

## **Testfile**

|                      |
|----------------------|
| <b>COLLABORATORS</b> |
|----------------------|

|               |                            |               |                  |
|---------------|----------------------------|---------------|------------------|
|               | <i>TITLE :</i><br>Testfile |               |                  |
| <i>ACTION</i> | <i>NAME</i>                | <i>DATE</i>   | <i>SIGNATURE</i> |
| WRITTEN BY    |                            | July 31, 2024 |                  |

|                         |
|-------------------------|
| <b>REVISION HISTORY</b> |
|-------------------------|

| NUMBER | DATE | DESCRIPTION | NAME |
|--------|------|-------------|------|
|        |      |             |      |

# Contents

|          |                                |          |
|----------|--------------------------------|----------|
| <b>1</b> | <b>Testfile</b>                | <b>1</b> |
| 1.1      | Table Of Contents . . . . .    | 1        |
| 1.2      | MHI/MHIAAllocDecoder . . . . . | 1        |
| 1.3      | MHI/MHIFreeDecoder . . . . .   | 2        |
| 1.4      | MHI/MHIGetEmpty . . . . .      | 3        |
| 1.5      | MHI/MHIGetStatus . . . . .     | 3        |
| 1.6      | MHI/MHIPause . . . . .         | 4        |
| 1.7      | MHI/MHIPlay . . . . .          | 5        |
| 1.8      | MHI/MHIQuery . . . . .         | 5        |
| 1.9      | MHI/MHIQueueBuffer . . . . .   | 7        |
| 1.10     | MHI/MHISetParam . . . . .      | 8        |
| 1.11     | MHI/MHIStop . . . . .          | 9        |

# Chapter 1

## Testfile

### 1.1 Table Of Contents

#### TABLE OF CONTENTS

MHIAAllocDecoder  
MHIFreeDecoder  
MHIGetEmpty  
MHIGetStatus  
MHIPause  
MHIPlay  
MHIQuery  
MHIQueueBuffer  
MHISetParam  
MHIStop

### 1.2 MHI/MHIAAllocDecoder

#### NAME

MHIAAllocDecoder - allocate a decoder

#### SYNOPSIS

```
handle = MHIAAllocDecoder(task, mhisignal);  
d0          a0      d0
```

```
APTR MHIAAllocDecoder (Task *, ULONG mhisignal);
```

#### FUNCTION

Allocate a decoder. This function gives you a handle in d0 which is required by other MHI functions. The handle is private and it's contents will vary with each decoder, so don't poke it. Note that some decoders may allow more than one handle to be allocated, meaning that they can output more than one MPEG stream at once.

When you allocate a handle all resources required by the decoder are also allocated (memory, hardware etc). The decoder is then ready to start decoding.

---

## INPUTS

task - a pointer to the task you want to receive signals from MHI  
mhisignal - a signal mask to use when signaling your task

## RESULT

handle - a pointer to your handle

## EXAMPLE

```
/* Note: use more error checking! */  
/* See example code */  
mytask = FindTask(0);  
mysignal = AllocSignal(-1);  
sigmask = 1L << mysignal;  
handle = MHIAallocDecoder(mytask, sigmask);
```

## NOTES

You must dispose of your handle when done with it.

## BUGS

## SEE ALSO

MHIFreeDecoder

## 1.3 MHI/MHIFreeDecoder

## NAME

MHIFreeDecoder - free a decoder up

## SYNOPSIS

```
MHIFreeDecoder(handle);  
a3
```

```
VOID MHIFreeDecoder(APTR handle);
```

## FUNCTION

Free the given handle. You must call this function when you are finished with a handle and wish to free the decoder up.

## INPUTS

handle - the handle you were allocated

## RESULT

None

## EXAMPLE

```
MHIFreeDecoder(handle);
```

## NOTES

No checking is done, so don't pass a bad handle.

## BUGS

## SEE ALSO

MHIAAllocDecoder

## 1.4 MHI/MHIGetEmpty

### NAME

MHIGetEmpty

### SYNOPSIS

```
buffer = MHIGetEmpty(handle);  
d0      a3
```

```
APTR MHIGetEmpty(APTR handle);
```

### FUNCTION

Find the next empty buffer in the queue. If there are any empty buffers left the first one is freed and removed from the queue, and a pointer to it returned to you. You are then free to do what you like with that buffer (e.g. FreeMem() it, load more data into it etc).

