BASSMOD_ErrorGetCode

Retrieves the BASSMOD error code for the most recent BASSMOD function call.

```
DWORD WINAPI BASSMOD_ErrorGetCode();
```

Return value
If no error occured during the last BASSMOD function call then BASS_OK is returned, else one of the BASS_ERROR values is returned. See the function description for an explanation of what the error code means.

BASSMOD_Free

Frees all resources used by the digital output, including the MOD music.

```
void WINAPI BASSMOD_Free();
```

Remarks
BASSMOD_Free should be called before your program exits. It's not necessary to individually free the MOD music as it is automatically freed by this function.

If you wish to change device settings, having already called BASSMOD_Init, then BASSMOD_Free must be called before calling BASSMOD_Init again. You will also have to reload the MOD music.

See also
BASSMOD_Init

BASSMOD_GetCPU

Retrieves the current CPU usage of BASSMOD.

```
float WINAPI BASSMOD_GetCPU();
```

Return value
The BASSMOD CPU usage as a percentage of total CPU time.

BASSMOD_GetDeviceDescription

Retrieves the text description of a device.

```
char *WINAPI BASSMOD_GetDeviceDescription(
    int devnum
);
```

Parameters
devnum The device to get the description of... 0 = first.

Return value
If succesful, then a pointer to the description is returned, else NULL is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_DEVICE The device number specified is invalid.

Remarks
This function can be used to enumerate the available devices for a setup dialog.

Linux notes
This function will always return NULL in the Linux version.

Example
To get the total number of devices present.

```
int count=0; // the device counter
while (BASSMOD_GetDeviceDescription(count)) count++;
```

BASSMOD_GetVersion

Retrieves the version number of the BASSMOD.DLL that is loaded.

```
DWORD WINAPI BASSMOD_GetVersion();
```

Return value
The BASSMOD version (LOWORD.HIWORD)

Remarks
There is no guarantee that a previous or future version of BASSMOD supports all the BASSMOD functions that you are using, so you should always call this function to make sure the correct version is loaded.

Example
To check that BASSMOD 2.0 is loaded.

```
if (BASSMOD_GetVersion()!=MAKELONG(2,0)) {
    // version 2.0 not loaded!
}
```

BASSMOD_GetVolume

Retrieves the current volume level.

```
int WINAPI BASSMOD_GetVolume();
```

Return value
If successful, the volume level is returned, else -1 is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_INIT	BASSMOD_Init has not been successfully called. This error is also returned when trying to get the volume level of the "no sound" or "decode only" device.
BASS_ERROR_DRIVER *Linux only*	BASSMOD could not get access to the mixer.

See also
BASSMOD_SetVolume

BASSMOD_Init

Initializes BASSMOD.

```
BOOL WINAPI BASSMOD_Init(
    int device,
    DWORD freq,
    DWORD flags
);
```

Parameters

device	The device to use... 0 = first, -1 = default, -2 = no sound, -3 = decode only (see remarks). BASSMOD_GetDeviceDescription can be used to get the total number of devices.	
freq	Output sample rate.	
flags	Any combination of these flags.	
	BASS_DEVICE_8BITS	Use 8 bit resolution, else 16 bit.
	BASS_DEVICE_MONO	Use mono, else stereo.
	BASS_DEVICE_NOSYNC	Disable synchronizers. If you are not using any syncs, then you may as well use this flag to save a little CPU time. This is automatic when using the "decode only" device.

Return value
If BASSMOD was successfully initialized then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_ALREADY	BASSMOD has already been initialized. You must call BASSMOD_Free before calling this function again.
BASS_ERROR_DEVICE	The device number specified is invalid.
BASS_ERROR_DRIVER	There is no available device driver... the device may already be in use.
BASS_ERROR_FORMAT	The specified format is not supported by the device. Try changing the *freq* and *flags* parameters.
BASS_ERROR_MEM	There is insufficent memory.

Remarks
This function must be successfully called before calling any other BASSMOD functions, except BASSMOD_ErrorGetCode, BASSMOD_GetDeviceDescription and BASSMOD_GetVersion.

When using the "decode only" device (*device* = *-3*), BASSMOD only decodes the sample data (via BASSMOD_MusicDecode), without playing it. This allows the data to be outputted in any way wanted, for example, writing to disk. As no playback (and therefore syncing too) is performed, no decoding or syncing threads are created when using this "device".

