To: Distribution

From: Suzanne L. Krupp

Date: March 31, 1982

Subject: Multics - Level 6 RBF connection via X.25

INTRODUCTION

This MTB describes the work that needs to be done on Multics to set up a communications link between Multics and the Level 6 Remote Batch Facility (L6 RBF) over X.25.

WHAT WE HAVE NOW

Right now, there is only one type of communications link supported between Multics and L6 RBF. This link is over a communications channel that uses the Remote Computer Interface (RCI) protocol for data transmission.

There is another protocol used by Multics and L6 RBF to carry out the RBF functions. This is the Remote Batch Facility (RBF) protocol.

Both of these protocols are now implemented partially in the g115_ I/O module and partially in the Front-end Network Processor (FNP). This mixture of protocols makes it difficult for Multics to communicate with L6 RBF over any other type of link (see Figure 1).

WHAT WE WANT TO HAVE

What we want is to be able to choose the type of communications link between Multics and the L6 RBF. A different type of link between Multics and L6 RBF is over a network using the X.25 protocol. To accomplish this, we need to restructure the programs which handle these communications. The protocols (RCI and RBF) which have been mixed together must be separated out into layers. The RBF protocol would be in the top layer. The next layer would consist of whatever communications protocol we choose, RCI or X.25. This arrangement is shown in Figure 2.

Modules Affected

There is a new I/O module called rbf_. It works the same as the g115_ I/O module with the following exceptions:

1. The communications I/O module may now be specified via the -comm control argument. It can be either rci_ or tty_.

2. There is additional functionality that handles $*$ control cards. This functionality is duplicated here from the G115 tables in the FNP so that $*$ cards may be centrally processed (by rbf_ rather than in several places throughout the FNP).

3. The rbf_ I/O module implements record oriented I/O that is the same as the hasp_workstation_ I/O module.

The remote_driver_ I/O modules that call rbf_ do not need to be changed. They already support this type of record I/O.

A new I/O module, called rci_, implements the RCI protocol. A typical message block transmitted by rci_ is shown in Figure 3 below.

There will be new code in the FNP (RCI tables) that does the same job as the G115 tables minus the $*$ control card functionality. This means that there will also be a new line type called RCI.

The g115_ I/O module, G115 FNP tables, and G115 line type will stay around for compatiblity.

Documentation and SRB Notice

The documentation for rbf_ and rci_ (shown below) should be added to CC92.

The documentation in CC75 (page 2-2) should be changed to add the RCI line type to the list of line types provided by the Multics Communication System The description of the RCI line type is essentially the same as the description of the G115 line type.

The following should be in the MR10 SRB notice.

1. Any attach description in the iod tables for a remote_driver_ I/O module that specifies "-terminal g115_" should now specify "-terminal rbf_".

2. Any attach description in the iod tables for g115_ that correspond to item 1 above should specify "-comm STR" where STR is either rci_ or tty_.

Testing

Testing will be accomplished by setting up both an RCI and an X.25 link between the CISL Level 6 and System M. All new and existing functionality will be tested.

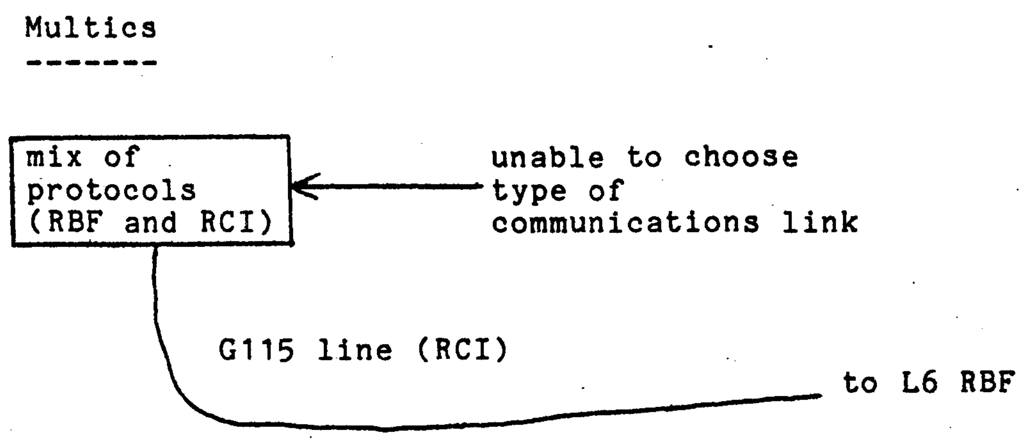
Multics
mix of protocols (RBF and RCI)
unable to choose type of communications link
G115 line (RCI)
to L6 RBF

