SharedSoundBuffer

Introduction and Overview

SharedSoundBuffer is a module for playing raw data using the SharedSound module. It requires SharedSound version 1.0 or later.

The module has been designed so it is easy to play sounds from any application.

Technical Details

In order to play audio data, SharedSoundBuffer is called in the background by SharedSound. This allows the module to continue in, and out of the desktop. Data may be passed during callbacks, allowing the system as a whole to run independent on any foreground application.

When streaming, an application must feed data to SharedSoundBuffer in a timely fashion. Data is supplied as arbitrary sized blocks, which are consulted in the order in which they were presented to the module. If the player runs out of data it will pause until some more data is supplied. Blocks are copied by SharedSoundBuffer, so they must only be kept for the duration of the call to AddBlock.

When the end of a buffer is reached, the player will continue seamlessly to the next buffer. The data should be frame-aligned, i.e. each block supplied should be a multiple of four bytes, but it is not necessary to ensure that they are a multiple of the SharedSound fill buffer.

A base handle is provided, this is always present and has a stream handle of zero. This does not need to be opened, and cannot be closed, and should be used by programs which only output short, atomic samples. It should not be used by programs which multi-task between adding blocks, otherwise other programs may intersperse their data in with it. It should also not be used by background routines for the same reasons.

SWI calls

SharedSoundBuffer_OpenStream (SWI &55FC0)

Opens a stream

On entry

R0 = flags :

Bit(s)	Meaning
0-31	Reserved, must be 0.

R1 = pointer to name for stream

On exit

R0 = handle of created stream

Interrupts

Interrupts are undefined
Fast interrupts are undefined

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to open a stream prior to passing blocks. The handle returned is a 32 bit opaque word, which will never be zero. The handle zero is reserved for the base handle.

Related SWIs

SWI SharedSoundBuffer_CloseStream (on page 4)
SWI SharedSoundBuffer_AddBlock (on page 5)

SharedSoundBuffer_CloseStream (SWI &55FC1)

Closes and stops a stream immediately

On entry

R0 = stream handle

On exit

None

Interrupts

Interrupts are undefined
Fast interrupts are undefined

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to close a stream, stopping playback and freeing any memory associated with it immediately. This is in contrast with *SWI SharedSoundBuffer_StreamEnd (on page 14)*, which continues playback until all data has drained.

Related SWIs

SWI SharedSoundBuffer_OpenStream (on page 3)
SWI SharedSoundBuffer_StreamEnd (on page 14)

SharedSoundBuffer_AddBlock (SWI &55FC2)

Adds a block to a stream's queue

On entry

R0 = stream handle
R1 = pointer to block
R2 = block size, in bytes

On exit

None

Interrupts

Interrupts are undefined
Fast interrupts are undefined

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to add a block to be queued.

If the queue is full, the error *Error_AudioQueueFull (on page 15)* is returned and the block is not added. It will also clear the appropriate bit in the current poll word.

Related SWIs

SWI SharedSoundBuffer_OpenStream (on page 3)

SharedSoundBuffer_PollWord (SWI &55FC3)

Sets up the buffer pollword

On entry

R0 = stream handle
R1 = flags :

Bit(s)	Meaning
0	**Value** **Effect**
	0 Supply the poll word
	1 Use the poll word specified
1-31	Reserved, must be 0.

R1 = if bit 0 of R0 set:
pointer to word
R2 = if bit 0 of R0 set:
bit number to set

On exit

R0 = if bit 0 of R0 clear on entry:
pointer to opaque poll word

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI sets up the poll word used by the specified stream. The poll word bit will be set when blocks are removed from the buffer, it will be cleared when a block cannot be fitted into the buffer. If the poll word is supplied by the module, it should be treated as an opaque word, and only compared with zero.

The poll word can be changed as often as is necessary.

SharedSoundBuffer_Volume (SWI &55FC4)

Set the volume of output

On entry

R0 = stream handle
R1 = Volume, as two unsigned 16 bit values packed into a 32 bit word
&LLLLRRRR

On exit

None

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call sets the volume of the stream.

Related APIs

None

SharedSoundBuffer_SampleRate (SWI &55FC5)

Set the sample rate

On entry

R0 = stream handle
R1 = new sample rate in 1024th Hz steps

On exit

None

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to set the sample rate of the data being played.

Related APIs

None

SharedSoundBuffer_ReturnSSHandle (SWI &55FC6)

Return the internal SharedSound handle

On entry

R0 = stream handle

On exit

R0 = current SharedSound handle

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to read the low-level SharedSound handle currently in use by the stream. This is a dynamic value, i.e. it may change at any time throughout the lifetime of a stream. Wherever possible you must use the defined SharedSoundBuffer calls.

Related APIs

None

SharedSoundBuffer_SetBuffer (SWI &55FC7)

Set the stream buffer limit

On entry

R0 = stream handle
R1 = buffer limit in bytes

On exit

None

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to set the maximum amount of data which may be buffered for that stream at any time.

Related SWIs

SWI SharedSoundBuffer_BufferStats (on page 12)

SharedSoundBuffer_BufferStats (SWI &52E08)

Find out information about the buffer

On entry

R0 = stream handle

On exit

R0 = number of unplayed bytes

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to find out how many bytes are left unplayed in the buffer.

Related SWIs

SWI SharedSoundBuffer_SetBuffer (on page 11)

SharedSoundBuffer_Pause (SWI &55FC9)

Pauses playback

On entry

R0 = stream handle
R1 = flags :

Bit(s)	Name	Meaning
0	Resume	Resumes playback.
1-31		Reserved, must be 0.

On exit

None

Interrupts

Interrupts are undefined
Fast interrupts are undefined

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to pause or resume playback. While paused, sound output is silenced, but blocks will be retained and may be added, up to the usual limits.

Related APIs

None

SharedSoundBuffer_StreamEnd (SWI &55FCA)

Closes a stream

On entry

R0 = stream handle

On exit

None

Interrupts

Interrupts are undefined
Fast interrupts are undefined

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This SWI is used to close a stream, allowing output to continue and retaining memory buffers until they are finished. The stream handle must not be used after this has been called.

Related SWIs

SWI SharedSoundBuffer_CloseStream (on page 4)

Errors

Error_AudioQueueFull (Error &81A140)

Audio queue full

Use

This error is returned by *SWI SharedSoundBuffer_AddBlock (on page 5)* when the block passed would cause there to be more data buffered than the threshold set with *SWI SharedSoundBuffer_SetBuffer (on page 11)*. The block passed is not added to the queue.

Document information

Maintainer(s): John Duffell <jd@eh.org>

History:

Revision	Date	Author	Changes
1		JD	First XML monitored version ● First XML version of the document.

Related: None

Disclaimer: This document is, to the best of my knowledge, a correct representation of the API of SharedSoundBuffer.