Linux notes
device -1 = "/dev/dsp", *device 0* = "/dev/dsp0", *device 1* = "/dev/dsp1", etc...

Example
To initialize BASSMOD, falling back to no sound if no device is available.

```
// try initializing the default device, at 44100hz stereo 16 bits
if (!BASSMOD_Init(-1,44100,0)) {
    // couldn't initialize device, so use no sound
    BASSMOD_Init(-2,44100,0)
}
```

See also
BASSMOD_Free, BASSMOD_MusicLoad, BASSMOD_SetVolume

BASSMOD_SetVolume

Sets the digital output master volume.

```
BOOL WINAPI BASSMOD_SetVolume(
    DWORD volume
);
```

Parameters
volume The volume level... 0 (min) - 100 (max).

Return value
If succesful, then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_INIT	BASSMOD_Init has not been successfully called. This error is also returned when trying to set the volume level of the "no sound" or "decode only" device.
BASS_ERROR_DRIVER	BASSMOD could not get access to the mixer. *Linux only*

See also
BASSMOD_GetVolume, BASSMOD_MusicSetVolume

BASSMOD_MusicDecode

Gets decoded sample data from the MOD music.

```
DWORD WINAPI BASSMOD_MusicDecode(
    void *buffer,
    DWORD length
);
```

Parameters
buffer Location to write the decoded data.
length Number of bytes wanted.

Return value
If an error occurs, -1 is returned, use BASSMOD_ErrorGetCode to get the error code. If successful, the number of bytes actually decoded will be returned.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.
BASS_ERROR_NOTAVAIL The "decode only" device was not specified in the BASSMOD_Init call.
BASS_ERROR_NOPLAY The MOD music has reached the end.

Remarks
The returned sample data is in the standard Windows PCM format: 8-bit samples are unsigned, 16-bit samples are signed. There are no intermediate buffers involved, so as much data as is available can be decoded in one go.

Example
Decode 10000 bytes of sample data.

```
BYTE buf[10000]; // buffer
BASSMOD_MusicDecode(buf,10000);
```

See also
BASSMOD_MusicIsActive

BASSMOD_MusicFree

Frees the MOD music's resources.

```
void WINAPI BASSMOD_MusicFree();
```

See also
BASSMOD_MusicLoad

BASSMOD_MusicGetLength

Retrieves the length of the MOD music.

```
DWORD WINAPI BASSMOD_MusicGetLength(
    BOOL playlen
);
```

Parameters
playlen The length to retrieve... TRUE = the playback length (in bytes), FALSE = the order length.

Return value
If succesful, then the music's length is returned, else -1 is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.
BASS_ERROR_ILLPARAM The BASS_MUSIC_CALCLEN flag was not used with BASSMOD_MusicLoad, or the playback length could not be calculated (the music does not end).

Example
To start playback of the MOD music from the beginning of the last order.

```
DWORD len=BASSMOD_MusicGetLength(FALSE); // get length
BASSMOD_MusicPlayEx(MAKELONG(len-1,0),-1,TRUE); // play
```

See also
BASSMOD_MusicSetPosition, BASSMOD_MusicPlayEx

BASSMOD_MusicGetName

Retrieves the MOD music's name.

```
char *WINAPI BASSMOD_MusicGetName();
```

Return value
If succesful, then a pointer to the music's name is returned, else NULL is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.

BASSMOD_MusicGetPosition

Retrieves the playback position of the MOD music.

```
DWORD WINAPI BASSMOD_MusicGetPosition();
```

Return value
If an error occurs, -1 is returned, use BASSMOD_ErrorGetCode to get the error code. If successful, the position is returned as follows... LOWORD = order, HIWORD = row * scaler (see BASSMOD_MusicSetPositionScaler).

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.

See also
BASSMOD_MusicIsActive, BASSMOD_MusicSetPosition

BASSMOD_MusicGetVolume

Retrieves the volume level of a channel or instrument in a MOD music.

```
DWORD WINAPI BASSMOD_MusicGetVolume(
    DWORD chanins
);
```

Parameters
chanins The channel or instrument to retrieve the volume of... if the HIWORD is 0, then the LOWORD is a channel number (0 = 1st channel), else the LOWORD is an instrument number (0 = 1st instument).

Return value
If succesful, then the requested volume level is returned, else -1 is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.
BASS_ERROR_NOTAVAIL *chanins* is not valid.

