

audio

COLLABORATORS

	<i>TITLE :</i> audio		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		March 29, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	audio	1
1.1	audio.doc	1
1.2	audio.device/AbortIO	1
1.3	audio.device/ADCMD_ALLOCATE	2
1.4	audio.device/ADCMD_FINISH	3
1.5	audio.device/ADCMD_FREE	4
1.6	audio.device/ADCMD_LOCK	5
1.7	audio.device/ADCMD_PERVOL	6
1.8	audio.device/ADCMD_SETPREC	7
1.9	audio.device/ADCMD_WAITCYCLE	8
1.10	audio.device/BeginIO	8
1.11	audio.device/CloseDevice	9
1.12	audio.device/CMD_CLEAR	9
1.13	audio.device/CMD_FLUSH	10
1.14	audio.device/CMD_READ	11
1.15	audio.device/CMD_RESET	11
1.16	audio.device/CMD_START	12
1.17	audio.device/CMD_STOP	13
1.18	audio.device/CMD_UPDATE	14
1.19	audio.device/CMD_WRITE	14
1.20	audio.device/Expunge	15
1.21	audio.device/OpenDevice	16

Chapter 1

audio

1.1 audio.doc

```
AbortIO ()
ADCMD_ALLOCATE
ADCMD_FINISH
ADCMD_FREE
ADCMD_LOCK
ADCMD_PERVOL
ADCMD_SETPREC
ADCMD_WAITCYCLE
BeginIO ()
CloseDevice ()
CMD_CLEAR
CMD_FLUSH
CMD_READ
CMD_RESET
CMD_START
CMD_STOP
CMD_UPDATE
CMD_WRITE
Expunge ()
OpenDevice ()
```

1.2 audio.device/AbortIO

NAME

AbortIO - abort a device command

SYNOPSIS

```
AbortIO (ioRequest);
A1
```

FUNCTION

AbortIO tries to abort a device command. It is allowed to be unsuccessful. If the Abort is successful, the `io_Error` field of the `ioRequest` contains an indication that IO was aborted.

INPUTS

ioRequest -- pointer to the I/O Request for the command to abort

1.3 audio.device/ADCMD_ALLOCATE

ADCMD_ALLOCATE -- allocate a set of audio channels

FUNCTION

ADCMD_ALLOCATE is a command that allocates multiple audio channels. ADCMD_ALLOCATE takes an array of possible channel combinations (ioa_Data) and an allocation precedence (ln_Pri) and tries to allocate one of the combinations of channels.

If the channel combination array is zero length (ioa_Length), the allocation succeeds; otherwise, ADCMD_ALLOCATE checks each combination, one at a time, in the specified order, to find one combination that does not require ADCMD_ALLOCATE to steal allocated channels.

If it must steal allocated channels, it uses the channel combination that steals the lowest precedence channels.

ADCMD_ALLOCATE cannot steal a channel of equal or greater precedence than the allocation precedence (ln_Pri).

If it fails to allocate any channel combination and the no-wait flag (ADIOF_NOWAIT) is set ADCMD_ALLOCATE returns a zero in the unit field of the I/O request (io_Unit) and an error (IOERR_ALLOCFAILED). If the no-wait flag is clear, it places the I/O request in a list that tries to allocate again whenever ADCMD_FREE frees channels or ADCMD_SETPREC lowers the channels' precedences.

If the allocation is successful, ADCMD_ALLOCATE checks if any channels are locked (ADCMD_LOCK) and if so, replies (ReplyMsg) the lock I/O request with an error (ADIOERR_CHANNELSTOLEN). Then it places the allocation I/O request in a list waiting for the locked channels to be freed. When all the allocated channels are un-locked, ADCMD_ALLOCATE:

- . resets (CMD_RESET) the allocated channels,
- . generates a new allocation key (ioa_AllocKey), if it is zero,
- . copies the allocation key into each of the allocated channels
- . copies the allocation precedence into each of the allocated channels, and
- . copies the channel bit map into the unit field of the I/O request.

If channels are allocated with a non-zero allocation key, ADCMD_ALLOCATE allocates with that same key; otherwise, it generates a new and unique key.

ADCMD_ALLOCATE is synchronous:

- . if the allocation succeeds and there are no locked channels to be stolen, or
- . if the allocation fails and the no-wait flag is set.

