HOSTCM SPECIFICATIONS DOCUMENT

(Feb 2, 1982) by T. Wilkinson

Introduction

The Waterloo microSystems support a special device called "host" which is in fact a second computer running a program called HOSTCM. This program is required for easy transfer of files of data between the microcomputer and the host computer. Versions have been developed for both the IBM VM/CMS and the DEC RSTS/E systems. This document describes how HOSTCM works and provides enough information to facilitate development of versions for most other computer systems.

Implementing one of the distributed versions of HOSTCM is a relatively easy task but developing a new version requires an intimate knowledge of the file capabilities and the terminal handling features of the host machine. Communication is done via a full or half-duplex RS232C line running between the micro and the host. All messages are in ASCII code to allow data transparency through the more common host device handlers.

General Principles of Operation

In normal operation, the HOSTCM program is started on the host computer by simply executing it. This is done using a terminal or using the microcomputer in "passthru" mode (i.e., simple terminal emulation). Once HOSTCM is running, it is ready to receive requests from the microcomputer, act on them, and provide an appropriate response.

All messages between the two computers are standardized and contain check-sum characters which provide for data recovery in the event of line transmission errors.

We recommend that the HOSTCM implementor be familiar with the system and file concepts described in the Systems Overview Manual supplied with the particular Waterloo microSystems computer to be used.

General Message Format

All messages sent from the microcomputer to the host computer are in the following format:

message CS LE

where:

message -one of the standard operation requests described in the Command Summary section.

CS -the appropriate check-sum character. (not present with Quit and Negative Acknowledgement messages.)

LE -the specified "lineend" character (usually hex 0D).

On receiving a message from the microcomputer, the host computer responds with a 3-part message in the following format:

ignored RS message CS LE ignored PR

where:

message -one of the standard responses described in the Command Summary section.

CS -the appropriate check-sum character

LE -the specified "lineend" character (usually hex 0D).

RS -the specified "response" character

PR -the specified "prompt" character(s). (Up to 4 characters.)

ignored -characters appearing in this part of the transmission are ignored. (This may be null.)

This scheme of using a 3-part message is employed to provide compatibility with most common host terminal handlers which produce standard responses to input lines and prompts for more input.

Part 1 is usually generated by the host terminal handler in response to the "lineend" character being received.

Part 2 is produced by HOSTCM in reply to the request from the microcomputer.

Part 3 is usually generated by the host terminal handler to indicate that it is ready to receive another message.

NOTE: The characters RS ("response"), LE ("lineend") and PR ("prompt") are specified in the setup option of the Waterloo microSystems main menu. They are normally chosen to correspond with common terminal prompt characters generated by the host system's terminal handler.

The host computer terminal system settings must be such as to prevent an "idle" or "fill" character being inserted in the messages.

Typical Settings for VM/CMS are:

```
micro setup      PROMPT     11     (XON)
                 LINEEND    0D     (CR)
                 RESPONSE   13     (XOFF)

host terminal    SET BLIP OFF
                 CP TERM ESCAPE OFF
                 CP TERM LINEEND OFF
                 CP TERM LINEDEL OFF
                 CP TERM CHARDEL OFF
                 CP TERM TABCHAR OFF
                 CP SET MSG OFF
                 CP SET EMSG OFF
                 CP SET IMSG OFF
                 CP SET WNG OFF
```

Typical settings for RSTS/E are:

```
micro setup      PROMPT     0A     (LF)
                 LINEEND    0D     (CR)
                 RESPONSE   0A     (LF)

host terminal    SET FILL 0
                 SET LOCAL ECHO
```

The Check-sum Formula

The check-sum character is one of the first 16 letters of the alphabet chosen by adding up the ASCII equivalent codes of all the characters in "message" modulus 16. This number is used to index into the string 'ABCDEFGHIJKLMNOP'. The operation can be expressed as:

```
index <- (sum of ASCII characters) mod 16
CS    <- 'ABCDEFGHIJKLMNOP' [index]      (origin 0)
```

Transmission Error Handling

When either the microcomputer or HOSTCM sends a message, it appends a check-sum character computed by the method described above. Similarly, when either receives a message, it computes the check-sum character and compares the computed value with that received in the transmission. If these match, the transmission is assumed to have been error free. If, however, they do not match, the receiving computer sends the "negative acknowledgement" message which causes the sending computer to re-transmit the message. This retrying continues until a correct transmission with a correct check-sum is received. Implementors should note that, if the HOSTCM program is not running (or abnormally terminates) an infinite loop could result whereby the micro and the host each are sending negative acknowledgements.

