To: Distribution

From: Robert S. Coren

Date: 02/13/81

Subject: Communications Metering Interfaces

At the design review on MTB 457, "Communications Metering," it was agreed that a set of subroutine interfaces should be provided to return raw communications meters on a per-channel basis to allow commands to select channels and display statistics according to their own criteria. (See MTR 164.) Some of the data returned varies from one multiplexer to another; therefore, in addition to a subroutine to return metering information for selected channels, some generic subroutine interfaces are needed to fill in and display multiplexer-specific meters, on the model of the interfaces used by tty_dump and tty_analyze.

This MTB provides documentation for a subroutine named comm_meters_, which returns meters for a list of communications channels, and for two generic subroutine interfaces for use in connection with individual multiplexer types: get_MPX_meters_, which is called by comm_meters_ on a channel-by-channel basis to fill in multiplexer-specific meters, and display_MPX_meters_, which can be called by metering commands to display multiplexer-specific statistics. A gate entry, metering_ring_zero_peek_$get_comm_meters, is provided for use by the various get_MPX_meters_ subroutines to get meters from ring 0; a similar entry is provided in phcs_. Two gate entries are provided in order to allow a distinction to be made, at a future time, between access required to get metering information for the user's own channel and that required to get information for any channel. No such distinction is included in the present proposal. In addition, two control orders to MCM are provided: copy_meters, which is used by the answering service at dialup time to save the cumulative meters through the previous dialup, and get_meters, which is used internally by the get_comm_meters gate entries. More information on the use of these two orders is provided under "Implications for Multiplexers," below.

System-wide meters are maintained in the header of tty_buf. Commands and subroutines that are concerned with system-wide communications meters should copy tty_buf itself using ring_zero_peek_. The format of the tty_buf header is defined in tty_buf.incl.pl1.

Although the subroutine interfaces described herein allow sites, users, developers, etc. to design arbitrary communications metering commands, it is also desirable that the system provide some basic commands for the display of communications meters. Two such commands were proposed in MTB 457; the present document contains a revised version of this proposal.

IMPLICATIONS FOR MULTIPLEXERS

Any particular multiplexer may maintain meters specific to the multiplexer type for the multiplexed channel itself and/or for its subchannels. Accordingly, there are three types of meters potentially associated with any logical channel known to MCM: common meters maintained by channel_manager for all logical channels, hereafter referred to as "logical channel meters"; meters maintained by the multiplexer for the channel itself; and meters maintained on its behalf by its parent multiplexer. All of these meters must be obtainable by means of the get_meters control order. The following rules therefore apply:

o -- The priv_control entry of every multiplexer must support the copy_meters order. It must forward the order to the next level (unless it is a level-1 multiplexer) by calling channel_manager$control with a control type of "copy_meters".

o -- The priv_control entry of every multiplexer must support the get_meters order. It must forward the order to the next level as described above; this permits channel_manager to fill in logical channel meters and the parent multiplexer to fill in any meters that it maintains on behalf of the subchannel.

o -- The control entry of every multiplexer that maintains meters on behalf of its subchannels must support the copy_meters order and the get_meters order. These orders should not be forwarded.

o -- Every multiplexer that supports the copy_meters order is responsible for allocating space (preferably unwired) for the copied meters of its subchannels at the time that it initializes multiplexer-specific data bases, and for freeing such space at multiplexer shutdown.

o -- The answering service issues a copy_meters order on a non-multiplexed channel immediately before assigning it to a process. It makes a priv_control call with a control type of "copy_meters" on a multiplexed channel immediately after loading the multiplexer.

Name: comm_meters_

The comm_meters_ subroutine, given a list of communications channel names, returns metering information for all the specified channels. The exact information returned for each channel varies depending on the line type and multiplexer type of the channel. Callers of comm_meters_ should later call the comm_meters_$free entry point to release the space allocated for the returned metering information.

