Chapter 8:

OTHER VALDOCS FILE CONTROLS

MAIL

INTRODUCTION

Via MAIL, the user communicates electronically with the outside world. The ensuing chapter describes MAIL protocol (similar to Ward Christensen's modem program, see Appendix A), the HASH algorithm, address book, send and receive logs, and error messages. The Mail program utilizes standard system calls.

MAIL PROTOCOL

The following are basic definitions of terms commonly used with MAIL:

Protocol: The sequence of bytes transmitted back and forth which control the communications process. Such bytes may be used to indicate the beginning or end of a portion of data, or communication with another system. Also used to name the protocol which controls such communications.

Talk: Used synonymously with *Person to Person.*

SOH: Acronym for *Start of Header.*

Start of Header: A byte with the hex value 01, used to signify the beginning of a *block* in *file transmission.*

EOT: Acronym for *End of Transmission.*

End of Transmission: A byte with the hex value 04, used to indicate the completion of all *file transmissions* for this session.

ENQ: Abbreviation of *Enquire.*

Enquire: A byte with the hex value 05 signifying the end of a single *file transmission* and indicating that another file will be sent immediately.

ACK: Abbreviation of *Acknowledge.*

Acknowledge: A byte with the hex value 06, indicating that the previous *block* or *protocol* information was received and understood.

NAK: Abbreviation of *Not Acknowledged.*

Not Acknowledged: A byte with the hex value 15, indicating that the previous *block* or *protocol* information was either not received, or not understood.

XON: A byte with the hex value 11 sent by one system to tell the other system it may transmit characters.

XOFF: A byte with the hex value 13 sent by one system to tell the other not to send any characters until an *XON* has been sent.

Block: A grouping of information used for *file transmission* consisting of *SOH* followed by a sequential *block* number, followed by the complement of that block number. This is followed by 128 bytes of data, and a 1-byte *checksum.*

Checksum: A single byte which is the sum of all 128 data bytes sent in a *block.* ONLY the data bytes are summed.

Bell: A byte with the hex value 07 which makes the terminal beep or ring. It is used in *talk* mode at the very beginning of communications.

The *protocol* in the MAIL program is very similar to the *MODEM protocol* as developed by Ward Christensen (the two are compatible--see end of chapter). For the purposes of the MAIL program, the computer answering the phone call is the *receiver* or *answerer* and the computer making the phone call is the *sender* or *originator.*

Once carrier has been established, the originator waits for *NAK* from the receiver, at which point, a *block* is sent. Sequence numbers sent after the *SOH* begin at 1 and are incremented by 1 for each subsequent block.

On receipt of the block, the receiver sends an *ACK* if the *checksum* calculated from the received data bytes matches the one sent. If all the characters in the block were not received (due, for example, to line problems) the receiver waits no more than one second for a character in the middle of a block. Upon timeout, the receiver sends a NAK to inform the sender that the block was not accepted.

Only a few conditions are fatal. Obviously, communications are broken if the carrier is lost. If the receiver gets a block with the same sequence number as the previous block (e.g., the sender did not receive an ACK), he simply sends an ACK to tell the other side to send the next

block. Likewise, if the receiver gets a block with a sequence number greater than that expected, he must send an *external abort* signal (a byte with the value of 18 Hex). The user has missed an entire block and there is no way to tell the sender to retransmit a block that it thinks was given an *ACK*.

When the last block has been ACKed by the receiver, the sender transmits an *EOT*, a signal to the receiver that the session has been completed.

The above, in summary, describes the basic MODEM protocol, with which MAIL is compatible. MAIL uses a few extensions to this protocol, which indicate the name of the file and who is sending it. There is also a special *talk protocol*.

As mentioned, sequential block numbers start with a 1. MAIL sends a block number zero before this--a regular 128-byte block with a header and checksum, whose contents are the file name (first 11 characters in FCB format) and the name of the sender (from the twelfth character to a 0 terminator). The remaining block data is unimportant, but must be included in the checksum.

MAIL sends this I.D. block and waits for an ACK. If it receives a NAK instead, it will try to retransmit three times. If all three responses are NAKs, it simply sends the file, starting with block number 1.

An additional protocol facility aids in sending a group of files to the same person. If, after a regular file transfer, another file is sent to the same person, an *ENQ* is transmitted, instead of the EOT. On receipt of an ACK, the sender sends the I.D. block for the file, etc. If ENQ receives a NAK, it disconnects and redials.

