Maxmod Programming Reference

Function Reference

Initialization/Main Functions

mmInitDefault

-

Prototype

```
void mmInitDefault( char* soundbank );
void mmInitDefault( mm_addr soundbank, mm_word number_of_channels );
```

Parameters

soundbank (ds)

Filename of soundbank. A soundbank file can be created with the Maxmod Utility.

soundbank (gba)

Memory address of soundbank (in ROM). A soundbank file can be created with the Maxmod Utility.

number_of_channels

Number of module/mixing channels to allocate. Must be greater or equal to the channel count in your modules.

Description

Initialize Maxmod with default settings.

For GBA, this function uses these default settings (and allocates memory): 16KHz mixing rate, channel buffers in EWRAM, wave buffer in EWRAM, and mixing buffer in IWRAM. It also links the VBlank interrupt to mmVBlank with the libgba interrupt handler.

For DS, this function also sets up the internal soundbank interface to use the file specified.

mmInitDefaultMem

-

Prototype

```
void mmInitDefaultMem( mm_addr soundbank );
```

Parameters

soundbank

Memory address of soundbank file.

Description

Initializes Maxmod with default settings. This function also sets up the internal soundbank interface to use the file that is located somewhere in memory.

---

mmInit

-

Prototype

```
void mmInit( mm_ds_system* system );
void mmInit( mm_gba_system* system );
```

Parameters

system

Maxmod setup configuration.

Description

Initializes Maxmod with the settings specified.

For DS projects, you must also setup a soundbank interface with one of the mmSoundBank* functions.

For GBA projects, irqInit() should be called before this function.

DS Example

```
void maxmodInit( void )
{
    mm_ds_system sys;
    // number of modules in your soundbank (defined in output header)
    sys.mod_count = MSL_NSONGS;

    // number of samples in your soundbank (defined in output header)
    sys.samp_count= MSL_NSAMPS;

    // memory bank, allocate BANKSIZE (or NSONGS+NSAMPS) words
    sys.mem_bank = malloc( MSL_BANKSIZE * 4 );

    // select fifo channel
    sys.fifo_channel = FIFO_MAXMOD;

    // initialize maxmod
    mmInit( &sys );

    mmSoundBankInMemory( (mm_addr)my_soundbank );

    // or
    //
    //mmSoundBankInFiles( "my_soundbank.msl" );
}
```

GBA Example

```
// Mixing buffer (globals should go in IWRAM)
// Mixing buffer SHOULD be in IWRAM, otherwise the CPU load
// will _drastially_ increase
u8 myMixingBuffer[ MM_MIXLEN_16KHZ ] __attribute((aligned(4)));

void maxmodInit( void )
{
    irqSet( IRQ_VBLANK, mmVBlank );

    u8* myData;
    mm_gba_system mySystem;

    // allocate data for channel buffers & wave buffer (malloc'd data goes to EWRAM)
    // Use the SIZEOF definitions to calculate how many bytes to reserve
    myData = (u8*)malloc( 8 * (MM_SIZEOF_MODCH
                               +MM_SIZEOF_ACTCH
                               +MM_SIZEOF_MIXCH)
                               +MM_MIXLEN_16KHZ );

    // setup system info
    // 16KHz software mixing rate, select from mm_mixmode
    mySystem.mixing_mode       = MM_MIX_16KHZ;

    // number of module/mixing channels
    // higher numbers offer better polyphony at the expense
    // of more memory and/or CPU usage.
    mySystem.mod_channel_count = 8;
    mySystem.mix_channel_count = 8;

    // Assign memory blocks to pointers
    mySystem.module_channels   = (mm_addr)(myData+0);
    mySystem.active_channels   = (mm_addr)(myData+(8*MM_SIZEOF_MODCH));
    mySystem.mixing_channels   = (mm_addr)(myData+(8*(MM_SIZEOF_MODCH
                                                      +MM_SIZEOF_ACTCH)));
    mySystem.mixing_memory     = (mm_addr)myMixingBuffer;
    mySystem.wave_memory       = (mm_addr)(myData+(8*(MM_SIZEOF_MODCH
                                                      +MM_SIZEOF_ACTCH
                                                      +MM_SIZEOF_MIXCH)));
    // Pass soundbank address
    mySystem.soundbank         = (mm_addr)soundbank;

    // Initialize Maxmod
    mmInit( &mySystem );
}
```