Usage

```
        dcl comm_meters_ entry ((*) char (32), fixed bin,
                 pointer, fixed bin, pointer, fixed bin (35));

        call comm_meters_ (chan_names, version, area_ptr,
                 n_channels, chan_meters_ptr, code);
```

chan_names
is an array of channel names, any of which may be starnames. (Input)

version
is the version number of the channel_meters structure to be returned. It must be 1. (Input)

area_ptr
is a pointer to an area in which the returned metering information is to be allocated. (Input)

n_channels
is the number of channels for which metering information is returned. (Output)

chan_meters_ptr
is a pointer to a linked list of structures containing the returned metering information. (Output)

code
is a standard system status code. (Output)

The structure pointed to by chan_meters_ptr has the following format:

```
dcl 1 channel_meters aligned based (chan_meterp),
      2 version fixed bin,
      2 multiplexer_type fixed bin,
      2 line_type fixed bin,
      2 flags,
        3 reserved bit (36) unaligned,
      2 channel_name char (32),
      2 mpx_specific_meterp pointer,
      2 physical_channel_meterp pointer,
      2 next_channelp pointer,
      2 last_dialup_time fixed bin (71),
      2 since_bootload,
        3 unconverted_input_chars fixed bin (35),
        3 converted_input_chars fixed bin (35),
        3 unconverted_output_chars fixed bin (35),
        3 converted_output_chars fixed bin (35),
        3 read_calls fixed bin,
        3 write_calls fixed bin,
        3 control_calls fixed bin,
        3 software_interrupts fixed bin,
        3 read_call_time fixed bin (71),
        3 write_call_time fixed bin (71),
        3 control_call_time fixed bin (71),
        3 interrupt_time fixed bin (71),
        3 chars_passed_input_interrupt fixed bin (35),
        3 pad (4) fixed bin,
      2 since_dialup like channel_meters.since_bootload;
```

version
    contains the value of the version argument (above).

multiplexer_type
    is the multiplexer type of the channel. It may have any of the values defined in multiplexer_types.incl.pl1.

line_type
    is the line type of the channel. It may have any of the values defined in line_types.incl.pl1.

flags
    are reserved for future use.

channel_name
: is the name of the channel.

mpx_specific_meterp
: is a pointer to additional meters that vary according to multiplexer type. Meters for FNPs and for non-multiplexed ("tty") channels are described below.

physical_channel_meterp
: is a pointer to additional meters for a physical channel (i.e., a direct subchannel of an FNP). If the channel is not a physical channel, this pointer is null.

next_channelp
: is a pointer to the channel_meters structure for the next channel in the list. If this is the last channel, next_channelp is null.

last_dialup_time
: is the clock time of the most recent dialup of the channel.

since_bootload
: contains meters for the channel accumulated since the most recent bootload of the system.

unconverted_input_chars
: is the number of characters input on the channel before conversion at the channel's multiplexing level.

converted_input_chars
: is the number of characters input on the channel after conversion.

unconverted_output_chars
: is the number of characters output on the channel before conversion at the channel's multiplexing level.

converted_output_chars
: is the number of characters output on the channel after conversion.

read_calls
: is the number of calls to channel_manager$read for this channel.

write_calls
is the number of calls to channel_manager$write for this channel.

control_calls
is the number of calls to channel_manager$control for this channel.

software_interrupts
is the number of calls to channel_manager$interrupt for this channel.

read_call_time
is the amount of time (in microseconds) spent in read calls.

write_call_time
is the amount of time spent in write calls.

control_call_time
is the amount of time spent in control calls.

interrupt_time
is the amount of time spent processing software interrupts.

chars_passed_input_interrupt
is the total number of characters passed with accept_input interrupts.

since_dialup
contains meters accumulated since the channel last dialed up (i.e., since last_dialup_time).

The structure pointed to by physical_channel_meterp has the following format:

```
dcl 1 physical_channel_meters aligned based (pcm_ptr),
    2 version fixed bin,
    2 dia_request_q_len fixed bin (35),
    2 dia_rql_updates fixed bin (35),
    2 pending_status fixed bin (35),
    2 pending_status_updates fixed bin (35),
    2 flags,
      3 synchronous bit (1) unaligned,
      3 reserved bit (35) unaligned,
2 since_fnp_load,
  3 output_overlaps fixed bin,
  3 software_status_overflows fixed bin,
  3 hardware_status_overflows fixed bin,
  3 input_alloc_failures fixed bin,
  3 sync_or_async (16) fixed bin,
2 since_dialup like physical_channel_meters.since_fnp_load;
```

version
    must be 1.

dia_request_q_len
    is the cumulative length of the channel's DIA request queue.

dia_rql_updates
    is the number of times dia_request_q_len has been updated.

pending_status
    is the cumulative length of the software status queue (for HSLA channels only).

pending_status_updates
    is the number of times pending_status has been updated.

