

If you are talking to a BBS which also has automatic word-wrap, it's probably best to leave this turned off, otherwise you could end up with double-spaced lines. But if you are talking to a human operator, or if you are using a packet radio TNC, turn it on.

The alternate port switch allows use of two ports with ZMP. The status line (printed when terminal mode is entered) will display which port is in use. The configuration menu allows you to define which of the two is the default port, selected when ZMP is started up.

M - Toggle capture mode in Memory.

Received characters will be saved in a buffer and saved in a file named 'ZMP.LOG' when the buffer fills or when the command is entered again. A control-S/control-Q sequence is sent to the remote computer while the buffer is being saved in an attempt to stop it sending more data.

P - Toggle Printer.

This is similar to the 'M' command, except incoming characters are sent to the printer. This functions best if your system performs the BIOS 'List Status' function correctly.

Q - Quit the program.

Obvious. You will be asked if this is really what you want. Any entry other than N or n will exit to CP/M.

R - Receive a file.

You will be asked which protocol you wish to use. The default is ZMODEM. The <X>modem option will allow either 128-byte blocks (standard XMODEM) or 1k blocks (XMODEM-1k), since this is decided by the transmit end. If an attempt is made to receive a file which has the same name as a file on the current drive/user area, the current one will be renamed to .BAK and the new one will then be received normally. (However, see below for the transfer resumption feature in ZMP v1.5 and above).

Note that the byte count on the screen is not kept up-to-date on Zmodem receive. This is because the data arrives non-stop and there is simply no time available with non-interrupt driven computers to update the screen. An update is performed if errors occur, and when the computer pauses to write to the disk.

Starting with version 1.4, Zmodem file receive will commence automatically upon receipt of the sender's ZRQINIT string. Thus all you need to do is have the sender initiate the transfer, select the drive and user area you wish the file(s) to be received on, and wait..

ZMP v1.5 adds the ability to resume an interrupted Zmodem transfer. If a Zmodem receive attempt fails (either because of manual cancellation or massive errors), you will be asked if you wish to save the portion of the file already received. If not, it will be erased. If so, and a subsequent Zmodem receive attempt would result in a file of the same name, you are asked if you wish to resume the transfer. If you do, transfer will start at the end of the file. Otherwise the old file will be renamed to .BAK as before.

Since this feature is a function of the receiver, it should work with any Zmodem implementation which conforms reasonably closely to Chuck Forsberg's standard. It has been tested with ZMP and RZMP, and I would like to hear of any programs with which it doesn't work. No attempt is made to determine if the files are the same up to the commencement point: the Zmodem protocol provides two ways in which this may be determined (file date and CRC), but neither has yet been implemented in ZMP.

S - Send a file.

Operation is similar to the receive function. Additional options available are ASCII send and the capability of distinguishing between normal Xmodem and Xmodem-1k. In Ymodem and Zmodem modes, wildcard filenames and multiple filenames are allowed. Multiple filenames should be entered separated by spaces. In all cases, files on different drives/user areas may be specified by supplying a zcpr3-style du: prefix (e.g. C7:NEATPROG.WOW).

Byte count information is displayed in Zmodem mode on transmit. This causes noticeable breaks between packets, but it is felt that this is outweighed by the usefulness of the information.

ZMP's Zmodem mode is capable of CRC-32 operation, although CRC-16 mode is used if the receiver is incapable of CRC-32. Some other terminal programs, however, do strange things when faced with a receiving program which claims it can do CRC-32 (we have encountered one for the Amiga which exhibits this problem). If this happens, the Esc-L menu and the configuration overlay have an option to disable CRC-32.

U - Execute user-defined overlay.

This feature is designed to allow execution of user-produced overlays, and other protocols which won't fit into the normal transfer overlay. These must be called ZMxxxxxx.OVR, and must be linked to be loaded at a specific address which will probably vary with different versions of ZMP. For v1.5, this address is 03ABh. I may produce a list of available functions in the main ZMP.COM file for use by these overlays if there is enough interest.

After typing ESC U, you will be prompted for the overlay name. The overlay will then be executed as normal. An example of a user overlay is ZMYAPP.OVR, which is described further below. Other possibilities include a Kermit overlay, but as this would be a lengthy project, and as I have no use at this time for a Kermit overlay, whether or not it gets done depends on the level of interest shown. If you are interested in a ZMP Kermit overlay, or if you can think of any other overlays which could/should be implemented, please let me know.

X - Hangup.

This function causes the modem to disconnect from the phone line, by momentarily dropping DTR.

Y - Print screen.

Allows the current screen to be dumped to printer. Note that this must be supported in the overlay: most terminals are incapable of this function. The standard overlay prints 'This function not supported.'. If you can make it work on your system, good luck!

Z - Clear screen.

Allows the screen to be cleared. Useful if it fills with rubbish.

4. Hints for overlay writers.

Most of the code is quite straight forward and will present no problem. The section most likely to cause trouble is that part concerned with setting the required baud rate and (especially) the code for setting the correct parity, number of stop bit(s) and data bits.