### INPUTS

handle - the handle you were allocated

### RESULT

buffer - a pointer to the used buffer

### EXAMPLE

```
while (usedbuf = MHIGetEmpty(handle))  
{  
    ...  
}
```

### NOTES

Requires a valid handle. You should use this function in a loop after you receive a signal from MHI, just as you would when checking a Message Port. The buffer that is freed is considered 'empty' and as such you cannot rely on it's contents at all. It may however contain data, as it is not automatically cleared.

### BUGS

### SEE ALSO

MHIQueueBuffer

## 1.5 MHI/MHIGetStatus

### NAME

MHIGetStatus

### SYNOPSIS

```
status = MHIGetStatus(handle);  
d0      a3
```

```
UBYTE MHIGetStatus(APTR handle);
```

#### FUNCTION

Return the current status of the MHI decoder. This give you information about what the decoder is doing, and if it has stalled (MHIF\_OUT\_OF\_DATA).

#### INPUTS

handle - the handle you were allocated

#### RESULT

status - one of: MHIF\_PLAYING (player currently outputting sound)  
          MHIF\_STOPPED (doing nothing)  
          MHIF\_OUT\_OF\_DATA (run out of data but still waiting for more)  
          MHIF\_PAUSED (play currently paused but can be restarted)

#### EXAMPLE

```
status = MHIGetStatus(handle);
```

#### NOTES

You can use this to check if an MPEG stream has finished decoding by waiting for buffers to be returned to you and then checking if status = MHIF\_OUT\_OF\_DATA. See example code.

#### BUGS

#### SEE ALSO

MHIPlay  
MHISop  
MHIPause

## 1.6 MHI/MHIPause

#### NAME

MHIPause

#### SYNOPSIS

```
MHIPause(handle);  
a3
```

```
VOID MHIPause(APTR handle);
```

#### FUNCTION

Halt decoding and audio output. The buffer queue is not altered and play may begin again at exactly where it left off.

#### INPUTS

handle - the handle you were allocated

#### RESULT

None

---

## EXAMPLE

```
MHIPause(handle);
```

## NOTES

Use MHIPlay to start decoding again. MHISop can also be used even if the decoder is paused.

## BUGS

## SEE ALSO

MHIGetStatus  
MHIPlay  
MHISop

## 1.7 MHI/MHIPlay

## NAME

MHIPlay

## SYNOPSIS

```
MHIPlay(handle);  
a3
```

```
VOID MHIPlay(APTR handle);
```

## FUNCTION

Set the MHI decoder into play mode. The decoder starts decoding the first buffer in the queue.

## INPUTS

handle - the handle you were allocated

## RESULT

None

## EXAMPLE

```
MHIPlay(handle);
```

## NOTES

You may call this function without any buffers in the queue, but it is usually best to buffer some data before starting the decoding.

## BUGS

## SEE ALSO

MHIGetStatus  
MHISop  
MHIPause

## 1.8 MHI/MHIQuery

---



## NAME

MHIQuery

## SYNOPSIS

```
result = MHIQuery(query);  
d0      d1
```

```
ULONG MHIQuery(ULONG query);
```

## FUNCTION

Query some aspect of the decoder. A complete list of available queries can be found in mhi.h.

MHIQ\_DECODER\_NAME

MHIQ\_DECODER\_VERSION

MHIQ\_AUTHOR

Return a string pointer to the name of the decoder/author/version string.

MHIQ\_IS\_HARDWARE

MHIF\_TRUE if decoder is hardware, MHIF\_FALSE if it's software based.

MHIQ\_IS\_68K

MHIQ\_IS\_PPC

Same as MHIQ\_IS\_HARDWARE for 68k/PPC processor based decoders.

MHIQ\_MPEG1

MHIQ\_MPEG2

MHIQ\_MPEG25

MHIQ\_MPEG4

Return MHIF\_TRUE if MPEG version is supported, or MHIF\_FALSE if not.

MHIQ\_LAYER1

MHIQ\_LAYER2

MHIQ\_LAYER3

Return MHIF\_TRUE if MPEG layer is supported, or MHIF\_FALSE if not.

MHIQ\_VARIABLE\_BITRATE

MHIQ\_JOINT\_STEREO

Return MHIF\_TRUE if encoding format is supported, or MHIF\_FALSE if not.

