## TERMINAL MODES

Mc Term allows the user to communicate with other computers in more than one way. The operator's needs will govern the method the user will choose. The user may communicate directly with another user or merely allow the disk drives to talk with each other. The user also has the opportunity to record the conversation for future use. These files may also be saved for either referencing or editting.

The basic modes for establishing the communication process have been explained in the previous pages. Now the user would like to know what needs to be done in order to establish this contact, and how the user will go about making use of this material. The next few sections will explain how this happens. Then the operator may begin to enjoy talking with other computers and the people who use them.

> [Most of the following examples will explain how the user may connect one computer to another with the use of a modem as the hardware intermediary. It is also possible to direct wire a PET computer with another PET computer. A different cable will be necessary for connecting the two PET computers. This cable is available from Madison Computer and comes in different lengths. We have found reliable results for PET to PET cables at 1200 baud for lengths up to 400 feet, and calculations allow for satisfactory lengths up to 1500 feet. Please call for more details on cable availability. Otherwise all of the explanations for the use of this configuarion with modems also apply for direct linked systems.]

### PREPARATION

By now the user is familiar with the program and how to load the program into the computer. The next step for the user is to set the protocol levels of the program so the computer may operate as a terminal. These levels should be set from the main menu.

Generally, when someone would like to connect a terminal to a computer there are certain protocol levels already known. If not, the user will have to assume certain protocol levels for the other computer, or use the default modes. Usually the user will know the baud rate, the parity, the form of duplex (local echo-on or off), and possibly whether the other computer may read upper case characters only. If this knowledge is established prior to the time of connection, everything will work quite smoothly. Otherwise some time will be needed using trial and error methods to find the proper protocol levels of the two computers. Sometimes the other computer's program is designed to give a great amount of flexibility for solving these problems. If it isn't, then the Mc Term program will be able to control the protocol levels for you.

## SIMPLE TERMINAL MODE

1) The user should set the protocol levels for communicating with the other computer. The items to be set are: the baud rate (this should also coincide with the connecting hardware, such as the modem), the parity, and the duplex mode (local echo). After setting these modes the user may proceed in the simplest configuration of terminals - the dumb terminal.

2) The user should type the letter "t" to enter the terminal mode. The main menu will be replaced with the menu prompt:

> "*** To get control mode, type the rvs key then the stop key."

This is the method for the user to return to the main menu. Doing this a couple of times will familiarize the user with the method. This prompt will scroll off the screen as the screen is filled with text.

3) It is in this "terminal" mode that the user can communicate with other computers. Once the user is in this mode, the other computer can be accessed by dialing its telephone number. When the telephone connection has been made properly a high pitched sound will come from the receiver. The telephone receiver should then be carefully and firmly placed in the modem. The carrier light will be ON if the modem has picked up the signal.

4) At this point the user should able to directly communicate with the other computer. Sometimes it may be necessary to hit a couple of carriage returns to get the other computer to acknowledge your presence. Other computers may require an immediate identification of some form, although this will generally be known to the user prior to accessing the system. While in this mode, the user's computer will act like a normal terminal. Information can be passed back and forth without any difficulty.

> NOTE: CONTROL KEYS - All terminals need to utilize control keys for speciality functions. Since they are not obvious, a brief discussion should clear this situation.
>
> The off/rvs key functions like a control key, when in the transmission modes. Whenever the the off/rvs key is pushed the next key will be a 'control' character rather than a text character. The control attribute will only last for one key stroke. If the user needs to type a string of control characters, then the off/rvs key needs to be pushed prior to each character. Visually nothing will occur when the off/rvs key is pushed, however when the next key is pushed it will appear in the inverse-video mode, representing a control character. If another key is typed, it will appear as normal text rather than as a control character.
>
> Some of these control characters represent specific functions. Some of these are:

Control @ = a break key
Control ' = a null key
Control u = an interrupt key

For further in-depth discussion of these characters, please refer to the ROM documentation section of this manual.

## RECORDED TERMINAL CONVERSATION

The next beneficial point of Mc Term is the feature to record to the disk the conversations of the computer while in the terminal mode. (This feature is not available to users with a cassette drive as the only means for mass storage.) Mc Term has made provisions for the naming of a file for the conversation, the ability to open and close this file, and the ability to link files together for editting through a WordPro program. The initial descriptions for this activity was mentioned in the discussion of "Filename/Open File."

