

MCR10192

window_io_ get_chars: buffer overrun, and documentation Revision 1.0

Eric Swenson

October 13, 2026

1 Problem Description

[Ticket #107](#): the `window_io_` documentation (AG93, page 3-264, and `window_io_.info`) is wrong about the `get_chars` operation and omits the `printer_on/printer_off` control orders.

- The documentation says `get_chars` “returns exactly one character, unechoed, regardless of the size of the caller’s buffer.” `window_io_iox_$get_chars` actually returns all the characters typed so far, without waiting (possibly none). It echoes each one through `put_chars` unless echoing has been turned off.
- Echoing is controlled by the `printer_off` and `printer_on` control orders (`wioctl_`), which are not documented. `printer_off` also makes `get_line` not display the line being typed and not save it in the kill ring.

While checking the code, a real bug was found: the `get_chars` loop is `do char_count = 1 by 1`, with no limit. If more characters have been typed than fit in the caller’s buffer, it stores them past the end of the buffer.

2 Proposed Fix

- `window_io_iox_.pl1`: the loop is now `do char_count = 1 to Buffer_len`. When the buffer fills, `N_returned` is set to `Buffer_len`. The remaining characters stay queued for the next call.
- `window_io_.info`: new description of the Get Chars Operation, and a new entry for `printer_off`, `printer_on` after `reset_more_handler`.
- `ag93.errata.info`: the same two changes, as errata for page 3-264 and the control operations.

The `window_io_iox_.pl1` change is shown below:

```
cpa [lpn window_io_iox_.pl1] ==
```

```
A3433      /* Read all the characters the user has typed so far. */
A3434          do char_count = 1 by 1;
```

Changed by B to:

```

B3433      /* Read all the characters the user has typed so far, but no more than
B3434          fit in the caller's buffer (the loop had no limit, so typed-ahead
B3435          input longer than the buffer overwrote storage after it; ticket 107). */
B3436
B3437          do char_count = 1 to Buffer_len;

```

Inserted in B:

```

B3461          N_returned = Buffer_len;                      /* buffer is full */
Preceding:
A3458          return;

```

Comparison finished: 2 differences, 8 lines.

The window_io..info change is shown below:

```

cpa >doc>info>window_io..info ==

```

```

A1          04/26/90 window_io_
Changed by B to:
B1          :Info: window_io_: 2026-10-10 window_io_
B2

```

```

A73          This operation returns exactly one character, unechoed, regardless of
A74          the size of the caller's buffer. The line editor is not invoked by
A75          this call.
Changed by B to:
B74          This operation returns the characters that the user has typed so far,
B75          up to the size of the caller's buffer. It does not wait for input; if
B76          nothing has been typed, it returns zero characters. The line editor
B77          is not invoked by this call. Each character returned is echoed
B78          (through the put_chars operation, so that output conversions are
B79          done), unless echoing has been turned off with the printer_off control
B80          order.

```

Inserted in B:

```

B258          printer_off, printer_on
B259          turn echoing of input off and on. While echoing is off, the
B260          characters returned by the get_chars operation are not echoed, and
B261          the line being typed for the get_line operation is not displayed
B262          and is not saved in the kill ring (as when reading a password).
B263          Echoing is on when the switch is attached. No info structure is
B264          used; the info pointer should be null. These control orders are
B265          available from command level with the following io_call commands:
B266              io_call control window_switch printer_off
B267              io_call control window_switch printer_on
B268
B269
Preceding:
A253          get_output_conversion

```

Comparison finished: 3 differences, 25 lines.

The ag93.errata.info change is shown below:

cpa >doc>info>ag93.errata.info ==

Inserted in B:

B478 Page 3-264; replace the description of GET CHARS OPERATION with
B479
B480 This operation returns the characters that the user has
B481 typed so far, up to the size of the caller's buffer. It
B482 does not wait for input; if nothing has been typed, it
B483 returns zero characters. The line editor is not invoked by
B484 this call. Each character returned is echoed (through the
B485 put_chars operation, so that output conversions are done),
B486 unless echoing has been turned off with the printer_off
B487 control order.
B488
B489
B490 Under CONTROL OPERATION, after the description of
B491 "reset_more_handler", add the following.
B492
B493 printer_off, printer_on
B494 turn echoing of input off and on. While echoing is off,
B495 the characters returned by the get_chars operation are not
B496 echoed, and the line being typed for the get_line operation
B497 is not displayed and is not saved in the kill ring (as when
B498 reading a password). Echoing is on when the switch is
B499 attached. No info structure is used; the info pointer
B500 should be null. These control orders are available from
B501 command level with the following io_call commands:
B502
B503 io_call control window_switch printer_off
B504 io_call control window_switch printer_on
B505
B506
Preceding:
A478 Page 3-266; under STRUCTURE ELEMENTS replace

Comparison finished: 1 difference, 29 lines.

3 Documentation

As above.

4 Test

1. With the video system on, call `iox_$get_chars` on `user_i/o` with a 4-character buffer after typing ahead 10 characters: 4 are returned, then the rest on the following calls, and nothing after the buffer is changed.
2. `io_call` control `user_i/o` `printer_off`, then `get_chars` and `get_line`: no echo; `printer_on` restores it.
3. `verify_info` on `window_io..info`.

5 Files Changed

<code>bound_video..s.archive</code>	<code>window_io_iox..pl1</code>
<code>>sss</code>	<code>bound_video_ (rebound)</code>
<code>>doc>info</code>	<code>window_io..info</code> , <code>ag93.errata.info</code>

6 Version History

2026-10-10	1.0	Initial version of the MCR.
------------	-----	-----------------------------