---

mmInstall

-

Prototype

```
void mmInstall( int fifo_channel );
```

Parameters

fifo_channel

FIFO communication channel to use. Must equal the channel the ARM9 side will be sending data to. (usually FIFO_MAXMOD)

Description

Installs the Maxmod system on the ARM7 side. Normally, this is the only function you need to call on ARM7.

---

mmSelectMode

Prototype

```
void mmSelectMode( mm_mode_enum mode );
```

Parameters

mode

New audio mode. Pass MM_MODE_A for complete hardware mixing, MM_MODE_B for interpolated mixing, or MM_MODE_C for extended mixing.

Description

Switches the audio mode for Maxmod DS.

Hardware mixing offers 16-channel audio with minimal CPU load.

Interpolated mixing extends the capability of the hardware channels by adding linear interpolation in software.

Extended mixing increases the channel count to 30 with software mixing.

---

mmLockChannels

Prototype

```
void mmLockChannels( mm_word bitmask );
```

Parameters

bitmask

Selection of channels to lock. Bit0 = Channel0, Bit1 = Channel1 ... Bit15 = Channel15

Description

This function locks audio channels to prevent Maxmod from using them. This is for when you need to operate on the DS hardware channels directly. Note that if you use this function while music or sound effects are playing, any active notes that are using the channels to be locked will be cut.

---

mmUnlockChannels

Prototype

```
void mmUnlockChannels( mm_word bitmask );
```

Parameters

bitmask

Selection of channels to unlock. Bit0 = Channel0, Bit1 = Channel1, Bit2 = Channel2 ... Bit15 = Channel15

Description

Unlocks audio channels to allow Maxmod to use them. This function can be used to restore channel usage to Maxmod when you are finished using certain channels.

Note that in the "Interpolated Audio" mode, channels **can not** be unlocked. To unlock channels in the interpolated mode you must reset the audio system. To reset the audio system, use mmSelectMode.

---

mmVBlank

-

Prototype

```
void mmVblank( void )
```

Description

This function **must** be linked directly to the VBlank IRQ. During this function, the sound DMA is reset. The timing is extremely critical, so make sure that it is not interrupted, otherwise garbage may be heard in the output.

If you need another function to execute after this process is finished, use mmVBlankReturn to install a your handler.

Example setup with libgba system

```
void setup_interrupts( void )
{
    irqInit();
    irqSet( IRQ_VBLANK, mmVBlank );
    irqEnable( IRQ_VBLANK );

    mmVBlankReturn( myVBlankHandler ); // this is optional
}
```

---

mmSetVBlankHandler

-

Prototype

```
void mmSetVBlankHandler( void* function );
```

Parameters

function

Pointer to your VBlank handler.

Description

Installs a custom handler to be processed after the sound DMA is reset. If you need to have a function linked to the VBlank interrupt, use this function (the actual VBlank interrupt must be linked directly to mmVBlank).

---

mmSetEventHandler

-

Prototype

```
void mmSetEventHandler( mm_callback handler );
```

Parameters

handler

Function pointer to event handler.

Description

Use this function to receive song events. Song events occur in two situations. One is by special pattern data in a module (which is triggered by SFx/EFx commands). The other occurs when a module finishes playback (in MM_PLAY_ONCE mode).

Note for GBA projects: During the song event, Maxmod is in the middle of module processing. Avoid using any Maxmod related functions during your song event handler since they may cause problems in this situation.

---

mmFrame

-

Prototype

```
void mmFrame( void );
```

Description

This is the main routine-function that processes music and updates the sound output.

For GBA, this function **must** be called every frame. If a call is missed, garbage will be heard in the output and module processing will be delayed.

For DS (ARM7), this function is called automatically.

---

mmSoundBankInFiles

-

Prototype

```
void mmSoundBankInFiles( char* filename );
```

Parameters

filename

Filename of your soundbank binary.