The explanation of this process will begin with a quick review of the "Simple Terminal" mode.

1) Set the various protocol levels (baud rate, parity, local echo, and upper/lower case) for conversing with the other computer.

2) Create the filename ("f") for this conversation, either use the default name or create a new filename. Be sure to include the number of the disk drive, the filename, and the file type (prg, rel or seq). The program will generate the linking numbers in the filename.

3) Open ("o") the file, and respond to the question regarding the scratching of other files which may have the same name. At this time the disk drive light will be ON, indicating an open file on the disk - DO NOT REMOVE THE DISK during this time. Disks may switched prior to opening the file, however once the file has been openned, make sure the disk stays in the drive until the file is closed.

4) Enter the terminal mode ("t") and establish contact with the other computer.

5) When the user is ready to record the conversation, press the rvs key and then the home key. The word "start" will appear in the inverse video mode. At this time the user will be recording the conversation of the two computers. Hitting the rvs key and then the home key again will stop the recording of the conversation. This can be repeatedly turned on and off until the file has been closed. While recording, anything which is typed by the user, and anything sent to the user via the computer, will be recorded to the disk. It will be recorded on the disk under the filename established on the main menu.

If the user had the translation mode for WordPro ON, then the file will be only 140 lines (the default amount unless altered) long. [User's with 40 column Commodore computers may wish to change the default length for WordPro files, please see instructions on altering the default mode.] If the conversation is longer than this amount, the Mc Term program will create and link the next file. If the translation for WordPro is OFF, then there will be only one file. This single file will also be a sequential file.

A possible confusion may arise at this point, however with a little practice this will be cleared up. The sequence of RVS and then HOME will turn the recording session either on or off. The sequence of RVS and then STOP will return the user to the main menu. Returning to the main menu can be done during the recording of the conversation with no effect to the recording of data.

To to close the file of the conversation, return to the main menu ys and stop). The user may then close the file either by typing "o" for close, or typing the letter "e" to exit the Mc Term program. Typing "c" will allow the user to continue in the Mc Term program and perform other desired operations.

## SEND/RECEIVE DISK FILES

So far this manual has explained how to communicate with another computer and how to record that conversation. The conversation could be simple text created by the interaction of the two operators, or it could be a conversation from a computer which "dumps" files of information across the lines. For example, these files could be news reports or item listings. It's even possible to receive program listings from other computers this way. This method at times is relatively slow and delibrate, it is fine for interactive transmissions. However at times it will be preferable to merely send an entire file, and review or edit the file at a later time. Entire reports could be sent to a central office from a remote site. Mc Term has the provision to send entire files of information, either sequentail,relative or program. There are two ways to send files: 1) with crc, cyclic redundancy checking, or 2) without crc. For PET to PET transmissions, the user will probably wish to take advantage of transmitting with crc. The Mc Term program will automatically check for characters which may have been lost during the transmission. Many other computers will not be able to support cyclic redundancy checks, therefore the user will need to send the file in the other form, without the checks. In that method the transmission will be done character by character but at a rate much faster than normal typing speeds. The next two sections will explain how to send and receive the disk files, with or without crc. Another later section will explain how to send and receive WordPro files. The combination of writing/editting WordPro files and sending them in the block form will prove to be a powerful and beneficial activity for any PET user. This is the beginning of electronic mail.

> NOTE: These sections do not presently apply to users with the cassette version or with the downloaded ROM version. Mc Term is designed to transmit files to and from a computer with a disk drive. Therefore it is necesary to have a disk drive for block transmissions.

## SEND FILE - WITH CRC

### SENDING COMPUTER

1) Set the various protocol levels (baud rate, parity, local echo, upper/lower case) for conversing with the other computer. BOTH of the computers MUST be in the PET to PET communication mode and have the same protocol levels.

2) Establish contact with the other computer to coordinate communication activities. Make sure both terminals are in the PET to PET communication mode. If they are not the same then there will be some difficulties when trying to send the file.