In either case, ADCMD_ALLOCATE only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear; otherwise, the allocation is

asynchronous, so it clears the quick flag and replies the I/O request after the allocation is finished. If channels are stolen, all audio device commands return an error (IOERR_NOALLOCATION) when the former user tries to use them again. Do not use ADCMD_ALLOCATE in interrupt code.

If you decide to store directly to the audio hardware registers, you must either lock the channels you've allocated, or set the precedence to maximum (ADALLOC_MAXPREC) to prevent the channels from being stolen.

Under all circumstances, unless channels are stolen, you must free (ADCMD_FREE) all allocated channels when you are finished using them.

INPUTS

ln_Pri - allocation precedence (-128 thru 127)
 mn_ReplyPort- pointer to message port that receives I/O request after the allocation completes is asynchronous or quick flag (ADIOF_QUICK) is set
 io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
 io_Command - command number for ADCMD_ALLOCATE
 io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 (SET) only reply I/O request only if asynchronous (see above text)
 ADIOF_NOWAIT- (CLEAR) if allocation fails, wait till it succeeds
 (SET) if allocation fails, return error (ADIOERR_ALLOCFAILED)
 ioa_AllocKey- allocation key, zero to generate new key; otherwise, it must be set by (or copied from I/O block set by) OpenDevice function or previous ADCMD_ALLOCATE command
 ioa_Data - pointer to channel combination options (byte array, bits 0 thru 3 correspond to channels 0 thru 3)
 ioa_Length - length of the channel combination option array (0 thru 16, 0 always succeeds)

OUTPUTS

io_Unit - bit map of successfully allocated channels (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Flags - IOF_QUICK flag cleared if asynchronous (see above text)
 io_Error - error number:
 0 - no error
 ADIOERR_ALLOCFAILED - allocation failed
 ioa_AllocKey- allocation key, set to a unique number if passed a zero and command succeeds

1.4 audio.device/ADCMD_FINISH

NAME

ADCMD_FINISH -- abort writes in progress to audio channels

FUNCTION

ADCMD_FINISH is a command for multiple audio channels. For each

selected channel (`io_Unit`), if the allocation key (`ioa_AllocKey`) is correct and there is a write (`CMD_WRITE`) in progress, `ADCMD_FINISH` aborts the current write immediately or at the end of the current cycle depending on the sync flag (`ADIOF_SYNCCYCLE`). If the allocation key is incorrect `ADCMD_FINISH` returns an error (`ADIOERR_NOALLOCATION`). `ADCMD_FINISH` is synchronous and only replies (`mn_ReplyPort`) if the quick flag (`IOF_QUICK`) is clear. Do not use `ADCMD_FINISH` in interrupt code at interrupt level 5 or higher.

INPUTS

`mn_ReplyPort` - pointer to message port that receives I/O request if the quick flag (`IOF_QUICK`) is clear

`io_Device` - pointer to device node, must be set by (or copied from I/O block set by) `OpenDevice` function

`io_Unit` - bit map of channels to finish (bits 0 thru 3 correspond to channels 0 thru 3)

`io_Command` - command number for `ADCMD_FINISH`

`io_Flags` - flags, must be cleared if not used:

- `IOF_QUICK` - (CLEAR) reply I/O request
- `ADIOF_SYNCCYCLE` - (CLEAR) finish immediately (SET) finish at the end of current cycle

`ioa_AllocKey` - allocation key, must be set by (or copied from I/O block set by) `OpenDevice` function or `ADCMD_ALLOCATE` command

OUTPUTS

`io_Unit` - bit map of channels successfully finished (bits 0 thru 3 correspond to channels 0 thru 3)

`io_Error` - error number:

- 0 - no error
- `ADIOERR_NOALLOCATION` - allocation key (`ioa_AllocKey`) does not match key for channel

1.5 audio.device/ADCMD_FREE

NAME

`ADCMD_FREE` -- free audio channels for allocation

FUNCTION

`ADCMD_FREE` is a command for multiple audio channels. For each selected channel (`io_Unit`), if the allocation key (`ioa_AllocKey`) is correct, `ADCMD_FREE` does the following:

- . restores the channel to a known state (`CMD_RESET`),
- . changes the channels allocation key, and
- . makes the channel available for re-allocation.
- . If the channel is locked (`ADCMD_LOCK`) `ADCMD_FREE` unlocks it and clears the bit for the channel (`io_Unit`) in the lock I/O request. If the lock I/O request has no channel bits set `ADCMD_FREE` replies the lock I/O request, and
- . checks if there are allocation requests (`ADCMD_ALLOCATE`) waiting for the channel.