We suggest that you first get the main part of the code going by putting a return in the init: section at the point where it says "put your code here". This will let you check the rest of the code. If the opening screen looks 'neat and tidy' then you know that 1) the code is in the right place, and 2) your cursor addressing is working. Check the I/O by talking to your modem and then by exchanging a file with another zmodem user/board after you set the I/O chip up with another program such as IMP.

Now get the baud-setting working and then finally the bit-bashing required to set the parity, etc.

5. Other information.

a) ZMP at higher baud rates.

When I first produced ZMP, I was more interested in producing a universal Zmodem program than anything else. Originally (several C compilers ago!) I had difficulty getting it to work even at 300 baud, and so little thought was given to accommodate higher transmission speeds. In particular, there is a "designed-in" bug/feature which would probably preclude ZMP working at much over 4800 baud. The problem is in the user overlay, in the mrd: routine. The requirement here is to have a routine which returns either when a character is available at the modem (in which case we return TRUE), or 100 mS has elapsed (in which case we return FALSE). The catch is that I used 100 x 1 mS waits, between which we test for a character. A little calculation will show that a 9600 baud character will take a little over 1 mS to transfer, and 19200 baud characters take half this time. Thus we are practically guaranteed to miss characters at 19200 baud, and even 9600 baud characters leave little processing time to spare. Two possible ways to overcome this are:

i) Make the wait time shorter. Thus we could wait 1000 x 100 uS periods instead. This, however, makes the actual wait time more unpredictable, since subroutine call/return times are comparable to the wait time. It also just puts off the evil day.

ii) Use a hardware timer to determine whether 100 mS has elapsed. This is the preferred approach. Thus the mrd: routine would loop continuously, exiting when either there was a character at the modem or when the hardware timer expired. An embryo CP/M-68K version of ZMP using this approach has proved capable of reliable transfers at 19,200 baud (although one must admit that it IS running a 68010 at 10 MHz!).

I would like to hear from anyone who has had any success with either of these two approaches.

b) The ZMYAPP overlay.

This performs the amateur packet radio YAPP protocol, and is accessed by the ESC-U function. Operation is pretty well obvious, but there are a couple of problems with it, depending on the hardware capabilities of your computer. The main one is that, if the file to be received is greater than the buffer size in ZMP, the ZMP computer MUST be capable of hardware handshaking (CTS/RTS). There appears to be no other way of stopping a TNC (which must be in transparent mode) while we write the buffer to disk. The same thing will occur if the TNC buffer fills: if the TNC cannot stop the ZMP program with CTS (DTR???) then its buffer will overrun. The ZMYAPP overlay works fine with small files. Again, I would like to hear from anyone who has success with larger files on different machines.

6. Acknowledgements.

ZMP was developed from Hal Maney's Heath-specific HMODEM II. I would like to thank Hal for writing HMODEM: CP/M users have been without ZMODEM capability for far too long. As requested in the source file, acknowledgement is given to him therein. Appreciations also go to the authors of the Hi-Tech C compiler, which proved to be capable of producing fast and compact code for Z80 machines.

ZMP in its various incarnations is refined by suggestions from you, the user. In particular, I would like to thank Mike Allen, Richard Kopplin and Fred Haines for their invaluable suggestions. I may sometimes be a little slow at implementation, but I usually get there eventually! Fred Haines has also kindly offered to be the U.S. collection point for bug reports, suggestions, etc. His address appears at the bottom of this document, and he'll forward them to me via what he calls 'U.S. Snail'. I will try and respond using what I call 'Australia Pest'.

I would also like to thank Lindsay Allen, sysop of Z-Node 62. His name was removed from the original zmp11 title screen at his own request, since I had done most of the work in modifying Hmodem to work on other machines. Without Lindsay's encouragement at difficult times, suggestions as to how to go about recalcitrant procedures, and experience in file transfers, ZMP would not have been produced. Thank you.

7. Finally...

The files contained in this library are placed in the public domain. Just don't sell it, claim you wrote it, or do anything similar that might annoy me. Above all, don't bother trying to sue me if it doesn't work, or you tripped over the disk, or anything similar. I haven't distributed the source files partly due to the size of them, partly due to the fact that compilation is messy (several modifications were needed to "standard" library functions!), and partly due to the fact that there's still work to do on them. Besides, I feel a certain fatherly feeling towards ZMP, having spent most of my spare time for the last four months working on it. So here's the deal: I will continue to support ZMP (and RZMP) until I get sick of it. This could take an unknown amount of time! At that point, I will release the sources into the RCP/M community, and you may make of them what you will.

ZMP has a remote system relative, called RZMP. This allows Zmodem transfers to and from remote systems. It should be available from the same place from which you obtained ZMP.

I have also produced an extremely cut-down version of ZMP which runs under CP/M-68K, currently running quite well as a single .68K file (no overlays!) on a system with 128k bytes of memory. If there is enough interest from CP/M-68k users (are there any??), I could be persuaded to upgrade this version to the point where it could be released.

Comments and suggestions are welcome. Bug reports are not so welcome, but we'd like them anyway! Send either to:

Z-Node 62
Perth, Western Australia
(061+) 09-450-0200
(Soon to be on FidoNet)

U.S. users may send reports/comments to:

Fred Haines,
733 North King's Road, Apt. 356
Los Angeles, California 90069

-- Ron Murray
26th March, 1989