Figure 1. Current Arrangement

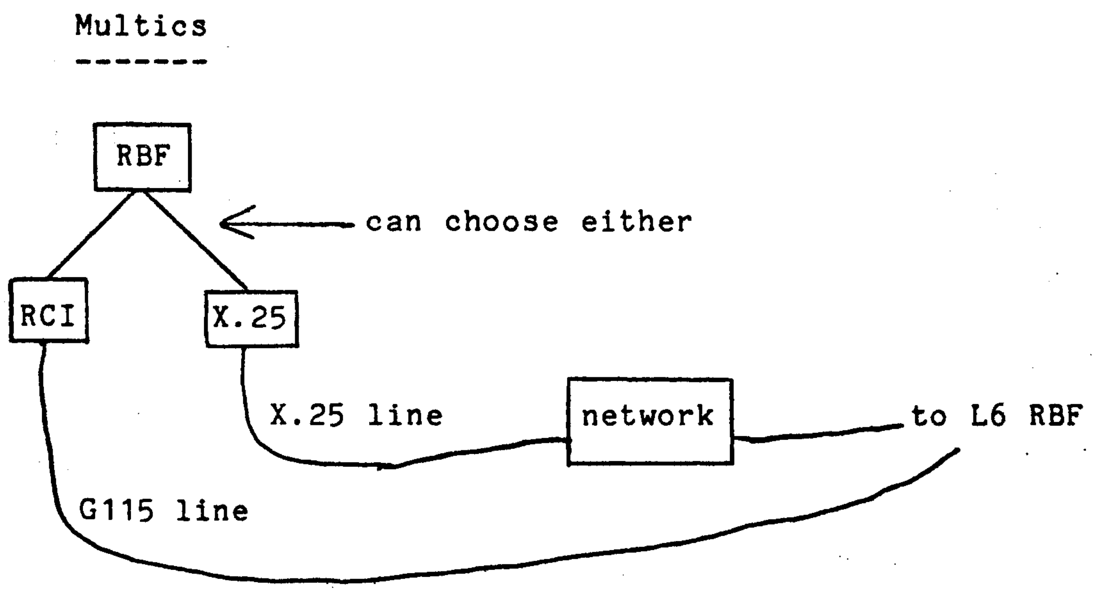

Figure 2. New Arrangement

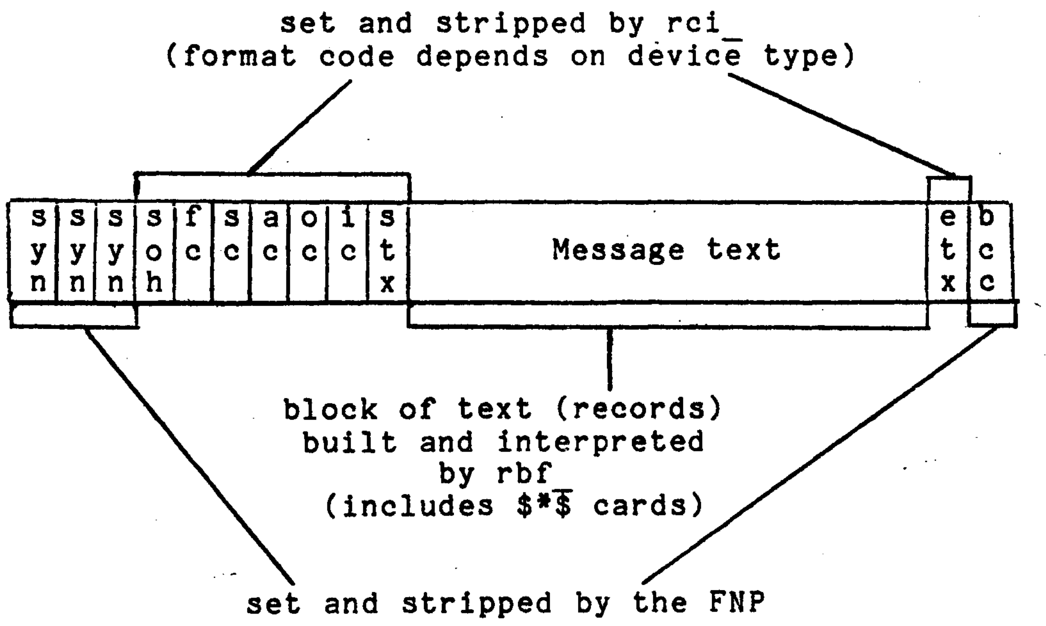

Figure 3. Typical Message Block

syn	-- synchronization character
soh	-- start of header
fc	-- format code
sc	-- sequence code
ac	-- address code
oc	-- operations code
ic	-- identification code
stx	-- start of text
etx	-- end of text
bcc	-- block check character