Otherwise, `ADCMD_FREE` returns an error (`ADIOERR_NOALLOCATION`). `ADCMD_FREE` is synchronous and only replies (`mn_ReplyPort`) if the quick

flag (IOF_QUICK) is clear. Do not use ADCMD_FREE in interrupt code.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request
if the quick flag (IOF_QUICK) is clear
io_Device - pointer to device node, must be set by (or copied from
I/O block set by) OpenDevice function
io_Unit - bit map of channels to free (bits 0 thru 3 correspond to
channels 0 thru 3)
io_Command - command number for ADCMD_FREE
io_Flags - flags, must be cleared if not used:
IOF_QUICK - (CLEAR) reply I/O request
ioa_AllocKey- allocation key, must be set by (or copied from I/O block
set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully freed (bits 0 thru 3
correspond to channels 0 thru 3)
io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey)
does not match key for channel

1.6 audio.device/ADCMD_LOCK

NAME

ADCMD_LOCK -- prevent audio channels from being stolen

FUNCTION

ADCMD_LOCK is a command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, ADCMD_LOCK locks the channel, preventing subsequent allocations (ADCMD_ALLOCATE or OpenDevice) from stealing the channel. Otherwise, ADCMD_LOCK returns an error (ADIOERR_NOALLOCATION) and will not lock any channels.

Unlike setting the precedence (ADCMD_SETPREC, ADCMD_ALLOCATE or OpenDevice) to maximum (ADALLOC_MAXPREC) which would cause all subsequent allocations to fail, ADCMD_LOCK causes all higher precedence allocations, even no-wait (ADIOF_NOWAIT) allocations, to wait until the channels are un-locked.

Locked channels can only be unlocked by freeing them (ADCMD_FREE), which clears the channel select bits (io_Unit). ADCMD_LOCK does not reply the I/O request (mn_ReplyPort) until all the channels it locks are freed, unless a higher precedence allocation attempts to steal one the locked channels. If a steal occurs, ADCMD_LOCK replies and returns an error (ADIOERR_CHANNELSTOLEN). If the lock is replied (mn_ReplyPort) with this error, the channels should be freed as soon as possible. To avoid a possible deadlock, never make the freeing of stolen channels dependent on another allocations completion.

ADCMD_LOCK is only asynchronous if the allocation key is correct, in which case it clears the quick flag (IOF_QUICK); otherwise, it is synchronous and only replies if the quick flag (IOF_QUICK) is clear.

Do not use ADCMD_LOCK in interrupt code.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request
if the quick flag (IOF_QUICK) is clear
io_Device - pointer to device node, must be set by (or copied from
I/O block set by) OpenDevice function
io_Unit - bit map of channels to lock (bits 0 thru 3 correspond to
channels 0 thru 3)
io_Command - command number for ADCMD_LOCK
io_Flags - flags, must be cleared
ioa_AllocKey- allocation key, must be set by (or copied from I/O block
set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of successfully locked channels (bits 0 thru 3
correspond to channels 0 thru 3) not freed (ADCMD_FREE)
io_Flags - IOF_QUICK flag cleared if the allocation key is correct
(no ADIOERR_NOALLOCATION error)
io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey)
does not match key for channel
ADIOERR_CHANNELSTOLEN- allocation attempting to steal
locked channel

1.7 audio.device/ADCMD_PERVOL

NAME

ADCMD_PERVOL -- change the period and volume for writes in progress to
audio channels

FUNCTION

ADCMD_PERVOL is a command for multiple audio channels. For each
selected channel (io_Unit), if the allocation key (ioa_AllocKey) is
correct and there is a write (CMD_WRITE) in progress, ADCMD_PERVOL
loads a new volume and period immediately or at the end of the current
cycle depending on the sync flag (ADIOF_SYNCCYCLE). If the allocation
key is incorrect, ADCMD_PERVOL returns an error
(ADIOERR_NOALLOCATION). ADCMD_PERVOL is synchronous and only replies
(mn_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use
ADCMD_PERVOL in interrupt code at interrupt level 5 or higher.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request
if the quick flag (IOF_QUICK) is clear
io_Device - pointer to device node, must be set by (or copied from
I/O block set by) OpenDevice function
io_Unit - bit map of channels to load period and volume (bits 0
thru 3 correspond to channels 0 thru 3)
io_Command - command number for ADCMD_PERVOL
io_Flags - flags, must be cleared if not used:
IOF_QUICK - (CLEAR) reply I/O request
ADIOF_SYNCCYCLE- (CLEAR) load period and volume
immediately

