

parallel

COLLABORATORS

	<i>TITLE :</i> parallel		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	parallel	1
1.1	parallel.doc	1
1.2	parallel.device/CMD_CLEAR	1
1.3	parallel.device/CMD_FLUSH	1
1.4	parallel.device/CMD_READ	2
1.5	parallel.device/CMD_RESET	2
1.6	parallel.device/CMD_START	3
1.7	parallel.device/CMD_STOP	3
1.8	parallel.device/CMD_WRITE	3
1.9	parallel.device/OpenDevice	4
1.10	parallel.device/PDCMD_QUERY	5
1.11	parallel.device/PDCMD_SETPARAMS	5

Chapter 1

parallel

1.1 parallel.doc

CMD_CLEAR	CMD_RESET	CMD_WRITE	PDCMD_SETPARAMS
CMD_FLUSH	CMD_START	OpenDevice()	
CMD_READ	CMD_STOP	PDCMD_QUERY	

1.2 parallel.device/CMD_CLEAR

NAME

Clear -- clear the parallel port buffer

FUNCTION

This command just RTS's (no buffer to clear)

IO REQUEST

io_Message	mn_ReplyPort initialized
io_Device	set by OpenDevice
io_Unit	set by OpenDevice
io_Command	CMD_CLEAR (05)

1.3 parallel.device/CMD_FLUSH

NAME

Flush -- clear all queued I/O requests for the parallel port

FUNCTION

This command purges the read and write request queues for the parallel device. The currently active request is not purged.

IO REQUEST

io_Message	mn_ReplyPort initialized
io_Device	set by OpenDevice
io_Unit	set by OpenDevice
io_Command	CMD_FLUSH (08)

1.4 parallel.device/CMD_READ

NAME

Read -- read input from parallel port

FUNCTION

This command causes a stream of characters to be read from the parallel I/O register. The number of characters is specified in `io_Length`. The EOF and EOL modes are supported, but be warned that using these modes can result in a buffer overflow if the proper EOL or EOF character is not received in time. These modes should be used only when the sender and receiver have been designed to cooperate. A safety guard can be implemented to EOF by setting `io_Length` to a maximum allowed value. That cannot be done with EOL since the EOL mode is identified by `io_Length=-1`.

The `parallel.device` has no internal buffer; if no read request has been made, pending input (i.e. handshake request) is not acknowledged.

IO REQUEST

<code>io_Message</code>	<code>mn_ReplyPort</code> initialized
<code>io_Device</code>	set by <code>OpenDevice</code>
<code>io_Unit</code>	set by <code>OpenDevice</code>
<code>io_Command</code>	<code>CMD_READ</code> (02)
<code>io_Flags</code>	If <code>IOF_QUICK</code> is set, driver will attempt Quick IO
<code>io_Length</code>	number of characters to receive.
<code>io_Data</code>	pointer where to put the data.

RESULTS

`io_Error` -- if the Read succeeded, then `io_Error` will be null.
If the Read failed, then `io_Error` will contain an error code.

SEE ALSO

`parallel.device/PDCMD_SETPARAMS`

1.5 parallel.device/CMD_RESET

NAME

Reset -- reinitializes the parallel device

FUNCTION

This command resets the parallel device to its freshly initialized condition. It aborts all I/O requests both queued and current and sets the devices's flags and parameters to their boot-up time default values. At boot-up time the `PTermArray` is random, and it will be so also here.

IO REQUEST

<code>io_Message</code>	<code>mn_ReplyPort</code> initialized
<code>io_Device</code>	set by <code>OpenDevice</code>
<code>io_Unit</code>	set by <code>OpenDevice</code>
<code>io_Command</code>	<code>CMD_RESET</code> (01)

RESULTS

Error -- if the Reset succeeded, then `io_Error` will be null.
if the Reset failed, then the `io_Error` will be non-zero.

1.6 parallel.device/CMD_START

NAME

Start -- restart paused I/O over the parallel port

FUNCTION

This command restarts the current I/O activity on the parallel port by reactivating the handshaking sequence.

IO REQUEST

<code>io_Message</code>	<code>mn_ReplyPort</code> initialized
<code>io_Device</code>	set by <code>OpenDevice</code>
<code>io_Unit</code>	set by <code>OpenDevice</code>
<code>io_Command</code>	<code>CMD_START</code> (07)

SEE ALSO

`parallel.device/CMD_STOP`

1.7 parallel.device/CMD_STOP

NAME

Stop -- pause current activity on the parallel device

FUNCTION

This command halts the current I/O activity on the parallel device by discontinuing the handshaking sequence. The stop and start commands may not be nested.

IO REQUEST

<code>io_Message</code>	<code>mn_ReplyPort</code> initialized
<code>io_Device</code>	set by <code>OpenDevice</code>
<code>io_Unit</code>	set by <code>OpenDevice</code>
<code>io_Command</code>	<code>CMD_STOP</code> (06)

SEE ALSO

`parallel.device/CMD_START`

1.8 parallel.device/CMD_WRITE

NAME

Write -- send output to parallel port

FUNCTION

This command causes a stream of characters to be written to the parallel output register. The number of characters is specified in `io_Length`, unless `-1` is used, in which case output is sent until

a zero byte occurs in the data. This is independent of, and may be used simultaneously with setting the EOFMODE in `io_ParFlags` and using the `PTermArray` to terminate the read or write.