synchronous
    is "1"b for a synchronous channel or "0"b for an asynchronous channel.

since_fnp_load
    contains meters for the channel accumulated since the FNP was last loaded.

output_overlaps
    is the number of times output arriving in the FNP has been added to a currently active output chain.

software_status_overflows
    is the number of times the software status queue has overflowed (for HSLA channels only).

hardware_status_overflows
    is the number of times the hardware status queue has overflowed (for HSLA channels only).

input_alloc_failures
is the number of times an attempt to allocate an input buffer for the channel has failed.

sync_or_async
is space for meters (described below) that vary depending on whether the channel is synchronous or asynchronous.

since_dialup
contains meters accumulated since the channel last dialed up (i.e.since channel_meters.last_dialup_time).

The following structure describes the meters for synchronous channels that appear in sync_or_async (above):

```
dcl 1 sync_channel_meters based aligned,
     2 input,
       3 message_count fixed bin (35),
       3 cum_length fixed bin (35),
       3 min_length fixed bin,
       3 max_length fixed bin,
     2 output like sync_channel_meters.input,
     2 counters (8) fixed bin;
```

input
contains statistics for input messages.

message_count
is the number of messages.

cum_length
is the cumulative length (in characters) of all messages.

min_length
is the length (in characters) of the shortest message.

max_length
is the length (in characters) of the longest message.

output
contains statistics for output messages.

counters
contain counts of up to 8 types of events metered for the channel (e.g., errors of various kinds). The meaning of each type depends on the line type and protocol being used on the channel.

The following structure describes the meters for asynchronous channels that appear in sync_or_async (above):

```
dcl 1 async_channel_meters based aligned,
    2 pre_exhaust fixed bin,
    2 exhaust fixed bin,
    2 echo_buf_overflows fixed bin,
    2 software_xte fixed bin,
    2 bell_quits fixed bin,
    2 pad (11) fixed bin;
```

pre_exhaust
is the number of times "pre-exhaust" status has occurred.

exhaust
is the number of times "exhaust" has occurred.

echo_buf_overflows
is the number of times the channel's echo buffer has overflowed.

software_xte
is the number of times "transfer timing error" status has been generated because an input ICW could not be refreshed in time.

bell_quits
is the number of times a BEL character has been output and a line break simulated on the channel because of exhaust or transfer timing error status.

If the channel is an FNP, channel_meters.mpx_specific_meterp points to a structure of the following form:

```
dcl 1 fnp_wide_meters based (fnp_meterp) aligned,
    2 version fixed bin,
    2 channels_dialed_cum fixed bin (35),
    2 channels_dialed_updates fixed bin (35),
    2 space_available_cum fixed bin (35),
    2 space_available_updates fixed bin (35),
    2 space_alloc_failures fixed bin,
    2 abnormal_dia_status fixed bin,
    2 input_mbx_in_use_cum fixed bin (35),
    2 input_mbx_updates fixed bin (35),
    2 output_mbx_in_use_cum fixed bin (35),
    2 output_mbx_updates fixed bin (35),
    2 output_mbx_unavailable fixed bin (35),
    2 max_output_mbx_in_use fixed bin,
    2 queue_entries_made fixed bin (35),
    2 input_rejects fixed bin,
    2 processed_from_q fixed bin (35),
    2 fnp_channel_locked fixed bin (35),
    2 input_data_transactions fixed bin (35),
    2 output_data_transactions fixed bin (35),
    2 input_control_transactions fixed bin (35),
    2 output_control_transactions fixed bin (35),
    2 fnp_space_restricted_output fixed bin,
    2 fnp_mem_size fixed bin,
    2 interrupts_from_fnp fixed bin (35),
    2 interrupt_time fixed bin (71);
```

version
must be 1.

channels_dialed_cum
is the cumulative number of channels dialed.

channels_dialed_updates
is the number of times channels_dialed_cum has been updated.

space_available_cum
is the cumulative total of the number of words of free space in the FNP.

space_available_updates
is the number of times space_available_cum has been updated.

space_alloc_failures
is the number of times an attempt to allocate space in the FNP failed.

abnormal_dia_status
is the number of times abnormal status was returned from a connect to the DIA.

input_mbx_in_use_cum
is the cumulative number of inbound (FNP-to-CS) mailboxes in use.

input_mbx_updates
is the number of times input_mbx_in_use_cum has been updated.