Description

This function sets up the standard interface for a soundbank that is loaded in the file system. This function should be called after mmInit and before any Load or Unload operations. This function is called by mmInitDefault.

---

mmSoundBankInMemory

-

Prototype

```
void mmSoundBankInMemory( mm_addr address );
```

Parameters

address

Memory address of soundbank file.

Description

This function enables the standard interface for a soundbank that is loaded into memory. This function should be called after mmInit and before any Load or Unload functions. This function is called by mmInitDefaultMem.

---

mmSetCustomSoundBankHandler

-

Prototype

```
void mmSetCustomSoundBankHandler( mm_callback p_loader );
```

Parameters

p_loader

Function pointer to soundbank request handler.

Description

This function installs a custom routine to interface with the soundbank data. The routine will be responsible for handling requests from Maxmod to access data in the soundbank.

Module Playback

mmLoad

Prototype

```
void mmLoad( mm_word module_ID );
```

Parameters

module_ID

Index of module to be loaded. Values are defined in the soundbank header output. (prefixed with "MOD_")

Description

Loads a module into memory for playback. May be played with mmStart afterwards.

---

mmUnload

Prototype

```
void mmUnload( mm_word module_ID );
```

Parameters

module_ID

Index of module to be unloaded from memory.

Description

Unloads and frees memory space occupied by a module.

Caution: Ensure that the module to be unloaded isn't playing!

---

mmStart

-

Prototype

```
void mmStart( mm_word module_ID, mm_pmode mode );
```

Parameters

module_ID

Index of module to be played. Values are defined in the soundbank header output. (prefixed with "MOD_")

mode

Mode of playback. Can be **MM_PLAY_LOOP** (play and loop until stopped manually) or **MM_PLAY_ONCE** (play until end).

Description

Begins playback of module. For DS, the module must be loaded into memory first (mmLoad). For GBA, the module data is read directly from the cartridge space, so no loading is needed.

---

mmPause

-

Prototype

```
void mmPause( void );
```

Description

Pauses playback of the active module. Resume with mmResume.

DS Note: The DS hardware channels do not allow actual pausing. To implement pausing, Maxmod sets the channel frequencies to the minimal value (256Hz). This may cause some problems with certain samples (such as drum-loops) drifting out of sync with the song temporarily. This is not an issue with the interpolated audio mode since the channels are fed by software.

---

mmResume

-

Prototype

```
void mmResume( void );
```

Description

Resumes playback of a paused module. Pause with mmPause.

---

mmStop

-

Prototype

```
void mmStop( void );
```

Description

Stops playback of the active module. Start again with mmStart.

Any channels used by the active module will be freed.

---

mmPosition

-

Prototype

```
void mmPosition( mm_word position );
```

Parameters

position

New sequence position.

Description

This function sets the current sequence [aka order-list] position for the active module.

---

mmActive

-

Prototype

```
mm_bool mmActive( void );
```

Return

Nonzero if a module is currently playing.

---

mmJingle

-

Prototype

```
void mmJingle( mm_word module_ID );
```

Parameters

module_ID

Index of module to be played. (Defined in soundbank header)

Description

Plays a jingle (same as a normal module). Jingles can be mixed with the normal module playback. For DS, the module must be loaded into memory first (mmLoad). For GBA, the module is read directly from the cartridge space. For jingles, the playback mode is fixed to **MM_PLAY_ONCE**. Note that jingles must be limited to 4 channels only.

---

mmSetModuleVolume

-

Prototype

```
void mmSetModuleVolume( mm_word volume );
```

Parameters

volume

New volume level. Ranges from 0 (silent) to 1024 (normal).

Description

Use this function to change the master volume scale for muscial playback.

mmSetJingleVolume

-

Prototype

```
void mmSetJingleVolume( mm_word volume );
```

Parameters

volume

New volume level. Ranges from 0 (silent) to 1024 (normal).

Description

Use this function to change the master volume scale for jingle playback.

mmSetModuleTempo

-

Prototype

```
void mmSetModuleTempo( mm_word tempo );
```

Parameters

tempo

New tempo value. Tempo value = (speed_percentage * 1024) / 100.