Remarks
This function can also be used to count the number of channels and instruments in a MOD Music.

Example
Count the number of channels and instruments in a MOD music.

```
int channels=0,instruments=0;
while (BASSMOD_MusicGetVolume(channels)!=-1) channels++;
while (BASSMOD_MusicGetVolume(MAKELONG(instruments,1))!=-1) instruments++;
```

See also
BASSMOD_MusicSetVolume

BASSMOD_MusicIsActive

Checks if the MOD music is active (playing).

```
DWORD WINAPI BASSMOD_MusicIsActive();
```

Return value
The return value is one of the folowing.
BASS_ACTIVE_STOPPED The MOD music is not active.
BASS_ACTIVE_PLAYING The MOD music is playing.
BASS_ACTIVE_PAUSED The MOD music is paused.

Remarks
When using the "decode only" device, BASS_ACTIVE_PLAYING will be returned until the end of the MOD music is reached, when BASS_ACTIVE_STOPPED will be returned.

BASSMOD_MusicLoad

Loads a MOD music.

```
BOOL WINAPI BASSMOD_MusicLoad(
    BOOL mem,
    void *file,
    DWORD offset,
    DWORD length,
    DWORD flags
);
```

Parameters

mem	TRUE = load the MOD music from memory.	
file	Filename (mem = FALSE) or a memory location (mem = TRUE).	
offset	File offset to load the MOD music from (only used if mem = FALSE).	
length	Data length (only used if mem = FALSE)... 0 = use all data up to the end of file. If *length* over-runs the end of the file, it'll automatically be lowered to the end of the file.	
flags	A combination of these flags.	
	BASS_MUSIC_LOOP	Loop the music.
	BASS_MUSIC_RAMP	Use "normal" ramping (as used in FastTracker 2).
	BASS_MUSIC_RAMPS	Use "sensitive" ramping.
	BASS_MUSIC_NONINTER	Use non-interpolated mixing. This generally reduces the sound quality, but can be good for chip-tunes.
	BASS_MUSIC_FT2MOD	Play .MOD file as FastTracker 2 would.
	BASS_MUSIC_PT1MOD	Play .MOD file as ProTracker 1 would.
	BASS_MUSIC_POSRESET	Stop all notes when moving position (using BASSMOD_MusicSetPosition or BASSMOD_MusicPlayEx).
	BASS_MUSIC_SURROUND	Apply XMPlay's surround sound to the music (ignored in mono).
	BASS_MUSIC_SURROUND2	Apply XMPlay's surround sound mode 2 to the music (ignored in mono).
	BASS_MUSIC_STOPBACK	Stop the music when a backward jump effect is played. This stops musics that never reach the end from going into endless loops. Some MOD musics are designed to jump all over the place, so this flag would cause those to be stopped prematurely. If this flag is used together with the BASS_MUSIC_LOOP flag, then the music would not be stopped but any BASS_SYNC_END sync would be called.
	BASS_MUSIC_CALCLEN	Calculate the playback length of the music. This also slightly increases the time taken to load the music, depending on how long it is. Use BASSMOD_MusicGetLength to retrieve the calculated length. Note that it's not always possible to calculate a length because some musics never actually reach an end.
	BASS_MUSIC_NOSAMPLE	Don't load the music's samples. This slightly reduces the time taken to load the music, which is useful if you just want to get the name and length of the music without playing it.
	BASS_UNICODE *Win32 only*	*file* is a Unicode (16-bit characters) filename.

Return value
If successful, then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_INIT	BASSMOD_Init has not been successfully called.
BASS_ERROR_ALREADY	A MOD music has already been loaded, you must call BASSMOD_MusicFree first.
BASS_ERROR_FILEOPEN	The file could not be opened.
BASS_ERROR_FILEFORM	The file's format is not recognised/supported.
BASS_ERROR_MEM	There is insufficent memory.

Remarks
BASS uses the same code as XMPlay for it's MOD music support, giving the most accurate reproduction of IT / XM / S3M / MTM / MOD / UMX files available from any sound system.

Ramping doesn't take a lot of extra processing and improves the sound quality by removing "clicks". Sensitive ramping leaves sharp attacked samples, while normal ramping can cause them to lose a bit of their impact. Generally, normal ramping is recommended for XMs, and sensitive ramping for the other formats. But, some XMs may also sound better using sensitive ramping.

When loading a MOD music from memory, BASS does not use the memory after it's loaded the MOD music. So you can do whatever you want with the memory after calling this function.