output_mbx_in_use_cum
is the cumulative number of outbound (CS-to-FNP) mailboxes in use.

output_mbx_updates
is the number of times output_mbx_in_use has been updated.

output_mbx_unavailable
is the number of times no outbound mailbox was available when one was needed.

max_output_mbx_in_use
is the largest number of outbound mailboxes ever in use at once.

queue_entries_made
is the number of times an entry was added to the delay queue for outbound mailbox transactions.

input_rejects
is the number of times the CS rejected input from the FNP because insufficient space was available in tty_buf.

processed_from_q
is the number of times dn355 has processed a queued interrupt from the FNP before unlocking the FNP channel lock.

fnp_channel_locked
is the number of times dn355$interrupt has found the FNP channel lock to be locked.

input_data_transactions

is the number of transactions initiated by this FNP to send data to the CS.

output_data_transactions

is the number of transactions initiated by the CS to send data to this FNP.

input_control_transactions

is the number of transactions initiated by this FNP to send control information to the CS.

output_control_transactions

is the number of transactions initiated by the CS to send control information to this FNP.

fnp_space_restricted_output

is the number of times the CS sent less output to the FNP than was available because insufficient FNP space was available.

fnp_mem_size

is the number of 18-bit words configured in this FNP's memory.

interrupts_from_fnp

is the number of interrupts that have been received from this FNP.

interrupt_time

is the total amount of time, in microseconds, that has been spent handling interrupts from this FNP.

If the channel is non-multiplexed, channel_meters.mpx_specific_meterp points to a structure of the following form:

```
dcl 1 tty_channel_meters aligned based (tty_meterp),
      2 version fixed bin,
      2 pad fixed bin,
      2 since_mpx_load,
        3 read_calls fixed bin (35),
        3 write_calls fixed bin (35),
        3 read_chars fixed bin (35),
        3 write_chars fixed bin (35),
        3 read_time fixed bin (71),
         3 write_time fixed bin (71),
         3 pad2 (2) fixed bin,
       2 since_dialup like tty_channel_meters.since_mpx_load;
```

version
must be 1.

since_mpx_load
contains meters accumulated since the channel's parent multiplexer was last loaded.

read_calls
is the number of calls to all entries of tty_read.

write_calls
is the number of calls to all entries of tty_write.

read_chars
is the total number of characters returned by tty_read (after conversion).

write_chars
is the total number of characters processed by tty_write (before conversion).

read_time
is the amount of time (in microseconds) spent in tty_read.

write_time
is the amount of time (in microseconds) spent in tty_write.

since_dialup
contains meters accumulated since the channel last dialed up (i.e., since channel_meters.last_dialup_time).

Entry: comm_meters_$free

This entry is called to release space allocated by comm_meters_ to return metering information. Any program that calls comm_meters_ should subsequently call comm_meters_$free to release the allocated space.

Usage

```
          dcl comm_meters_$free  entry  (pointer,  pointer, fixed
                    bin (35));

          call comm_meters_$free    (area_ptr,   chan_meters_ptr,
                    code);
```

area_ptr
: is a pointer to the area in which the space was allocated. (Input)

chan_meters_ptr
: is a pointer to the list of metering structures returned by comm_meters_ (above). (Input)

code
: is a standard system status code. (Output)

Name: get_MPX_meters_

This documentation describes the calling sequence of a collection of subroutines named get_MPX_meters_, where MPX is the name of a multiplexer type defined in multiplexer_types.incl.pl1. These subroutines are called by comm_meters_ to provide multiplexer-specific metering data for a specified communications channel of the appropriate multiplexer type. Any caller of such a subroutine should subsequently call get_MPX_meters_$free to release the space allocated by get_MPX_meters.

Usage

```
dcl get_MPX_meters_ entry (char (*), fixed bin,
        pointer, pointer, fixed bin (35));

call get_MPX_meters_ (chan_name, version, area_ptr,
        meter_ptr, code);
```

chan_name
is the name of the communications channel for which meters are to be returned. (Input)

version
is the version number of the metering structure to be returned. Its value depends on the multiplexer type. (Input)

area_ptr
is a pointer to an area in which the multiplexer-specific metering structure is to be allocated. (Input)

meter_ptr
is a pointer to the meters for the specified channel. The format of the meters pointed to by meter_ptr depends on the multiplexer type. (Output)

code
is a standard system status code. (Output)

Entry: get_MPX_meters_$free

Each get_MPX_meters_ subroutine has an entry, described here, that is called in order to free the metering structure allocated by the subroutine.