(SET) load period and volume at the end of the current cycle

ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

ioa_Period - new sample period in 279.365 ns increments (124 thru 65536, anti-aliasing filter works below 300 to 500 depending on waveform)

ioa_Volume - new volume (0 thru 64, linear)

OUTPUTS

io_Unit - bit map of channels that successfully loaded period and volume (bits 0 thru 3 correspond to channels 0 thru 3)

io_Error - error number:

- 0 - no error
- ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

1.8 audio.device/ADCMD_SETPREC

NAME

ADCMD_SETPREC -- set the allocation precedence for audio channels

FUNCTION

ADCMD_SETPREC is a command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, ADCMD_SETPREC sets the allocation precedence to a new value (ln_Pri) and checks if there are allocation requests (ADCMD_ALLOCATE) waiting for the channel which now have higher precedence; otherwise, ADCMD_SETPREC returns an error (ADIOERR_NOALLOCATION). ADCMD_SETPREC is synchronous and only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use ADCMD_SETPREC in interrupt code.

INPUTS

ln_Pri - new allocation precedence (-128 thru 127)

mn_ReplyPort- pointer to message port that receives I/O request if the quick flag (IOF_QUICK) is clear

io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function

io_Unit - bit map of channels to set precedence (bits 0 thru 3 correspond to channels 0 thru 3)

io_Command - command number for ADCMD_SETPREC

io_Flags - flags, must be cleared if not used:

- IOF_QUICK - (CLEAR) reply I/O request

ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels that successfully set precedence (bits 0 thru 3 correspond to channels 0 thru 3)

io_Error - error number:

- 0 - no error
- ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

1.9 audio.device/ADCMD_WAITCYCLE

NAME

ADCMD_WAITCYCLE -- wait for an audio channel to complete the current cycle of a write

FUNCTION

ADCMD_WAITCYCLE is a command for a single audio channel (io_Unit). If the allocation key (ioa_AllocKey) is correct and there is a write (CMD_WRITE) in progress on selected channel, ADCMD_WAITCYCLE does not reply (mn_ReplyPort) until the end of the current cycle. If there is no write in progress, ADCMD_WAITCYCLE replies immediately. If the allocation key is incorrect, ADCMD_WAITCYCLE returns an error (ADIOERR_NOALLOCATION). ADCMD_WAITCYCLE returns an error (IOERR_ABORTED) if it is canceled (AbortIO) or the channel is stolen (ADCMD_ALLOCATE). ADCMD_WAITCYCLE is only asynchronous if it is waiting for a cycle to complete, in which case it clears the quick flag (IOF_QUICK); otherwise, it is synchronous and only replies if the quick flag (IOF_QUICK) is clear. Do not use ADCMD_WAITCYCLE in interrupt code at interrupt level 5 or higher.

INPUTS

mn_ReplyPort - pointer to message port that receives I/O request, if the quick flag (IOF_QUICK) is clear, or if a write is in progress on the selected channel and a cycle has completed

io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function

io_Unit - bit map of channel to wait for cycle (bits 0 thru 3 correspond to channels 0 thru 3), if more than one bit is set lowest bit number channel is used

io_Command - command number for CMD_WAITCYCLE

io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 (SET) only reply I/O request if a write is in progress on the selected channel and a cycle has completed

ioa_AllocKey - allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channel that successfully waited for cycle (bits 0 thru 3 correspond to channels 0 thru 3)

io_Flags - IOF_QUICK flag cleared if a write is in progress on the selected channel

io_Error - error number:
 0 - no error
 IOERR_ABORTED - canceled (AbortIO) or channel stolen
 ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

1.10 audio.device/BeginIO

NAME

BeginIO - dispatch a device command

SYNOPSIS

```
BeginIO(ioRequest);  
    A1
```

FUNCTION

BeginIO has the responsibility of dispatching all device commands. Immediate commands are always called directly, and all other commands are queued to make them single threaded.