PocketPC notes
All filenames are Unicode on PocketPC, so *file* is always Unicode (if not loading from memory).

See also
BASSMOD_MusicDecode, BASSMOD_MusicFree, BASSMOD_MusicGetLength, BASSMOD_MusicGetName, BASSMOD_MusicPlay, BASSMOD_MusicPlayEx, BASSMOD_MusicSetAmplify, BASSMOD_MusicSetPanSep, BASSMOD_MusicSetPositionScaler, BASSMOD_MusicSetSync

BASSMOD_MusicPause

Pauses the MOD music.

```
BOOL WINAPI BASSMOD_MusicPause();
```

Return value
If successful, TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOPLAY The MOD music is not playing.

Remarks
Use BASSMOD_MusicPlay to resume playback. BASSMOD_MusicStop can be used to stop the paused MOD music.

Linux notes
Pausing is not instantaneous in the Linux version.

See also
BASSMOD_MusicPlay, BASSMOD_MusicStop

BASSMOD_MusicPlay

Plays the MOD music.

```
BOOL WINAPI BASSMOD_MusicPlay();
```

Return value
If successful, TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.

Remarks
Playback continues from where it was last stopped or paused. If the MOD music has just been loaded, then playback starts from the beginning.

See also
BASSMOD_MusicGetPosition, BASSMOD_MusicIsActive, BASSMOD_MusicPause, BASSMOD_MusicStop, BASSMOD_MusicLoad, BASSMOD_MusicPlayEx

BASSMOD_MusicPlayEx

Plays the MOD music, using the specified start position and flags.

```
BOOL WINAPI BASSMOD_MusicPlayEx(
    DWORD pos,
    int flags,
    BOOL reset
);
```

Parameters

pos — Position to start playback from... LOWORD = order, HIWORD = row. If HIWORD = 0xFFFF, then LOWORD = position in seconds. If LOWORD and HIWORD are both 0xFFFF, then the position is left unchanged. Setting the position in seconds requires that the BASS_MUSIC_CALCLEN flag was used when the MOD music was loaded.

flags — Override the MOD music's current flags... -1 = use current flags, else a combination of these flags.

BASS_MUSIC_LOOP	Loop the music.
BASS_MUSIC_RAMP	Use "normal" ramping (as used in FastTracker 2).
BASS_MUSIC_RAMPS	Use "sensitive" ramping.
BASS_MUSIC_NONINTER	Use non-interpolated mixing. This generally reduces the sound quality, but can be good for chip-tunes.
BASS_MUSIC_FT2MOD	Play .MOD file as FastTracker 2 would.
BASS_MUSIC_PT1MOD	Play .MOD file as ProTracker 1 would.
BASS_MUSIC_POSRESET	Stop all notes when moving position (using BASSMOD_MusicSetPosition or this function).
BASS_MUSIC_SURROUND	Apply XMPlay's surround sound to the music (ignored in mono).
BASS_MUSIC_SURROUND2	Apply XMPlay's surround sound mode 2 to the music (ignored in mono).
BASS_MUSIC_STOPBACK	Stop the music when a backward jump effect is played. This stops musics that never reach the end from going into endless loops. Some MOD musics are designed to jump all over the place, so this flag would cause those to be stopped prematurely.

reset — TRUE = Stop all playing notes and reset BPM, etc... This is ignored if not also setting the position.

Return value

If successful, TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_NOMUSIC	A MOD music has not been loaded.
BASS_ERROR_DECODE	The "decode only" device is being used, so the MOD music is not playable (*pos* must be *-1*).
BASS_ERROR_POSITION	*pos* is invalid.

Remarks

When the position is left unchanged (*pos* = *-1*), this function does not start the music playing, but it will continue playing if it is already. This allows a music's *flags* (ramping, etc...) to be changed at any time.

Example

To reset and start playback of the MOD music at row 10 of order 5.

```
BASSMOD_MusicPlayEx(MAKELONG(5,10),-1,TRUE);
```

See also

BASSMOD_MusicGetPosition, BASSMOD_MusicIsActive, BASSMOD_MusicPause, BASSMOD_MusicStop, BASSMOD_MusicGetLength, BASSMOD_MusicLoad, BASSMOD_MusicPlay

BASSMOD_MusicRemoveSync

Removes a synchronizer from the MOD music.

```
BOOL WINAPI BASSMOD_MusicRemoveSync(
    HSYNC sync
);
```

Parameters
sync Handle of the synchronizer to remove.