Name: rbf_

The rbf_ I/O module performs record oriented I/O to a remote I/O terminal that has the characteristics of the Honeywell Level 6 remote batch facility. The hardware options currently supported are defined by the control arguments described below.

Entry points in this module are not called directly by users; rather, the module is accessed through the I/O system.

Attach Description

```
rbf_ -control_args
```

where control arguments may be chosen from the following and are optional with the exception of -device, -comm and -tty:

-device STR
: attaches the subdevice specified by STR. STR may be printer, punch, reader, or teleprinter.

-auto_call N
: specifies the phone number, N, to be called via the auto call unit on the specified communications channel.

-tty STR
: connects the remote I/O terminal to the logical communications channel named STR.

-comm STR
: uses the communications I/O module specified by STR. STR may be either tty_ or rci_.

-ascii
: uses the ASCII character set. This is the default. This argument is accepted for compatibility with other terminal I/O modules.

-physical_line_length N, -pll N
: specifies the physical line length, N, of the output device. This argument is accepted for compatibility with other terminal I/O modules.

-terminal_type STR, -ttp STR
: STR specifies the terminal type whose conversion, translation, and special tables defined in the user

or system terminal type table (TTT) are used to convert and translate input and output to and from the device. If not specified, no conversion or translation is performed. For more information about the allowable conversion values see "Notes" below.

Open Operation

The rbf_ I/O module supports the sequential_input, sequential_output, and sequential_input_output opening modes.

Write Record Operation

The write_record entry performs the appropriate conversion and translation on the data record, converts the supplied slew control into the proper carriage control sequences for line printer attachments and performs data compression. The records are put into blocks of up to 324 characters and transmitted to the specified communications channel.

The format of the record supplied to this I/O module follows. This structure and the referenced constants are contained in the terminal_io_record include file:

```
     dcl 1 terminal_io_record aligned based,
           2 version fixed binary,
           2 device_type fixed binary,
           2 slew_control,
             3 slew_type fixed binary (18) unaligned unsigned,
             3 slew_count fixed binary (18) unaligned unsigned,
           2 flags,
             3 binary bit (1) unaligned,
             3 preslew bit (1) unaligned,
             3 pad bit (34) unaligned,
           2 element_size fixed binary,
           2 n_elements fixed binary (24),
           2 data,
             3 bits (terminal_io_record_n_elements refer
                         (terminal_io_record.n_elements))
                 bit (terminal_io_record_element_size refer
                         (terminal_io_record.element_size))
      unaligned;
```

where:

version (Input)
is the current version of this structure. This version of the structure is given by the value of the named constant terminal_io_record_version_1.

device_type (Input)
is the type of device to which this record it to be written. The acceptable values are TELEPRINTER_DEVICE, PRINTER_DEVICE, or PUNCH_DEVICE.

slew_control (Input)
need only be supplied by the caller if device_type is PRINTER_DEVICE and specifies the slew operation to be performed after printing the data in the record.

slew_type (Input)
specifies the type of slew operation. The possible values are SLEW_BY_COUNT, SLEW_TO_TOP_OF_PAGE, SLEW_TO_INSIDE_PAGE, SLEW_TO_OUTSIDE_PAGE, or SLEW_TO_CHANNEL.

slew_count (Input)
is interpreted according to the value of slew_control.slew_type.

flags.binary (Input)
must be set to "0"b. (This I/O module does not support binary data transmission.)

flags.preslew (Input)
must be set to "0"b. (This I/O module does not support slew operations before printing the record's data.)

element_size (Input)
must be set to 9. (This I/O module only supports transmission of characters.)

n_elements (Input)
is the number of characters in the record to be written.

data.bits (Input)
is the actual data.

Read Record Operation

The read_record entry reads blocks of up to 324 characters and returns a single record from the device, basically performing the inverse of the functions described for the write_record operation.