INPUTS

ioRequest -- pointer to the I/O Request for this command

1.11 audio.device/CloseDevice

NAME

CloseDevice - terminate access to the audio device

SYNOPSIS

```
CloseDevice(ioRequest);  
    A1
```

FUNCTION

The CloseDevice routine notifies the audio device that it will no longer be used. It takes an I/O audio request block (IOAudio) and clears the device pointer (io_Device). If there are any channels allocated with the same allocation key (ioa_AllocKey), CloseDevice frees (ADCMD_FREE) them. CloseDevice decrements the open count, and if it falls to zero and an expunge (Expunge) is pending, the device is expunged.

INPUTS

ioRequest - pointer to audio request block (struct IOAudio)
io_Device - pointer to device node, must be set by (or copied from I/O block set by) open (OpenDevice)
io_Unit - bit map of channels to free (ADCMD_FREE) (bits 0 thru 3 correspond to channels 0 thru 3)
ioa_AllocKey- allocation key, used to free channels

OUTPUTS

ioRequest - pointer to audio request block (struct IOAudio)
io_Device - set to -1
io_Unit - set to zero

1.12 audio.device/CMD_CLEAR

NAME

CMD_CLEAR -- throw away internal caches

FUNCTION

CMD_CLEAR is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_CLEAR does nothing; otherwise, CMD_CLEAR returns an error (ADIOERR_NOALLOCATION). CMD_CLEAR is synchronous and only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear
 io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
 io_Unit - bit map of channels to clear (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Command - command number for CMD_CLEAR
 io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully cleared (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Error - error number:
 0 - no error
 ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

1.13 audio.device/CMD_FLUSH

NAME

CMD_FLUSH -- cancel all pending I/O

FUNCTION

CMD_FLUSH is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_FLUSH aborts all writes (CMD_WRITE) in progress or queued and any I/O requests waiting to synchronize with the end of the cycle (ADCMD_WAITCYCLE); otherwise, CMD_FLUSH returns an error (ADIOERR_NOALLOCATION). CMD_FLUSH is synchronous and only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use CMD_FLUSH in interrupt code at interrupt level 5 or higher.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear
 io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
 io_Unit - bit map of channels to flush (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Command - command number for CMD_FLUSH
 io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully flushed (bits 0 thru 3 correspond to channels 0 thru 3)

io_Error - error number:

- 0 - no error
- ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

1.14 audio.device/CMD_READ

NAME

CMD_READ -- normal I/O entry point

FUNCTION

CMD_READ is a standard command for a single audio channel (io_Unit). If the allocation key (ioa_AllocKey) is correct, CMD_READ returns a pointer (io_Data) to the I/O block currently writing (CMD_WRITE) on the selected channel; otherwise, CMD_READ returns an error (ADIOERR_NOALLOCATION). If there is no write in progress, CMD_READ returns zero. CMD_READ is synchronous and only replies (mn_ReplyPort) if the quick bit (IOF_QUICK) is clear.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear

io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function

io_Unit - bit map of channel to read (bit 0 thru 3 corresponds to channel 0 thru 3), if more then one bit is set lowest bit number channel read

io_Command - command number for CMD_READ

io_Flags - flags, must be cleared if not used:

- IOF_QUICK - (CLEAR) reply I/O request

ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channel successfully read (bit 0 thru 3 corresponds to channel 0 thru 3)

io_Error - error number:

- 0 - no error
- ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

ioa_Data - pointer to I/O block for current write, zero if none is progress

1.15 audio.device/CMD_RESET

NAME

CMD_RESET -- restore device to a known state

FUNCTION

CMD_RESET is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_RESET:

- . clears the hardware audio registers and attach bits,
- . sets the audio interrupt vector,
- . cancels all pending I/O (CMD_FLUSH), and
- . un-stops the channel if it is stopped (CMD_STOP),

Otherwise, CMD_RESET returns an error (ADIOERR_NOALLOCATION).

CMD_RESET is synchronous and only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use CMD_RESET in interrupt code at interrupt level 5 or higher.