Return value
If succesful, TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.
BASS_ERROR_HANDLE *sync* is not valid.

See also
BASSMOD_MusicSetSync, SYNCPROC callback

BASSMOD_MusicSetAmplify

Sets the MOD music's amplification level.

```
BOOL WINAPI BASSMOD_MusicSetAmplify(
    DWORD amp
);
```

Parameters
amp Amplification level... 0 (min) - 100 (max)... the default when a MOD music is loaded is 50.

Return value
If succesful, then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.

Remarks
As the amplification level get's higher, the sample data's range increases, and therefore, the resolution increases. But if the level is set too high, then clipping can occur, which can result in distortion of the sound.

See also
BASSMOD_MusicSetPanSep

BASSMOD_MusicSetPanSep

Sets the MOD music's pan seperation level.

```
BOOL WINAPI BASSMOD_MusicSetPanSep(
    DWORD pan
);
```

Parameters
pan Pan seperation... 0 (min) - 100 (max), 50 = linear (which is the default when a MOD music is loaded).

Return value
If succesful, then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.

Remarks
By default BASSMOD uses a linear panning "curve". If you want to use the panning of FT2, use a pan seperation setting of around 35. To use the Amiga panning (ie. full left and right) set it to 100.

See also
BASSMOD_MusicSetAmplify

BASSMOD_MusicSetPosition

Sets the playback position of the MOD music.

```
BOOL WINAPI BASSMOD_MusicSetPosition(
    DWORD pos
);
```

Parameters

pos The position... LOWORD = order, HIWORD = row. If HIWORD = 0xFFFF, then LOWORD = position in seconds. Setting the position in seconds requires that the BASS_MUSIC_CALCLEN flag was used when the MOD music was loaded.

Return value

If succesful, then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_NOMUSIC	A MOD music has not been loaded.
BASS_ERROR_POSITION	The requested position is illegal.

Remarks

When the BASS_MUSIC_POSRESET flag is active, all notes that were playing before the position changed will be stopped. Otherwise, the notes will continue playing until they are stopped in the MOD music. When setting the position in seconds, the BPM & tempo are updated to what they would normally be at the new position. Otherwise they are left as they were prior to the postion change.

Example

To set the position of the MOD music to row 20 of order 10.

```
BASSMOD_MusicSetPosition(MAKELONG(10,20));
```

See also

BASSMOD_MusicGetPosition, BASSMOD_MusicIsActive, BASSMOD_MusicGetLength

BASSMOD_MusicSetPositionScaler

Sets the MOD music's BASSMOD_MusicGetPosition scaler.

```
BOOL WINAPI BASSMOD_MusicSetPositionScaler(
    DWORD scale
);
```

Parameters
scale The scaler... 1 (min) - 256 (max)... the default when a MOD music is loaded is 1.

Return value
If succesful, then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.

Remarks
When you call BASSMOD_MusicGetPosition, the row (HIWORD) will be scaled by this value. By using a higher scaler, you can get a more precise position indication.

Example
To get the position of the MOD music accurate to within a 10th of a row.

```
DWORD pos,order,row,row10th;
BASSMOD_MusicSetPositionScaler(10); // set the scaler
pos=BASSMOD_MusicGetPosition();
order=LOWORD(pos); // the order
row=HIWORD(pos)/10; // the row
row10th=HIWORD(pos)%10; // the 10th of a row
```

See also
BASSMOD_MusicGetPosition

BASSMOD_MusicSetSync

Sets up a synchronizer on the MOD music.

```
HSYNC WINAPI BASSMOD_MusicSetSync(
    DWORD type,
    DWORD param,
    SYNCPROC *proc,
    DWORD user
);
```

Parameters

type	The type of sync... see the table below. If you want the sync to occur only once, then also use the BASS_SYNC_ONETIME flag.
param	The sync parameters, depends on the sync *type*... see the table below.
proc	The callback function.
user	User instance data to pass to the callback function.

Sync types, with ***param*** and SYNCPROC ***data*** definitions.