Command Summary

The following pages contain a list of the valid commands which are sent from the microcomputer to the host computer. No blanks exist between the various operands unless specifically indicated and upper and lower case characters must be used as shown.

Explanations of the various concepts involved in this system are included in subsequent sections.

Open Request

```
        o fmode ftype blank fname

where:  fmode:  R - read
                W or w - write
                U or u - update
                A or a - append
                L or l - load file (binary)
                S or s - store file (binary)

        ftype:  b - binary file
                t - text file

        blank:  a single space character

        fname:  name of file to be opened

responses:
        b fnum
                where:  fnum:   - host-assigned file number
                                  (note 1,4)
        xInvalid open mode
        xInvalid open type
        xInvalid file name
        xExceeded maximum number of open files
        xInsufficient memory for file buffer
```

Get Request

```
        g fnum seq

where:  fnum:   host-assigned file number (note 1,4)
        seq:    1 - re-send previous partial/complete record
                (if this operand is omitted, the next
                partial/complete record is transmitted)

responses:
        bz data
                where:  data - all or last part of record (note 5)
        bn data
                where:  data - partial record (note 5)
        e       (meaning end-of-file)
        xInvalid file number
        xFile not open
        xFile not open for input or update
```

Put Request

```
        p fnum mode data

where:  fnum:   host-assigned file number (note 1,4)
        mode:   n - partial record
                z - all or last part of record
        data:   characters to be written to file (note 5)

responses:
        b       (meaning OK)
        xInvalid file number
        xFile not open
        xFile not open for output, update or append
        x..system msg..
```

Seek Request

```
        r fnum blank rnum

where:  fnum:   host-assigned file number (note 1,4)
        blank:  a single space character
        rnum:   number of the record to be "seeked" (note 1)

responses:
        b       (meaning OK)
        xInvalid file number
        xFile not open
        xFile not open for input or update
        xInvalid record number
```

Close Request

```
        c fnum

where:  fnum:   host-assigned file number (note 1,4)

responses:
        b       (meaning OK)
        xInvalid file number
        xFile not open
```

Close All Request

```
        a

responses:
        b       (meaning OK)
```

Attribute Request (note 3)

```
        ?

responses:
        b hsize , msize
                where:  hsize - host buffer size (note 1)
                        msize - micro buffer size (note 1)
```

Size Request

```
        v num

where:  num:    size of micro's buffer (note 1)

responses:
        b       (meaning OK)
        xInvalid buffer size
```

Directory Open

```
        d fname

where:  fname:  name of sub-directory

responses:
        b       (meaning OK)
        xDirectory file already open
        xNo files found
```

Directory Get

```
        f

responses:
        b data
                where:  data:   directory record
        e       (meaning end-of-file)
        xDirectory file not open
```

Directory Close

```
        k

responses:
        b       (meaning OK)
        xDirectory file not open
```

Scratch Request

```
        y fname

where:  fname:  file name to be scratched

responses:
        b       (meaning OK)
        xInvalid file name
        x..system msg..
```

Rename Request

```
        w fname

where:  fname:  name of file to be renamed

responses:
        b       (meaning send-new-file-name)
        xInvalid filename

        next message will be:

                b fname
                        where:  fname:  new name for file

                        responses:
                                b       (meaning OK)
                                xExpecting file name
                                xInvalid file name
                                x..system msg..
```

Mount Request (note 3)

```
        m dev

where:  dev:    name of device to be mounted

responses:
        b       (meaning OK)
        xInvalid file name
        x..system msg..
```

Dismount Request (note 3)

```
        u dev
```

where: dev: name of device to be dismounted

responses:

```
        b       (meaning OK)
        xInvalid file name
        x..system msg..
```

Quit Request (Note 2)

```
        q
```

responses:

```
        none  (HOSTCM terminated)
```

Negative Acknowledgement (Note 2)

```
        N
```

responses:

```
        retransmit last message
```

Note 1 All numbers are represented in decimal using the ASCII characters '0123456789'.

Note 2 No check-sum character is included in these messages. The lineend character is included.

Note 3 These requests are not implemented in the Commodore SuperPET or the Northern Digital MicroWAT microcomputers.