MHIQ\_BASS\_CONTROL

MHIQ\_TREBLE\_CONTROL

MHIQ\_MID\_CONTROL

Return MHIF\_TRUE if tone control is supported, or MHIF\_FALSE if not.

MHIQ\_VOLUME\_CONTROL

MHIQ\_PANNING\_CONTROL

MHIQ\_CROSSMIXING

Return MHIF\_TRUE if output control is supported, or

MHIF\_FALSE if not.

#### INPUTS

query - an MHIQ\_#? query flag.

#### RESULT

result - the result code, see mhi.h and above

#### EXAMPLE

```
/* see if decoder support variable bit rates */  
result = MHIQuery(MHIQ_VARIABLE_BITRATE);
```

#### NOTES

#### BUGS

#### SEE ALSO

## 1.9 MHI/MHIQueueBuffer

#### NAME

MHIQueueBuffer - add a memory buffer to the decoder queue

#### SYNOPSIS

```
success = MHIQueueBuffer(handle, buffer, size);  
d0                a3      a0      d0
```

```
BOOL MHIQueueBuffer(APTR handle, APTR buffer, ULONG size);
```

#### FUNCTION

Add a buffer to the decoder queue. Once the buffer is in the queue you are not allowed to alter it in any way until it is released by MHI.

#### INPUTS

handle - the handle you were allocated  
buffer - a pointer to the start of the buffer to be queued  
size - the byte size of the buffer

#### RESULT

success - TRUE (-1) when buffer was queued or FALSE (0) when for some reason the buffer could not be queued

#### EXAMPLE

```
MHIQueueBuffer(handle, bufmem, size);
```

#### NOTES

Requires a valid handle.

#### BUGS

#### SEE ALSO

MHIGetEmpty

---

## 1.10 MHI/MHISetParam

### NAME

MHISetParam

### SYNOPSIS

```
MHISetParam(handle, param, value);
```

```
VOID MHISetParam(APTR handle, UWORD param, ULONG value);  
                a3          d0          d1
```

### FUNCTION

Alter one of the decoder parameters. Commonly used to set volume, bass, treble, panning etc. A complete list of parameters can be found in mhi.h. Details of current parameters:

#### MHIP\_VOLUME

Overall sound volume. 100 is max, 0 is silence.

#### MHIP\_PANNING

Sound panning. 50 is centre (default), 0 is full left and 100 is full right.

#### MHIP\_CROSSMIXING

Crossmixing is where some of the sound from the left channel is mixed onto the right channel, and vice versa. It is often used to lessen the stereo effect for people using headphones. Also similar to 'surround' sound effects used in some software. 0 is no mixing (default) and 100 is maximum (effectively mono).

#### MHIP\_BASS

#### MHIP\_MID

#### MHIP\_TREBLE

These parameters control tone. Bass is low frequencies, and treble is high. Mid is everything in between. MHI does not specify the exact frequency ranges that these cover, as it can vary with hardware for instance. 50 is the default, with no tone modification. 100 is maximum boost, and 0 is is maximum cut.

#### MHIP\_PREFACTOR

Prefactor allows sound levels to be boosted or reduced before it goes through tone control. This is useful to prevent 'chopping', where the signal is boosted too much by tone control and distorts. It is not the same as volume. The default is 50, which is no prefactor. 100 is maximum cut, where sound levels are lowered, and 0 is maximum boost.

### INPUTS

|        |                                   |
|--------|-----------------------------------|
| handle | - the handle you were allocated   |
| param  | - the parameter you want to alter |
| value  | - the value to set                |

?

### RESULT

---

Parameter is set if value is valid

#### EXAMPLE

```
/* pump up the bass man */  
MHISetParam(handle, MHIP_BASS, 90);
```

?

#### NOTES

#### SEE ALSO

## 1.11 MHI/MHIStop

#### NAME

MHIStop

#### SYNOPSIS

```
MHIStop(handle);  
a3
```

```
VOID MHIStop(APTR handle);
```

#### FUNCTION

Stop all decoding and audio output. All buffers in the queue are flushed.

#### INPUTS

handle - the handle you were allocated

#### RESULT

None

#### EXAMPLE

```
MHIStop(handle);
```

#### NOTES

This function will flush the buffer queue. Use MHIPause if you want to resume play where you left off later without emptying the buffer queue.

#### BUGS

#### SEE ALSO

MHIGetStatus  
MHIPlay  
MHIPause

---