Usage

```
dcl get_MPX_meters_$free entry (pointer, pointer, fixed
          bin (35));

call get_MPX_meters_$free (area_ptr, meter_ptr, code);
```

area_ptr
is a pointer to the area in which the metering structure was allocated. (Input)

meter_ptr
is a pointer to the structure to be freed. (Input)

code
is a standard system status code. (Output)

This documentation describes the calling sequence of a collection of subroutines named display_MPX_meters_, where MPX is the name of a multiplexer type defined in multiplexer_types.incl.pl1. Each such subroutine displays multiplexer-specific statistics for a specified communications channel on a specified I/O switch. The format of the statistics displayed depends on the type of multiplexer. These subroutines are called by commands that display general communications meters.

Usage

```
dcl display_MPX_meters_ entry (char (*), pointer,
        pointer, fixed bin (35));

call display_MPX_meters_ (chan_name, iocb_ptr,
        meter_ptr, code);
```

chan_name
    is the name of the channel for which staistics are to be displayed. (Input)

iocb_ptr
    is a pointer to the I/O control block for the I/O switch on which the meters are to be displayed. If it is null, the user_output switch is used. (Input)

meter_ptr
    is a pointer to the raw metering data for the channel. The format of this data depends on the multiplexer type. (Input)

code
    is a standard system status code. (Output)

Entry: phcs_$get_comm_meters

This entry is used to copy communications metering information for a specified channel from ring 0. Logical channel meters for the specified channel are returned, as are any mutliplexer-specific meters maintained for the channel by its own multiplexer module or that of its parent.

Usage

```
dcl  phcs_$get_comm_meters  entry  (char  (*), pointer,
          fixed bin (35));

call phcs_$get_comm_meters (chan_name, info_ptr, code);
```

chan_name
is the name of the channel. (Input)

info_ptr
is a pointer to a structure of the same form as that described for the get_meters control order described later in this document. (Input)

code
is a standard system status code. (Output)

mrzp$get_comm_meters mrzp$get_comm_meters

Entry: metering_ring_zero_peek_$get_comm_meters

This entry is identical in function to phcs_$get_comm_meters; it exists for hte use of callers who lack access to the phcs_ gate. The arguments are the same as for phcs_$get_comm_meters.

copy_meters
causes the current cumulative meters associated with the channel to be copied to unwired storage, so that the statistics for the channel can be determined both for the life of the system and for the current dialup. This order can only be issued by the "owning" process (normally the initializer). The info_ptr should be null.

get_meters
causes current values of meters associated with the channel to be returned. The info_ptr must point to a structure of the following form:

```
dcl 1 get_comm_meters_info aligned based,
    2 version fixed bin,
    2 pad fixed bin,
    2 subchan_ptr pointer,
    2 logical_chan_ptr pointer,
    2 parent_ptr pointer,
    2 subchan_type fixed bin,
    2 parent_type fixed bin;
```

version
must be 1. (Input)

subchan_ptr
is a pointer to a structure in which multiplexer-specific meters kept at the subchannel level are to be returned. The format of this structure depends on the channel type as specified by subchan_type (see below). If no meters are kept for this channel type, then subchan_ptr may be null. (Input)

logical_chan_ptr
is a pointer to a structure in which logical channel meters (those maintained for every logical channel) are to be returned. The format of this structure is described below. (Input)

parent_ptr
is a pointer to a structure in which multiplexer-specific meters maintained by the channel's parent multiplexer are to be returned. The format of this structure depends on the

channel type as specified by parent_type (see below). (Input)

subchan_type
is the channel type of the channel. It may have any of the values described in multiplexer_types.incl.pl1. (Output)

parent_type
is the channel type of the channel's parent multiplexer. It may have any of the values described in multiplexer_types.incl.pl1. (Output)

The structure pointed to by logical_chan_ptr has the following form:

```
dcl 1 logical_chan_meters based aligned,
      2 current_meters like lcte.meters,
      2 saved_meters like lcte.meters;
```

current_meters
contains the current values of the logical channel meters. The format of lcte.meters is described by lct.incl.pl1.

saved_meters
contains the values of logical channel meters the last time a copy_meters order was issued.

system_comm_meters system_comm_meters

Name: system_comm_meters

The system_comm_meters command prints out metering information for ring 0 Multics Communications Management.