The format of the record this I/O module returns in the supplied buffer follows. This structure and the referenced constants are contained in the terminal_io_record include file:

```
dcl 1 terminal_io_record aligned based,
      2 version fixed binary,
      2 device_type fixed binary,
      2 slew_control,
        3 slew_type fixed binary (18) unaligned unsigned,
        3 slew_count fixed binary (18) unaligned unsigned,
      2 flags,
        3 binary bit (1) unaligned,
        3 preslew bit (1) unaligned,
        3 pad bit (34) unaligned,
      2 element_size fixed binary,
      2 n_elements fixed binary (24),
      2 data,
        3 bits (terminal_io_record_n_elements refer
                    (terminal_io_record.n_elements))
            bit (terminal_io_record_element_size refer
                    (terminal_io_record.element_size))
 unaligned;
```

where:

version (Output)
is the current version of this structure. This version of the structure is given by the value of the named constant terminal_io_record_version_1.

device_type (Output)
is the type of device from which this record was read. Its possible values are TELEPRINTER_DEVICE or READER_DEVICE.

slew_control.slew_type (Output)
is always set to SLEW_BY_COUNT.

slew_control.slew_count (Output)
is always set to 1.

flags.binary (Output)
    is always set to "0"b.

flags.preslew (Output)
    is always set to "0"b.

element_size (Output)
    is always set to 9.

n_elements (Output)
    is set to the number of characters returned in the record.

data.bits (Output)
    is the actual returned data.

Control Operation

This I/O module supports all the control operations supported by the tty_ I/O module. In addition, it supports the following:

select_device
    selects the subdevice, either printer, punch, or teleprinter, to which output is next directed. The input structure is of the form:

```
dcl device char(32);
```

runout
    transmits any data stored in the output buffer. There is no input structure.

hangup_proc
    sets up a specified event call channel to be signalled over, and a procedure to be called, if the communications channel hangs up. The hangup_proc structure has the following form:

```
dcl 1 hangup_proc aligned,
      2 entry entry variable,
      2 datap ptr,
      2 prior fixed bin;
```

where:

entry
    is the entry to call when a hangup is detected.

datap
: is a pointer to data for the hangup procedure.

prior
: is the ipc_ event call priority to be associated with hangup notification.

reset
: sets the edited mode of output conversion. There is no input structure.

end_write_mode
: prevents the rbf_ module from returning until all outstanding output has been written to the attached channel. There is no input structure.

Modes Operation

This I/O module supports the rawi and rawo modes. It also supports the nonedited and default modes, which set and reset the edited output conversion, if it has been enabled by the -terminal_type control argument.

Notes

The only allowable values in the output conversion table are 00 and any values greater than 16. All values defined in the description of the tty_ I/O module are allowed for input conversion. Input and output translation tables may be up to 256 characters in length.

Name: rci_

The rci_ I/O module performs stream I/O over a G115 communications channel using the Remote Computer Interface Protocol.

Entry points in this module are not called directly by users; rather, the module is accessed through the I/O system.

Attach Description

```
     rci_ -control_args
```

where control arguments may be chosen from the following and are optional with the exception of -device and -tty:

-auto_call N
: specifies the phone number, N, to be called via the auto call unit on the specified communications channel.

-device STR
: attaches the subdevice specified by STR. STR may be printer, punch, reader, or teleprinter.

-tty STR
: connects the remote I/O terminal to the communications channel named STR.

Open Operation

The rci_ I/O module supports stream_input, stream_output, and stream_input_output opening modes.

Put Chars Operation

The put chars entry can transmit blocks of up to 324 characters to the specified communications channel.

The Get Chars Operation

The get chars entry reads blocks of up to 324 characters from the specified communications channel.

Control Operation

The following control operations are supported by this I/O module:

select_device
: selects the subdevice, either printer, punch, or teleprinter, to which output is next directed. The input structure is of the form:

```
dcl device char(32);
```

hangup_proc
: sets up a specified event call channel to be signalled over, and a procedure to be called, if the communications channel hangs up. The hangup_proc structure has the following form:

```
dcl 1 hangup_proc aligned,
      2 entry entry variable,
      2 datap ptr,
      2 prior fixed bin;
```

where:

entry
: is the entry to call when a hangup is detected.

datap
: is a pointer to data for the hangup procedure.

prior
: is the ipc_ event call priority to be associated with hangup notification.

Modes Operation

This I/O module does not support the modes operation.