INPUTS

- mn_ReplyPort- pointer to message port that receives I/O request if the quick flag (IOF_QUICK) is clear
- io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
- io_Unit - bit map of channels to reset (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Command - command number for CMD_RESET
- io_Flags - flags, must be cleared if not used:
IOF_QUICK - (CLEAR) reply I/O request
- ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

- io_Unit - bit map of channels to successfully reset (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

1.16 audio.device/CMD_START

NAME

CMD_START -- start device processing (like ^Q)

FUNCTION

CMD_START is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct and the channel was previously stopped (CMD_STOP), CMD_START immediately starts all writes (CMD_WRITE) to the channel. If the allocation key is incorrect, CMD_START returns an error (ADIOERR_NOALLOCATION). CMD_START starts multiple channels simultaneously to minimize distortion if the channels are playing the same waveform and their outputs are mixed. CMD_START is synchronous and

only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do

not use CMD_START in interrupt code at interrupt level 5 or higher.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear

io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function

io_Unit - bit map of channels to start (bits 0 thru 3 correspond to channels 0 thru 3)

io_Command - command number for CMD_START

io_Flags - flags, must be cleared if not used:
IOF_QUICK - (CLEAR) reply I/O request

ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully started (bits 0 thru 3 correspond to channels 0 thru 3)

io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

1.17 audio.device/CMD_STOP

NAME

CMD_STOP -- stop device processing (like ^S)

FUNCTION

CMD_STOP is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_STOP immediately stops any writes (CMD_WRITE) in progress; otherwise, CMD_STOP returns an error (ADIOERR_NOALLOCATION). CMD_WRITE queues up writes to a stopped channel until CMD_START starts the channel or CMD_RESET resets the channel. CMD_STOP is synchronous and only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use CMD_STOP in interrupt code at interrupt level 5 or higher.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear

io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function

io_Unit - bit map of channels to stop (bits 0 thru 3 correspond to channels 0 thru 3)

io_Command - command number for CMD_STOP

io_Flags - flags, must be cleared if not used:
IOF_QUICK - (CLEAR) reply I/O request

ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully stopped (bits 0 thru 3 correspond to channels 0 thru 3)

io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey)

does not match key for channel

1.18 audio.device/CMD_UPDATE

NAME

CMD_UPDATE -- force dirty buffers out

FUNCTION

CMD_UPDATE is a standard command for multiple audio channels. For each selected channel (`io_Unit`), if the allocation key (`ioa_AllocKey`) is correct, CMD_UPDATE does nothing; otherwise, CMD_UPDATE returns an error (`ADIOERR_NOALLOCATION`). CMD_UPDATE is synchronous and only replies (`mn_ReplyPort`) if the quick flag (`IOF_QUICK`) is clear.

INPUTS

`mn_ReplyPort` - pointer to message port that receives I/O request after if the quick flag (`IOF_QUICK`) is clear
`io_Device` - pointer to device node, must be set by (or copied from I/O block set by) `OpenDevice` function
`io_Unit` - bit map of channels to update (bits 0 thru 3 correspond to channels 0 thru 3)
`io_Command` - command number for CMD_UPDATE
`io_Flags` - flags, must be cleared if not used:
 `IOF_QUICK` - (CLEAR) reply I/O request
`ioa_AllocKey` - allocation key, must be set by (or copied from I/O block set by) `OpenDevice` function or `ADCMD_ALLOCATE` command

OUTPUTS

`io_Unit` - bit map of channels successfully updated (bits 0 thru 3 correspond to channels 0 thru 3)
`io_Error` - error number:
 0 - no error
 `ADIOERR_NOALLOCATION` - allocation key (`ioa_AllocKey`) does not match key for channel

1.19 audio.device/CMD_WRITE

NAME

CMD_WRITE -- normal I/O entry point

FUNCTION

CMD_WRITE is a standard command for a single audio channel (`io_Unit`). If the allocation key (`ioa_AllocKey`) is correct, CMD_WRITE plays a sound using the selected channel; otherwise, it returns an error (`ADIOERR_NOALLOCATION`). CMD_WRITE queues up requests if there is another write in progress or if the channel is stopped (`CMD_STOP`). When the write actually starts; if the `ADIOF_PERVOL` flag is set, CMD_WRITE loads volume (`ioa_Volume`) and period (`ioa_Period`), and if the `ADIOF_WRITEMESSAGE` flag is set, CMD_WRITE replies the write message (`ioa_WriteMsg`). CMD_WRITE returns an error (`IOERR_ABORTED`) if it is canceled (`AbortIO`) or the channel is stolen (`ADCMD_ALLOCATE`). CMD_WRITE is only asynchronous if there is no error, in which case it

clears the quick flag (IOF_QUICK) and replies the I/O request (mn_ReplyPort) after it finishes writting; otherwise, it is synchronous and only replies if the quick flag (IOF_QUICK) is clear. Do not use CMD_WRITE in interrupt code at interrupt level 5 or higher.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request after the write completes

io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function

io_Unit - bit map of channel to write (bit 0 thru 3 corresponds to channel 0 thru 3), if more then one bit is set lowest bit number channel is written

io_Command - command number for CMD_WRITE

io_Flags - flags, must be cleared if not used:
 ADIOF_PERVOL - (SET) load volume and period
 ADIOF_WRITEMESSAGE - (SET) reply message at write start

ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

ioa_Data - pointer to waveform array (signed bytes (-128 thru 127) in custom chip addressable ram and word aligned)

ioa_Length - length of the wave array in bytes (2 thru 131072, must be even number)

ioa_Period - sample period in 279.365 ns increments (124 thru 65536, anti-aliasing filter works below 300 to 500 depending on waveform), if enabled by ADIOF_PERVOL

ioa_Volume - volume (0 thru 64, linear), if enabled by ADIOF_PERVOL

ioa_Cycles - number of times to repeat array (0 thru 65535, 0 for infinite)

ioa_WriteMsg- message replied at start of write, if enabled by ADIOF_WRITEMESSAGE

OUTPUTS

io_Unit - bit map of channel successfully written (bit 0 thru 3 corresponds to channel 0 thru 3)

io_Flags - IOF_QUICK flag cleared if there is no error

io_Error - error number:
 0 - no error
 IOERR_ABORTED - canceled (AbortIO) or channel stolen
 ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

BUGS

If CMD_WRITE starts the write immediately after stopping a previous write, you must set the ADIOF_PERVOL flag or else the new data pointer (ioa_Data) and length (ioa_Length) may not be loaded.

1.20 audio.device/Expunge

NAME

EXPUNGE - indicate a desire to remove the Audio device

FUNCTION

The Expunge routine is called when a user issues a RemDevice call. By

the time it is called, the device has already been removed from the device list, so no new opens will succeed. The existence of any other users of the device, as determined by the device open count being non-zero, will cause the Expunge to be deferred. When the device is not in use, or no longer in use, the Expunge is actually performed.

1.21 audio.device/OpenDevice

NAME

OpenDevice - open the audio device

SYNOPSIS

```
error = OpenDevice("audio.device", unitNumber, iORequest, flags);
```

FUNCTION

The OpenDevice routine grants access to the audio device. It takes an I/O audio request block (iORequest) and if it can successfully open the audio device, it loads the device pointer (io_Device) and the allocation key (ioa_AllocKey); otherwise, it returns an error (IOERR_OPENFAIL). OpenDevice increments the open count keeping the device from being expunged (Expunge). If the length (ioa_Length) is non-zero, OpenDevice tries to allocate (ADCMD_ALLOCATE) audio channels from a array of channel combination options (ioa_Data). If the allocation succeeds, the allocated channel combination is loaded into the unit field (ioa_Unit); otherwise, OpenDevice returns an error (ADIOERR_ALLOCFAILED). OpenDevice does not wait for allocation to succeed and closes (CloseDevice) the audio device if it fails. To allocate channels, OpenDevice also requires a properly initialized reply port (mn_ReplyPort) with an allocated signal bit.

INPUTS

unitNumber- not used
iORequest - pointer to audio request block (struct IOAudio)
 ln_Pri - allocation precedence (-128 thru 127), only necessary for allocation (non-zero length)
mn_ReplyPort- pointer to message port for allocation, only necessary for allocation (non-zero length)
ioa_AllocKey- allocation key; zero to generate new key. Otherwise, it must be set by (or copied from I/O block that is set by) previous OpenDevice function or ADCMD_ALLOCATE command (non-zero length)
ioa_Data - pointer to channel combination options (byte array, bits 0 thru 3 correspond to channels 0 thru 3), only necessary for allocation (non-zero length)
ioa_Length - length of the channel combination option array (0 thru 16), zero for no allocation
flags - not used

OUTPUTS

iORequest - pointer to audio request block (struct IOAudio)
 io_Device - pointer to device node if OpenDevice succeeds, otherwise -1
 io_Unit - bit map of successfully allocated channels (bits

```
                                0 thru 3 correspond to channels 0 thru 3)
io_Error      - error number:
                0                - no error
                IOERR_OPENFAIL    - open failed
                ADIOERR_ALLOCFAILED - allocation failed, no open
ioa_AllocKey- allocation key, set to a unique number if passed
                a zero and OpenDevice succeeds
error         - copy of io_Error
```