Description

This function changes the master tempo for module playback. Specifying 1024 will play the module at its normal speed. Minimum and maximum values are 50% (512) and 200% (2048). Note that increasing the tempo will also increase the module processing load.

mmSetModulePitch

Prototype

```
void mmSetModulePitch( mm_word pitch );
```

Parameters

pitch

New pitch scale. Value = 1024 * 2^(semitones/12)

Description

This funciton changes the master pitch scale for module playback. Specifying 1024 will play the module at its normal pitch. Minimum/Maximum range of the pitch change is +-1 octave.

Sound Effects

mmLoad

-

Prototype

```
void mmLoadEffect( mm_word sample_ID );
```

Parameters

sample_ID

Index of sample to be loaded. Values are defined in the soundbank header output (prefixed with "SFX_)

Description

Loads a sound effect into memory for playback. May be played with mmEffect/mmEffectEx afterwards.

---

mmUnloadEffect

-

Prototype

```
void mmUnloadEffect( mm_word sample_ID );
```

Parameters

sample_ID

Index of sample to be unloaded.

Description

Unloads a sound effect and frees the memory if the reference count becomes zero.

---

mmEffect

Prototype

```
mm_sfxhand mmEffect( mm_word sample_ID );
```

Parameters

sample_ID

Index of sample to be played. Values are defined in the soundbank header. (prefixed with "SFX_")

Return

Sound effect handle. This value can be used to modify parameters of the sound effect while it is playing.

Description

Plays a sound effect with default settings. Default settings are: Volume=Max, Panning=Center, Rate=Center (specified in sample).

The value returned from this function is a handle and can be used to modify the sound effect while it's actively playing.

For DS, make sure the sample is loaded (mmLoadEffect).

---

mmEffectEx

Prototype

```
mm_sfxhand mmEffectEx( mm_sound_effect* sound );
```

Parameters

sound

Structure containing information about the sound to be played.

Return

Sound effect handle that may be used to modify the sound later.

Description

Plays a sound effect with custom settings.

---

mmEffectVolume

-

Prototype

```
void mmEffectVolume( mm_sfxhand handle, mm_byte volume );
```

Parameters

handle

Sound effect handle received from mmEffect/mmEffectEx.

volume

New volume level. Ranges from 0 (silent) to 255 (normal).

Description

Changes the volume of a sound effect.

---

mmEffectPanning

-

Prototype

```
void mmEffectPanning( mm_sfxhand handle, mm_byte panning );
```

Parameters

handle

Sound effect handle received from mmEffect/mmEffectEx.

panning

New panning level. Ranges from 0 (far-left) to 255 (far-right).

Description

Changes the panning of a sound effect.

---

mmEffectRate

-

Prototype

```
void mmEffectRate( mm_sfxhand handle, mm_word rate );
```

Parameters

handle

Sound effect handle received from mmEffect/mmEffectEx

rate

New playback rate. The actual playback rate depends on this value and the base frequency of the sample. This parameter is a 6.10 fixed point value, passing 1024 will return the sound to its original pitch, 2048 will raise the pitch by one octave, and 512 will lower the pitch by an octave. To calculate a value from semitones: *Rate = 1024 * 2^(Semitones/12)*. (please don't try to do that with integer maths)

Description

Changes the playback rate for a sound effect.

mmEffectScaleRate

-

Prototype

```
void mmEffectScaleRate( mm_sfxhand handle, mm_word factor );
```

Parameters

handle

Sound effect handle received from mmEffect/mmEffectEx.

factor

6.10 fixed point factor.

Description

Scales the rate of the sound effect by a certain factor.

mmEffectCancel

-

Prototype

```
void mmEffectCancel( mm_sfxhand handle );
```

Parameters

handle

Sound effect handle received from mmEffect/mmEffectEx.

Description

Stops a sound effect. The handle will be invalidated.

---

mmEffectRelease

-

Prototype

```
void mmEffectRelease( mm_sfxhand handle );
```

Parameters

handle

Sound effect handle received from mmEffect/mmEffectEx.