The *talk* mode is initiated by the sender. Once communications are established (i.e., carrier) the sender transmits three control-Gs (hex 07, called *bel*). The receiving MAIL program returns the message, "You are connected with VALDOCS", and, on the bottom line of the screen, informs the user that a "[PERSON IS WAITING]". An external abort (hex 018) received by either side at ANY time during person-to-person communications is a signal to end communications.

The above is a summary of the protocol for communications with a Valdocs MAIL (or regular MODEM) program. Such aspects as the Address Book and the Send and Receive Logs interface with MAIL on a more intimate basis.

PRINT

Applications can print a file in one of two ways: call the print module and spooler, or print directly. Because Draw, Mail, and Index, *do* print directly, files cannot be stacked up, nor can printing proceed while spooling is active.

PRNT.CHN

Prnt.chn is the print module. When called, prnt.chn first checks to see if the printer is available. If it is not (e.g., the power is off or the printer is offline), prnt.chn displays a message to that effect.

Otherwise, prnt.chn chains directly to the Indexer and requests that the user choose a file. Indexer chains back to prnt.chn with the chosen file, and prnt.chn displays the menu of printer options. To start printing, simply hit the Print key a second time. Prnt.chn hands the file to SYSINIT.

SYSINIT SPOOLER OPERATION

SYSINIT sets up the spooler (actually part of the SYSINIT module) in bank 1. To access the spooler routine--which is in the normal bankswitching area of memory--SYSINIT goes through the "backdoor"--the reserved, non-bank switching part of memory. Control is automatically passed back to prnt.chn, which then chains back to the original calling program.

SPOOLER CAPACITY

Spooler capacity is limited to five files: one printing file and four on the stack. 512 bytes are required to remember every file to be printed.

SCREEN SELECTIONS

The Print key allows two screen selections. C^ Print performs a screen dump in character or bit modes; C^ / Shift / Print prints whatever is on the present screen.

THE ADDRESS BOOK

The address book features all listings typically found in an old-fashioned address book. In addition, a label program allows the user to format his own mailing labels. Briefly, the address book keeps a record listing of address cards, in alphabetical order, that can accommodate first and last name, company name, address, post office box, etc. The address book locates the requested address card by utilizing the HASH concept.

THE HASH TABLE

A program can locate a certain item within a listing in one of two ways: it can move sequentially down the list looking for the item, which takes a great deal of time; or it can use a HASH formula which assigns a unique number, called a key, to that item. The key, and a pointer to the address card, go into the HASH table. The name on the address card cues the program to put the last name through the HASH formula and pop out the key-number. The program then goes to the HASH table, finds the key, and locates the address card. The second method, which is much faster, is used in the address book.

THE HASH FORMULA

A HASH formula is created by taking a last name and putting it into a 16-bit value. To find out the entry number of this last name, begin with a zero, shift to the left a certain number of times (determined by the individual HASH formula), add in a character from the name, then loop around until all of the characters have been exhausted. The number of shifts is inconsequential, as long as it's consistent. The general idea is to take a string of characters, encode it, and disperse it evenly over the address book.

HASH COLLISIONS

The ideal HASH algorithm would always assign a unique number to the last name. Unfortunately, sometimes a collision--two names with the same key--occurs.

Generally, a collision results from one of two situations: either a poor HASH algorithm (which puts the keys next to each other in the address book), or insufficient space. The Valdocs user determines his own file size in the address book. If he selects a small area, more collisions will occur, i.e., the system will slow down because the program starts examining all the records sequentially. This causes more file I/O.

If, for example, the program is inserting a new name with the same key, it will find the key, go to the next blank record, and insert it there. A blank record is determined by the first character in the last name in the HASH record (see the HASH record below for reference). If two zeros occur in the first character, the program recognizes a blank and inserts the new name.

DELETING A NAME

When the program finds a blank, it knows that nothing was ever stored there. If a name is deleted, however, the program *could* see a blank and assume that it is the end of a sequential list--which it isn't. Therefore, when a name is deleted it is NOT made a blank--instead, FF is inserted in the first character of the last name (see HASH record format). The FF simply indicates that this place was at one time HASHed out to a number that was later deleted.

ADDRESS BOOK AND MAILING LABEL HEADER FORMAT

The first record is the header, not the first data record. Total record length is 128 bytes.