Usage

```
     system_comm_meters {-control_args}
```

where control_args can be chosen from the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. The metering information is not printed. If -reset has never been given in a process the interval begins at system initialization time.

-report_reset, -rr
prints metering information and then resets the metering interval.

Access Required

Use of the system_comm_meters command requires access to either the metering_ring_zero_peek_ or the phcs_ gate.

Example

The following is a sample of the output of the system_comm_meters command.

```
Total metering time 05:43:27

THROUGHPUT
                              before conversion  after conversion   ratio
Total characters input          17,234,567         15,543,210        0.90
Total characters output        168,012,345        185,876,543        1.14
Average length of input             12.3 characters
Average length of output            59.7 characters
Input characters preconverted       20,435 (1.2% of total)
```

```
                              read                     write
Number of calls               1,456,789                26,357,924
Average time per call          6.37 msec.               9.63 msec.
Average chars. processed      13.5                     57.8
Average chars. per msec.       2.1                      5.8


CHANNEL INTERRUPTS
                          input         output          other        total
software "interrupts"     678,901        423,440        110,011     1,212,35
average time (msec.)        1.34           0.56           0.23          1.01


TTY_BUF SPACE MANAGEMENT

Total size of buffer pool           11,480 words
Number of channels configured          143
Number of multiplexed channels           8

% of buffer pool in use                current    average
    input                                6.9        6.5
    output                              13.4       15.6
    control structures                  15.8       15.3
    ------------------------------------------------------
    total                               36.1       37.4

Smallest amount of free space ever     4,358 words (38% of buffer pool)

                                  allocate         free         total
Number of calls                  24,657,988    20,665,443   45,323,431
Average time per call (msec.)          0.23          0.37         0.29
% of total CPU                         0.14          0.17         0.31
Calls requiring loop on tty_buf lock        1,249,340  (2.83% of total)
Average time spent looping on lock          0.14 msec. (0.01% of total CPU
Number of allocation failures                       0  (0.00% of attempts)


CHANNEL LOCK CONTENTION

Number of calls to tty_lock                        40,392,817
Times channel lock found locked                     2,364,758 (5% of attempts)
Average time spent waiting for lock                 1.8 msec.
Maximum time spent waiting for lock                 3.7 msec.
Number of interrupts queued because channel locked
                                                    25,437 (2.2% of interrupts
ECHO NEGOTIATION

Average time of transaction        3.2 msec.
Number of characters echoed by supervisor   21,576 (0.13% of input chars)
Number of characters echoed by FNPs        335,466 (1.87% of input chars)


ABNORMAL EVENTS

Input restarts                   12,576 (0.8% of read calls)
Output restarts                 304,289 (1.2% of write calls)
Output space overflows           16,384 (0.1% of write calls)
"needs_space" calls                   0
```

Name: channel_comm_meters

The channel_comm_meters command prints out metering information for a specified communications channel or channels.

Usage

```
channel_comm_meters channel_name {-control_args}
```

channel_name
: is the name of the channel for which information is to be printed. If it is the name of an FNP, totals for that FNP are reported. If channel_name is a starname, information for every channel matching the starname is printed.

control_args may be chosen from among the following:

-brief, -bf
: causes a reduced amount of information to be printed for each specified channel.

-error
: causes only those meters to be printed that reflect error conditions.

-since_bootload, -boot
: prints the meters accumulated since each channel's parent multiplexer (or, in the case of an FNP, the system) was last loaded. This control argument is incompatible with -since_dialup (below).

-since_dialup, -dial
: prints the meters accumulated since the channel last dialed up. This is the default. This control argument is incompatible with -since_bootload (above).

-summary, -sum
: causes a one-line summary to be printed for each specified channel. This control argument may not be specified if either -brief or -error is specified.

Notes

If a single channel is specified, the caller must either be the current user of the specified channel or have access to either the metering_ring_zero_peek_ gate or the phcs_ gate. If a starname is specified, the user must have access to one of the above-named gates.

If -brief and -error are both specified, then only those error indications that would be printed with -brief are printed. See the example below.