3) From the control mode, when the user is ready to send a disk file, push "s" to enter 'send a disk file', and "1" to 'send with crc'. The main menu will be replaced with a prompt indicating that you will be

## STEM CALL LOCATIONS

### stem setup (sys+0)

Reserves buffer space and activates serial send and receive by setting up internal values, altering the memory protect pointer, and setting the interrupt service vector. This call is to be done exactly once to set up the communication routines. Once it is done, other calls may be made. The setup call is not to be done again, however, unless an XX call is done.

Other calls will LOCK UP THE PET if the setup call has not been once first.

Usage: since the end-of-memory pointer is reset, this setup call MUST be followed immediately by a RUN statement, so that all of the BASIC pointers are properly set. Failure to do so will cause the BASIC program to crash, possibly after running correctly for some time.

Example (32K PET):
```
10 SYS 31800:RUN 20
20 REM PROGRAM CONTINUES HERE
```

### RC (sys+3)

Places characters from the receive buffer into the string QR$. All characters received in the buffer since the last call are used unless the CI leadin flag is enabled. No receive buffer contents placed in QR$ on previous calls are placed there again unless the CI leadin flag is enabled. If no new characters came in since the previous call, then QR$ is returned null. If more than one character came in, then LEN(QR$)>1, and the characters in QR$ are stored left to right in the order they were received.

A simple BASIC loop which receives characters and places them in proper order on the screen is:

```
100 SYS RC :PRINT QR$; :GOTO100
```

### SE (sys+6)

Transmits all of the characters, leftmost first, in the string QS$ in asynchronous serial format as defined by the CI flag and the BR baud rate index. Translation options are noted in the CI flag documentation. If QS$ is null, nothing is done.

A simple BASIC loop which takes characters from the keyboard, displays them on the screen, and transmits them is:

```
100 GETA$ :IFA$=""THEN 100
110 PRINT A$;
120 QS$=A$:SYS SE
130 GOTO 100
```

XX (sys+9)

Disables serial transmission and reception by restoring the interrupt service vector to the value it had just before the last SYS(SE) call. If two SYS(SE) calls are done without a SYS(US) call between them, then these routines cannot be disabled, because the previous service vector is lost.

BK (sys+12)

Toggles the serial transmit line, that is, if it is marking, change it to spacing, and if it is spacing, change it to marking. Character transmission leaves the line marking. Thus, code which transmits a BREAK is:

```
100 SYS BK:T=TI
110 IF TI-T¼60 THEN 110
120 SYS BK
```

US (sys+15)

Bit5 of the CI location enables a mode where QR$, when it begins with a specified leadin character, is returned at each SYS(RC) call still starting at that leadin character, rather than starting as normally with the first fresh character not previously returned. If this is happening, then the SYS(US) call is needed to release the beginning of QR$ to the first character not previously returned.Otherwise, QR$ would grow indefinitely.

If CI bit5 is disabled, then SYS(RC) always resets the beginning of QR$, so that SYS(US) has no effect.

CC (sys+18)

Compute CRC on the string QC$, if LEN(QC$)½2, else no action. In normal usage, one has a string, and appends two CHR$(0) to it. SYS(CC) replaces the two characters by the CRC checksum value. A second SYS(CC) restores the two CHR$(0) characters.

One normally does that first call just before transmitting a string. If the receiving computer assembles the received string in its entirety and does a SYS(CC), the last two characters will be restored to CHR$(0). If they are something else, then the received string contains a transmission error, usually due to electrical noise. The receiver can then request a retransmission.

Although this procedure does not absolutely reliably catch all errors, it is very good. The error detection ability varies with the polynomial used. (See locations for flag and value passing).

AP (sys+21)

Translates string QR$ from seven bit ASCII to eight bit displayable PETASCII. Bit 7 of each character is ignored.

The following program loop receives ASCII characters as the receive portion of a simple, backspacing ASCII terminal:

```
100 SYS RC :SYS AP: PRINT QR$; :GOTO100
```

Translations are as follows:

```
ASCII C/R CHR$(13):=CHR$(13)+CHR$(145)
ASCII L/F CHR$(10):=CHR$(17)
ASCII B/S CHR$(8) :=CHR$(157)
```

Alphanumeric characters are translated to PETASCII. ASCII braces become PET graphics. The ASCII DLE and all other ASCII control characters are REMOVED FROM QR$.