2 bytes = number of entries.
2 bytes = lowest alpha entry. (Pointer to a length list of names in the HASH table, in alphabetical order.)
2 bytes = position. First recorded in the files that contain data.
2 bytes = free data index. (Those records not yet in use).
25 bytes = your name in the "who am I" record.
3 bytes = area code.
1 byte = the character to send before the area code (1).
4 bytes = the callout prefix (e.g., 9).
1 byte = pulse or tone dial (t for tone, p for pulse).
41 bytes = not used. Reserved.
1 byte = label width of characters.
1 byte = number of horizontal spaces between labels being printed.
1 byte = number of lines per label.
1 byte = number of vertical lines between labels.
1 byte = labels per line (how many labels across one printer line).

8 blocks of 4 bytes each:

Designating the label section, which enables the user to design his own label format by selecting various fields in the address card.

All blocks are formatted in the same manner. Each block corresponds to one printed line, so a label can contain up to

eight lines.

A number of fields can be selected onto the label. Each field has an assigned letter. Simply put the letter corresponding to the selected field into the block.

S = first name
L = last name
M = company name
T = title
A = address
P = p.o.box number
# = suite number
C = city
S = state
Z = zipcode
n = notes

7 bytes of zeros follow the eight blocks.
1 byte = indicates type of HASH method used.

HASH RECORD

The number of HASH records in the file equals the number of allocated records divided by four. Each HASH record is 128 bytes.

The HASH record contains the first and last name, a link to the rest of the names record, and a link to the next HASH record in alphabetical order.

Format of HASH record:

128 bytes apiece. Fourentries of 32 bytes each per record. All entries are formatted identically.

13 bytes = last name.
12 bytes = first name.
2 bytes = reference to address card, the pointer itself.
2 bytes = pointer to the next HASH entry in alphabetical order.
3 bytes = zeros. Reserved.

(The first character of the last name indicates a free or unused entry. If the card is deleted, the first byte of the entry is either 00 or FF HEX. 00=never used. FF=deleted.)

ADDRESS CARD FORMAT

Each address card is two CPM records. A CPM is 128 bytes. There are 16 fields in each address card:

1st field 8 bytes = reserved.

2nd field 12 bytes = first name.
3rd field 13 bytes = last name.
4th field 35 bytes = company name.
5th field 25 bytes = title.
6th field 35 bytes = address.
8th field 8 bytes = P.O. Box.
9th field 8 bytes = suite number.
10th field 25 bytes = city.
11th field 5 bytes = state.
12th field 9 bytes = zipcode.
13th field 23 bytes = telephone number for voice line.
14th field 23 bytes = modem or computer telphone.
15th field 26 bytes = comment field.
16th field 1 byte = baud rate.

(The HASH records point to the address cards which are allocated sequentially.)

SENDLOG.MAL AND RECVLOG.MAL FORMATS

The two formats are basically the same. There are 256 bytes total:

128 bytes = reserved.
11 bytes = "filename" not "filename.fil".
25 bytes = name of person going to or recieved from.
8 bytes = ASCII TPM format of the date: month/day/year.
8 bytes = time started sending: hour/minute/second.
8 bytes = time it completed.
1 byte = status (see ERROR MESSAGES).
1 byte = baud rate (0=300 baud, 1=1200 baud).

The rest of the bytes are unused.

ERROR MESSAGES

All error messages are derived from the status byte in the send and receive guide.

0 = ok
1 = file error
2 = disk full
3 = comm error
4 = modem error
5 = phone error
6 = external abort
7 = checksum error
8 = no response
9 = not sent yet
10 = abort from log
11 = sending now
12 = being aborted
13 = done

MODEM 7 PROGRAM

by Mark M. Zeiger and James K. Mills
Modified for Epson QX-10 by
Bruce Ratoff & Roger Amidon

This program uses the file transfer routines written by Ward Christensen in his CP/M file transfer program (V2.0 as of 8/6/79) and is compatable with his program in single file transfer mode. Multi-file transfers are only possible between two systems running the program described below.

This program has three functions:

1. Communication
2. Program transfer
3. Modem control (for PMMI Modem)

COMMUNICATIONS

The program may emulate a terminal or echo data back to sender (act as a computer).

Terminal Mode - 'T' Option

The terminal mode may be called with or without a file name. If a file is specified (it should be a new file), then anything received by the modem may be saved in memory and later written on disk. The save feature is toggled ON/OFF by Control-Y. On an IMSAI the front panel LEDs will indicate that memory save is toggled on by showing the binary value of the ASCII character received. For those with no front panel, a colon (:) will be printed at the beginning of each line when memory save is active. The colon will not be transmitted over the modem nor will it be saved in memory. If a file is not specified, then memory save can not be activated.

If the memory buffer is full (the buffer is from the top of the program to the bottom of BDOS), the contents are automatically written to disk (but the file is not yet closed). Communications may then continue with the buffer reinitialized. The computer with which you are communicating must accept the X-ON and X-OFF (Control-Q and Control-S) conventions or data will be lost.