Examples

In the example below, code characters appear at the beginning of some lines; these characters do not appear in the actual output of the command. The interpretation of the characters is as follows:

A -- this line appears for asynchronous channels only
S -- this line appears for synchronous channels only
B -- this line is among those printed if -brief is specified
E -- this line is among those printed if -error is specified

Only lines marked with both B and E are printed if -brief and -error are both specified.

```
!    channel_comm_meters a.h000

     Total metering time 01:45:13

     a.h000

[The following meters are printed for all channels]

                                 before conversion   after conversion   ratio
  B  Total characters input          984                 935             0.95
  B  Total characters output      10,540              11,400             1.09
  B  Average length of input           8.7                 8.3
  B  Average length of output         63.1                69.4

                                            read       write    control    tota
     Number of calls                         175        194       53         42
     Average time per call (msec.)           2.3        5.8       1.7        4.
     Average chars. processed per call       5.6       56.1
```

```
                                       input     output     other      tota
     Number of software interrupts      113       163        28         30
     Average time per interrupt (msec.) 1.6       2.3       0.8         2.

  B  Effective speed (bps)              1.6      17.5

     Characters passed with average input interrupt   8.7

[The following meters are printed for physical FNP channels only]

                                        input     output
 SB  Messages transmitted                240        224
 SB  Minimum message length                5         12
 SB  Maximum message length              143        508
 SB  Average message length               10.3       57.6

 SBE Invalid input messages             6 (2.5% of total)
 SBE Output messages retransmitted      8 (1.6% of total)
 SBE Timeout waiting for acknowledge    2 (0.4% of output messages)

     Output overlaps in FNP               127
     Average length of DIA request queue  1.7 entries

A    Pre-exhaust status                    12
A  E Exhaust status                         7
A  E Software transfer timing errors        0
A  E Bell/quits                             8
A  E Echo buffer overflows                  2
   E Parity errors                          0

     Avg. number of pending status events   1.9
   E Software status queue overflows        1
   E Hardware status queue overflows        0
   E Input buffer allocation failures       1

[The following meters are printed for an entire FNP]

     FNP has been up for               04:15:12
  B  Number of channels configured        88
  B  Average number dialed up             43.7
  B  FNP idle                             74.9%
   E Abnormal DIA status events            3
   E Memory parity errors                  0

  B  Memory size                          64K
  B  Total available buffer pool       6,360 words
  B  Avg. amount of free space        21,876 words
  B  Average % of buffer pool available   34.7
  BE Buffer allocation failures           12
   E Output restricted by space           24


     Number of interrupts from this FNP   1,964,208
     Avg. time/interrupt (ms)               3.1
     % of total CPU time                    1.1

     Mailbox transactions:
        Input data                         220,349
        Output data                        543,210
        Input control                       14,111
        Output control                      23,456
   -----------------------------------------------------------
        Total                              801,126

     Average inbound mailboxes in use       1.1
     Average outbound mailboxes in use      3.1
     Maximum outbound mailboxes in use     16
   E No outbound mailbox available         37
   E Input rejects                         22
   E % of input transactions rejected       0.01
```

The following example shows the format of the output of the command when the -summary control argument is specified.

```
!    channel_comm_meters a.h00* -summary

  cps   cpsi   cpso  iotxXsbepQqa  err  ABE  name         user

  120    0.2    5.4     xX b  Q     12  aB   a.h000       Coren
  600    2.1  102.1    t X      a   73  s    a.h005       ABClone
   30    0.5    2.6         e        2  a E  a.h009       Parrish
```

The column headings are interpreted as follows:

cps
: the nominal speed of the channel, in characters per second.

cpsi
: the effective speed of input over the channel, in characters per second.

cpso
    the effective speed of output over the channel, in characters per second.

The following flags are printed if the corresponding condition has occurred at least once on the channel.

i -- invalid input message

o -- output message retransmitted

t -- timeout waiting for acknowledge

x -- pre-exhaust status

X -- exhaust status

s -- software transfer timing error

b -- bell/quit

e -- echo buffer overflow

p -- parity error

Q -- software status queue overflow

q -- hardware status queue overflow

a -- input buffer allocation failure

err
    the total number of errors of all kinds that have occurred on the channel.

A
    "a" for an asynchronous channel or "s" for a synchronous channel.

B
    the channel is in breakall mode.

E
    the channel is in echoplex mode.

name
    the name of the channel.

user
the Personid of the current user of the channel. If the channel is not in use, or the user's name is not available, this field is left blank.