Note 4 File numbers must be one digit only.

Note 5 The "data" field can contain any character except the lineend character. However, it normally contains only the printable ASCII characters.

Error Message Philosophy

Many of the responses from the HOSTCM program are error diagnostics of the form:

x error text

The "error text" is system dependent text which is usually not analysed by the microcomputer software but displayed on the screen. This means that any desirable message can be sent by HOSTCM to the microcomputer thereby providing a large degree of flexibility to the implementor of a specialized HOSTCM program.

Filename Philosophy

A number of the messages and responses in this system contain the name of a file or file group. This filename and/or devicename facility is designed to handle the wide variety of syntax found in the various host computer systems. As a result, the filenames specified in user programs on the microcomputer are passed directly to the HOSTCM program with only the "host." prefix removed. For example, the name "host.DOCUMENT SCRIPT" would appear as "DOCUMENT SCRIPT" in a HOSTCM message. Similarly, "host.*.MAC" would appear as "*.MAC" to HOSTCM. This technique is intended to provide complete flexibility to the HOSTCM implementor facilitating access to the various host computer devices and files.

Filenames received at the host will sometimes be prefixed with file type information enclosed in parentheses. This is sometimes supplied by the programmer and sometimes is generated by the microcomputer software. The specific syntax and its meanings are described in the System Overview Manual for the particular microSystem and its treatment by HOSTCM should be consistent with this description.

File types: Text vs Binary

When a file is opened, it is specified as being either "text" or "binary". The differences between these two file types can be described as follows.

Text files are assumed to contain only printable ASCII characters. These are often files or documents created with an editor. HOSTCM transmits these files character by character and they must not contain the "prompt", "lineend" or "response" characters defined by setup.

Binary files are assumed to contain any of the 256 valid hexadecimal data bytes. In order to preserve data transparency across the system, these files are transmitted in "exploded" form. That is, each hexadecimal byte is translated into two ASCII characters in the set '0123456789ABCDEF'. HOSTCM will translate data received in this manner back to its normal form for storage on the host computer. Any files requested in this mode will be "exploded" by HOSTCM as they are transmitted and the microcomputer software will translate them back into normal form.

The Buffer Philosophy

Messages transmitted between the host and the micro are received into buffers on the respective machines. The micro informs the host of its buffer size before every Open Request via the Size Request. It assumes that the host's buffer is at least as long as the micro's buffer.

Each of the two buffers must be large enough to receive the entire message, including the request codes, data and the check-sum character.

Partial/Complete Record Philosophy

In the interest of memory conservation, the microcomputer has only one buffer supporting the "host" device and this buffer is a fixed and arbitrary size (usually 80 bytes). The size of records transmitted from the host to the microcomputer must not exceed the size of this buffer. If longer records must be sent, they are broken down into "partial" records. A parameter in the response to the "Get Request" designates whether a complete or a partial record is being sent. (The microcomputer informs the host computer of its buffer size via the "Size Request" message.)

The single buffer technique employed here sometimes requires that a record which has been sent to the microcomputer be re-sent at a later time. The "seq" parameter of the "Get Request" message is used to indicate whether the "current" or "next" partial/complete record is to be transmitted. This situation only arises when processing is being done at the character level (e.g., the BASIC GET statement) and two or more host files are open at the same time.

APPENDIX A

EXAMPLE 1:

The following example illustrates the message transmissions involved when a program on the microcomputer reads records from a sequential file named "testfile script" on the host computer. Three complete records (2 with fewer than 80 bytes and 1 with more than 80 bytes) are transmitted before end-of-file is encountered. (The "prompt", "response", "lineend" and check-sum characters are not shown.

operation	Micro -> Host (request)	Host -> Micro (reply)
open file	v80	b
	ort testfile script	b1
read record 1	g1	bzrec1...
read record 2	g1	bzrec2...
read record 3	g1	bnrec3(part)...
read rest of 3	g1	bzrec3(rest)...
read record 4	g1	e
close file	c1	b

EXAMPLE 2:

The following example illustrates the message transmissions involved when the microcomputer requests that a host file named "testfile script" be renamed to "oldfile script". (The "prompt", "response", "lineend" and check-sum characters are not shown).

operation	Micro -> Host (request)	Host -> Micro (reply)
specify old name	wtestfile script	b
specify new name	boldfile script	b