When communications are over, use Control-E to exit from the Terminal mode and enter the Menu. The file to which you are writing must then be closed by using the 'WRT' command. If this is not done, all data will be lost. I decided not to close the file automatically since there will be times when you leave terminal mode and then decide to re-enter. This may be done while in the Menu by using the 'RET' command. You may re-enter Terminal mode and save in the same file as many times as you wish as long as you have not closed the file with the 'WRT' command.

While in Terminal mode, Control-T will put you in File Transfer mode. This will allow you to send the contents of an ASCII file over the modem. This routine does no error checking and there are no protocols specified between this program and the receiving computer other than that it should be ready to receive data via the modem. Control-X will cancel the transfer.

Computer mode - 'E' Option

This mode echos data received by other computer. Only one computer may be in this mode at one time. There is no save feature in this mode. Useful if you wish to communicate with somebody running the terminal portion of the program.

File transfer - 'S' and 'R' Options

These features are the same as in the CP/M Modem program written by Ward Christensen except that upon completion of the transfer, control returns to the Menu unless the secondary 'T' option has been selected. In the latter case, control returns to Terminal mode. Remember that if you are operating a remote computer using a timesharing program (such as Ward Christensen's "BYE"), the remote should be instructed to send or receive in the quiet (Q) mode as a secondary option.

Examples of commands for sending and receiving are listed below.

Multi-file transfers using the B (batch) secondary option, more than one file and ambiguous filenames may be transferred. To send files, use the primary option "S" and the secondary option "B" (along with any other secondary options and baudrate). To receive the files being sent, use the "R" primary option and the "B" secondary option. Files may not be named since filenames are sent by the sending program, but a disk drive may be specified (or else the files are written to the default drive).

Backup Option

A byte at the beginning of the program (106H) creates a backup file if a file on the disk has the same name as the file being received in multi-file transfer (see MODEM.SET). If this byte is set to OFFH, a backup file will be created. If it is zero the file on the disk will be deleted before the new file is received. If you are running CP/M 2 and a file on the disk is designated R/O or SYS, a backup will be created whether the byte at 106H is set or not.

Be careful--if you are running CP/M 1 and the drive on which you are receiving has an R/O file with the same name created by CP/M 2, the R/O file can not be accessed (found, changed, erased, etc.) by CP/M 1. You will therefore have two files with the same name when you are running CP/M 2. To fix this problem, use CP/M 1 to change the name of the file that is not R/O. Then use CP/M 2 to do what you want with the R/O file.

Multifile transfers may only be done from the menu. It may not be specified when the MODEM program is called. In other words A>MODEM SAB *.COM will result in an INVALID OPTION error message.

Return to Menu - 'M' option

When asked to select an option, 'M' returns to Menu.

THE MENU

If the Modem program is entered with no option, the Menu is called. The Menu gives the choice of selecting the standard options as defined by Ward Christensen (T, E, R, and S). The Terminal mode has been greatly expanded as described above.

The "R" and "S" commands must be called with a filename or you will be required to enter the primary command (S or R) and the filename again (but NOT the secondary options). If you want multi-file transfers, then the "R" option does not need a filename.

RET: You may also enter terminal mode using the 'RET' command, but no data will be saved since a file may not be named with 'RET'. Use the 'RET' command to RE-ENTER Terminal mode after it has first been entered with the 'T' option. If this is done, you will still be able to save the communications if you were doing so before you exited the terminal mode.

WRT: The 'WRT' command must be used after leaving Terminal mode. This writes the last buffer to disk and closes the file. Failure to use the 'WRT' command results in loss of all data and a file of length zero.

DEL: The 'DEL' command erases the most recent file accessed in Terminal mode. Useful if you decide after communications that you don't want to save information just gathered.

BYE: The 'BYE' command reboots and returns to the operating system.

EXP: The 'EXP' command is a toggle which causes the menu to be printed or not printed. Initially the menu is on unless the program is called with the "X" option (ie. A>MODEM X).

DIR: The 'DIR' command lists the directory of a CP/M disk. A drive may be specified (ie. DIR B:) or the default drive will be listed. After the directory is listed, the menu will not be printed.

Secondary options:

B - multi-file mode for sending and receiving files
T - return to terminal mode after transfer (memory save off)
R - view what is received in file transfer
S - view what is being sent in file transfer
V - view what is being sent or received in file transfer
T - terminal mode (used with remote-controlled computer)