BASS_SYNC_POS	Sync when the music reaches a position. ***param*** : LOWORD = order (0=first, -1=all), HIWORD = row (0=first, -1=all). ***data*** : LOWORD = order, HIWORD = row.
BASS_SYNC_END	Sync when the music reaches the end. Note that some MOD musics never reach the end, they may jump to another position first. If the BASS_MUSIC_STOPBACK flag is used with a MOD music (through BASSMOD_MusicLoad or BASSMOD_MusicPlayEx), then this sync will also be called when a backward jump effect is played. ***param*** : not used. ***data*** : 1 = the sync is triggered by a backward jump in a MOD music, otherwise not used.
BASS_SYNC_MUSICINST	Sync when an instrument (sample for the MOD/S3M/MTM formats) is played (not including retrigs). ***param*** : LOWORD = instrument (1=first), HIWORD = note (0=c0...119=b9, -1=all). ***data*** : LOWORD = note, HIWORD = volume (0-64).
BASS_SYNC_MUSICFX	Sync when the sync effect is used. The sync effect is E8x or Wxx for the XM/MTM/MOD formats, and S2x for the IT/S3M formats (where x = any value). ***param*** : 0 = the position is passed to the callback (***data*** : LOWORD = order, HIWORD = row), 1 = the value of x is passed to the callback (***data*** : x value).

Return value
If succesful, then the new synchronizer's handle is returned, else 0 is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes

BASS_ERROR_NOMUSIC	A MOD music has not been loaded.
BASS_ERROR_NOSYNC	Syncs are disabled, due to the BASS_DEVICE_NOSYNC flag being used in the BASSMOD_Init call.
BASS_ERROR_ILLPARAM	An illegal *param* was specified.
BASS_ERROR_ILLTYPE	An illegal *type* was specified.

Remarks
Multiple synchronizers may be used. Use BASSMOD_MusicRemoveSync to remove a synchronizer. If the BASS_SYNC_ONETIME flag is used, then the sync is automatically removed after it's occured (ie. there's no need to remove it manually).

The MOD music does not have to be playing to set a synchronizer, you can set synchronizers before or while playing the music. Equally, you can also remove synchronizers at any time.

Example
Do some processing until the MOD music reaches the 10th order.

```
BOOL order10=FALSE; // the order 10 flag
...
// the sync callback
void CALLBACK MySyncProc(HSYNC handle, DWORD data, DWORD user) {
```

```
    order10=TRUE; // set the order 10 flag
}
...
BASS_MusicSetSync(BASS_SYNC_POS|BASS_SYNC_ONETIME, MAKELONG(10,0), &MySyncProc, 0); //
set the one-time order 10 sync
while (!order10) {
    // order 10 has not arrived, so do some processing
}
// order 10 has arrived!
```

See also
BASSMOD_MusicRemoveSync, SYNCPROC callback

BASSMOD_MusicSetVolume

Sets the volume level of a channel or instrument in a MOD music.

```
BOOL WINAPI BASSMOD_MusicSetVolume(
    DWORD chanins,
    DWORD volume
);
```

Parameters
chanins The channel or instrument to set the volume of... if the HIWORD is 0, then the LOWORD is a channel number (0 = 1st channel), else the LOWORD is an instrument number (0 = 1st instument).
volume The volume level... 0 (min) - 100 (max).

Return value
If succesful, then TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.
BASS_ERROR_NOTAVAIL *chanins* is not valid.
BASS_ERROR_ILLPARAM *volume* is not valid.

Remarks
The effect of changes made with this function are not heard instantaneously, due to buffering. The volume level of all channels and instruments is initially 100. For MOD formats that do not use instruments, read "sample" for "instrument".

See also
BASSMOD_MusicGetVolume, BASSMOD_MusicSetAmplify

BASSMOD_MusicStop

Stops the MOD music.

```
BOOL WINAPI BASSMOD_MusicStop();
```

Return value
If successful, TRUE is returned, else FALSE is returned. Use BASSMOD_ErrorGetCode to get the error code.

Error codes
BASS_ERROR_NOMUSIC A MOD music has not been loaded.

See also
BASSMOD_MusicPlay, BASSMOD_MusicPlayEx

SYNCPROC callback

User defined synchronizer callback function.

```
void CALLBACK YourSyncProc(
    HSYNC handle,
    DWORD data
    DWORD user
);
```

Parameters

handle	The sync that has occured.
data	Additional data associated with the sync's occurance.
user	The user instance data given when BASSMOD_MusicSetSync was called.

Remarks

A sync callback function should be very quick as other syncs can't be processed until it has finished.

See also

BASSMOD_MusicSetSync