IO REQUEST

<code>io_Message</code>	<code>mn_ReplyPort</code> initialized
<code>io_Device</code>	set by <code>OpenDevice</code>
<code>io_Unit</code>	set by <code>OpenDevice</code>
<code>io_Command</code>	<code>CMD_WRITE</code> (03)
<code>io_Flags</code>	If <code>IOF_QUICK</code> is set, driver will attempt Quick IO
<code>io_Length</code>	number of characters to transmit, or if set to -1 send until zero byte encountered
<code>io_Data</code>	pointer to block of data to transmit

RESULTS

`io_Error` -- If the Write succeeded, then `io_Error` will be null.
If the Write failed, then `io_Error` will contain an error code.

SEE ALSO

`parallel.device/PDCMD_SETPARAMS`

1.9 parallel.device/OpenDevice

NAME

`Open` -- a request to open the parallel port

SYNOPSIS

```
error = OpenDevice("parallel.device", unit, ioExtPar, flags)
D0          A0          D0    A1    D1
```

FUNCTION

This function allows the requestor software access to the parallel device. Unless the shared-access bit (bit 5 of `io_ParFlags`) is set, exclusive use is granted and no other access is allowed until the owner closes the device.

A `FAST_MODE`, can be specified (bit 3 of `io_Parflags`) to speed up transfers to high-speed printers. Rather than waiting for the printer to acknowledge a character using the `*ACK` interrupt, this mode will send out data as long as the `BUSY` signal is low. The printer must be able to raise the `BUSY` signal within 3 micro-seconds on A2630s, otherwise data will be lost. Should be used only in an exclusive-access `OpenDevice()`.

A `SLOWMODE` mode can be specified (bit 4 of `io_ParFlags`) when very slow printers are used. If the printer acknowledges data at less than 5000 bytes per second, then this mode will actually save CPU time, although it consumes much more with high-speed printers.

The `PTermArray` of the `ioExtPar` is initialized only if the `EOFMODE` bit (bit 1 of `io_ParFlags`) is set. The `PTermArray` can be further modified using the `PDCMD_SETPARAMS` command.

INPUTS

"parallel.device" - a pointer to literal string "parallel.device"

unit - Must be zero for future compatibility
 ioExtPar - pointer to an IO Request block of structure IOExtPar to
 be initialized by the OpenDevice() function. The
 io_ParFlags field must be set as desired.
 flags - Must be zero for future compatibility

RESULTS

d0 -- same as io_Error
 io_Error -- if the Open succeeded, then io_Error will be null.
 If the Open failed, then io_Error will be non-zero.

SEE ALSO

exec/CloseDevice

1.10 parallel.device/PDCMD_QUERY

NAME

Query -- query parallel port/line status

FUNCTION

This command return the status of the parallel port lines and registers.

IO REQUEST

io_Message must have mn_ReplyPort initialized
 io_Device set by OpenDevice
 io_Unit set by OpenDevice
 io_Command PDCMD_QUERY (09)

RESULTS

io_Status	BIT	ACTIVE	FUNCTION
	0	high	printer busy toggle (offline)
	1	high	paper out
	2	high	printer selected on the A1000
			printer selected & serial "Ring Indicator" on the A500/A2000
			Use care when making cables.
	3	-	read=0,write=1
	4-7		reserved

BUGS

In a earlier version of this AutoDoc, BUSY and PSEL were reversed.
 The function has always been correct.

1.11 parallel.device/PDCMD_SETPARAMS

NAME

SetParams -- change parameters for the parallel device

FUNCTION

This command allows the caller to change the EOFMODE parameter for

the parallel port device. It will disallow changes if any reads or writes are active or queued.

The PARB_EOFMODE bit of `io_ParFlags` controls whether the `io_PTermArray` is to be used as an additional termination criteria for reads and writes. It may be set directly without a call to `SetParams`, setting it here performs the additional service of copying the `PTermArray` into the device default array which is used as the initial array for subsequent device opens. The Shared bit can be changed here, and overrides the current device access mode set at `OpenDevice` time.

IO REQUEST

<code>io_Message</code>	<code>mn_ReplyPort</code> initialized
<code>io_Device</code>	preset by <code>OpenDevice</code>
<code>io_Unit</code>	preset by <code>OpenDevice</code>
<code>io_Command</code>	<code>PDCMD_SETPARAMS</code> (0A)
	NOTE that the following fields of your <code>IORequest</code> are filled by <code>Open</code> to reflect the parallel device's current configuration.
<code>io_PExtFlags</code>	must be set to zero, unless used
<code>io_ParFlags</code>	see definition in <code>parallel.i</code> or <code>parallel.h</code>
	NOTE that <code>x00</code> yields exclusive access, <code>PTermArray</code> inactive.
<code>io_PTermArray</code>	ASCII descending-ordered 8-byte array of termination characters. If less than 8 chars used, fill out array w/lowest valid value. Terminators are used only if EOFMODE bit of <code>io_Parflags</code> is set. (e.g. <code>x512F040303030303</code>) This field is filled on <code>OpenDevice</code> only if the EOFMODE bit is set.

RESULTS

`io_Error` -- if the `SetParams` succeeded, then `io_Error` will be null.
if the `SetParams` failed, then `io_Error` will be non-zero.