Description

Marks a sound effect as unimportant. This enables the sound effect to be interrupted by music/other sound effects if the need arises. The handle will be invalidated.

---

mmEffectCancelAll

-

Prototype

```
void mmEffectCancelAll( void );
```

Description

This function stops all sound effects and reset the effect system.

Audio Streaming

mmStreamOpen

-

Prototype

```
void mmStreamOpen( mm_stream* stream );
```

Parameters

stream

Pointer to a structure containing information about how the stream will operate.

Description

Opens an audio stream. See the tutorials section for more information.

Caution: If you have some heavy interrupt-routines that are enabled while calling this function, you may corrupt the stream. The function cannot disable interrupts internally because it requires the communications to be active.

---

mmStreamUpdate

-

Prototype

```
void mmStreamUpdate( void );
```

Description

Check buffering state and fill stream with data. Requests will be made to your callback to fill parts of the wave buffer(s). This function only needs to be called manually if the stream isn't in *auto-fill* mode. This function shouldn't be used when the stream is automatically filled.

---

mmStreamClose

-

Prototype

```
void mmStreamClose( void );
```

Description

Close audio stream.

Reverb

mmReverbEnable

-

Prototype

```
void mmReverbEnable( void );
```

Description

Enables the reverb system. This function locks 2 channels (1 & 3) to reserve them for reverb output. This must be called before the configuration functions can be used.

If music is playing when this function is used, a note or two may be cut due to the locking.

---

mmReverbDisable

-

Prototype

```
void mmReverbDisable( void );
```

Description

Disables the reverb system and restores the channels.

Note: In the interpolated audio mode, the channels cannot be restored by this function and the mixer must be reset instead.

---

mmReverbConfigure

-

Prototype

```
void mmReverbConfigure( mm_reverb_cfg* config );
```

Parameters

config

Reverb configuration structure.

Description

Configures the reverb system. See the reverb tutorial for a full explanation of the configuration structure.

mmReverbStart

-

Prototype

```
void mmReverbStart( mm_reverbch channels );
```

Parameters

channels

Reverb channel selection. Can use MMRC_LEFT or MMRC_RIGHT or (MMRC_LEFT| MMRC_RIGHT).

Description

Starts reverb output in the selected channels.

mmReverbStop

-

Prototype

```
void mmReverbStop( mm_reverbch channels );
```

Parameters

channels

Reverb channel selection.

Description

Stops reverb output for the selected channels. (can select left output, right output, or both)

mmReverbBufferSize

Prototype

```
static inline mm_word mmReverbBufferSize( mm_word bit_depth, mm_word
sampling_rate, mm_word delay );
```

Parameters

bit_depth

May be 8 or 16 to represent 8-bit or 16-bit bit-depth.

sampling_rate

Sampling rate in Hertz.

delay

Reverb delay value in milliseconds.

Return

Amount of memory required, measured in *words* (32-bit units).

Description

This small function calculates the amount of memory needed for the reverb buffer.

Definitions/Types

Type Definitions

Name	Description
mm_byte	8-bit value.
mm_hword	16-bit value.
mm_word	32-bit value.
mm_bool	Boolean value.
mm_addr	Memory address.
mm_callback	Callback function.
mm_stream_func	Audio streaming function.
mm_sfxhand	Sound effect handle.

Structures

Name	Description
mm_sound_effect	Sound effect parameters.
mm_gba_system	GBA initialization parameters.
mm_ds_system	DS initialization parameters.
mm_stream	Software stream information.
mm_ds_sample	DS audio sample.

Enumerations

Name	Description
mm_pmode	Module playback mode.
mm_mode_enum	Audio mode selection.
mm_mixmode	Software mixing rate selection.
mm_stream_formats	Audio stream formats.
mm_stream_timer	Audio stream timer selection.
mm_reverbflags	Reverb configuration flags.
mm_reverbch	Reverb channel selection.
mm_reverb_cfg	Reverb configuration.

Song Events

Name	Description
MMCB_SONGMESSAGE	Song message.
MMCB_SONGFINISHED	Module has reached end marker.