Adjacent ASCII carriage returns (C/R) and linefeeds (L/F) are combined into single PET CHR$(13) characters, so that received page ejects in QR$ don't cause QR$ to become oversize on translation. Successive ASCII C/R characters are also combined, for the same reason.

If the translation causes QR$ to grow to a length over 255, then the first character LEFT$(QR$,1) will be returned with ASCII value 255, and QR$ is ruined.

Translation may be disabled by setting CI bit6.

PA (sys+24)

Translates string QS$ from eight bit PETASCII to seven bit ASCII. Bit 7 of each character is masked to zero.

Translations are as follows:

```
PETASCII CRSR DOWN CHR$(13):=CHR$(10)
PETASCII CRSR BACK CHR$(157):=CHR$(8)
PETASCII DELETE    CHR$( 20):=CHR$(127)
```

PETASCII uppercase is translated to ASCII uppercase. Pet upper/lowercase mode is assumed.

Translation may be disabled by setting CI bit6.

## LOCATIONS FOR FLAG AND VALUE PASSING

CI (base+0)

| | |
|---|---|
| Bit7 == 0 | End returned string QR$ on carriage return. |
| Bit7 == 1 | No effect. |
| Bit6 == 0 | No effect. |
| Bit6 == 1 | SYS(AP) and SYS(PA) translations are disabled. |
| Bit5 == 0 | No effect. |
| Bit5 == 1 | End returned string QR$ ahead of leadin character LC. |
| Bit4 == 0 | Stop key disabled and returns control-c (chr$(3)). |
| Bit4 == 1 | Stop key enabled and returns no character code. |
| Bit3 == 0 | Apply 7-bit mask to QR$ (return seven-bit characters). |
| Bit3 == 1 | Return full eight-bit characters in QR$. [illegible], POKE RI |
| Bit2 == 0 | Transmit seven-bit words. |
| [✓ Bit2 == 1] | Transmit eight-bit words. |
| Bit1 == 0 | No parity bit appended to transmitted word. |
| Bit1 == 1 | Parity bit appended to transmitted word. |
| Bit0 == 0 | Even parity (if enabled). |
| Bit0 == 1 | Odd parity (if enabled). |

Notes:

Enabling CI bit7 forces QR$ to end on every incoming carriage return, even though more characters are available. QR$ continues to end normally or be returned null if characters are not available. QR$ continues to start normally with the character following the last character in the previous QR$. This option permits one to test the incoming string for carriage returns using the form RIGHT$(QR$,1) without taking the time and code to test every imbedded character.

Enabling CI bit5 has three functions. FIRST,it forces QR$ to end on the character PRECEDING the specified LC leadin character. SECOND, once a QR$ is returned which begins with the LC, that is, following the QR$ with the forced ending, the beginning of QR$ is held at the LC. Each succeeding SYS(RC) returns a longer QR$, beginning at the LC, as more characters come in. This enables one to collect incoming escape sequences easily. THIRD, since the forced LC ending is enabled, the flag WF is set nonzero if the returned QR$ stops growing because another LC character is waiting.

To release the beginning of QR$ to the next previously unfetched character, use the SYS(US) call.

LC (base+1)

POKE the ASC value of the leadin character for escape sequence trapping here. This is usually the default ASCII ESC, which is CHR$(27).

WF (base+2)

PEEK the leadin character waiting flag here. Value is zero if no character is waiting, otherwise nonzero.

BR (base+3)

POKE the baud rate here, as follows:

| | |
|---|---|
| 75 baud: | POKE BR,0 |
| 110 baud: | POKE BR,6 |
| 300 baud: | POKE BR,12 |
| 600 baud: | POKE BR,18 |
| 1200 baud: | POKE BR,24 |

(base+4) and (base+5)

This is the CRC polynomial in 6502 (low-high) sequence. It is a 16-bit 6502 binary number representing the low order 16 terms of the CRC polynomial. The default value represents the IBM CRC16 polynomial. An X**16 most significant term is built into the CRC generation code.

OTHERS

Other memory locations are used internally and are not recommended for PEEKing and POKEing.