dream

an ATMEL Company

rev 11, Dec 1996

SAM9407 PROGRAMMER'S REFERENCE

Overview

This document details all the programming information for SAM9407 Windows applications or drivers development.

It is shared in 4 chapters

- WINDOWS API
- FIRMWARE API
- FIRMWARE Structures
- MIDI Implementation

The SAM9407 and Its associated firmware provides the following devices:

MIDI DEVICE:

- Wave table synthesis:
  34 to 64 voices of polyphony
  16/8 bits samples , any sampling rate, linear interpolation, forward/reverse loop
  12/24 dB resonant filter
  reverb & chorus send
  ROM and/or RAM download sound bank (up to 8 simultaneous sound bank)
- MPU401 compliant
- Complete MIDI implementation described in «SAM9407 MIDI IMPLEMENTATION» chapter

STREAMING AUDIO BUFFER DEVICE:

- 8 play + 1 record simultaneous tracks
  format: mono/stereo, 8/16 bits, any sampling rate
- Play tracks:
  12 dB filter, real time sampling rate controller
  Mode: 2 outputs with reverb & chorus send / 4 outputs

STATIC AUDIO BUFFER DEVICE:

- 34 to 62 tracks
  format: mono (stereo requires 2 tracks), 8/16 bits, any sampling rate
  Mode: 2 outputs with reverb & chorus send / 4 outputs
- looped wave:
  16/8 bits samples , any sampling rate, linear interpolation, forward/reverse loop
  24 dB resonant filter, real time sampling rate controller
  256 K sample maximum size
- 1 shot wave:
  16/8 bits samples , any sampling rate, linear interpolation, forward loop
  12 dB resonant filter, real time sampling rate controller
  no sample size limitation

AUDIO IN DEVICE:

The audio In device provides stereo input sampled at 39.25 KHz.
The audio in signal can be send to the reverb or to the embedded audio in echo.

This Audio In can be used to connect a microphone (karaoke application) or an external audio source like a CD audio (94PC32 add on card) or a CODEC (multi-media combo board).
In the last example this input gives the ability to add effects (reverb, equalizer & surround) to the CODEC inputs (Line In, Microphone, Audio CD)..

REVERB & CHORUS DEVICE

Provides 8 reverb + 8 chorus stereo programs compatible with GS standard.

EQUALIZER DEVICE:

Stereo four band parametric equalizer
Default setting 450Hz , 900Hz, 4kHz, 9kHz with +/-12dB band level.

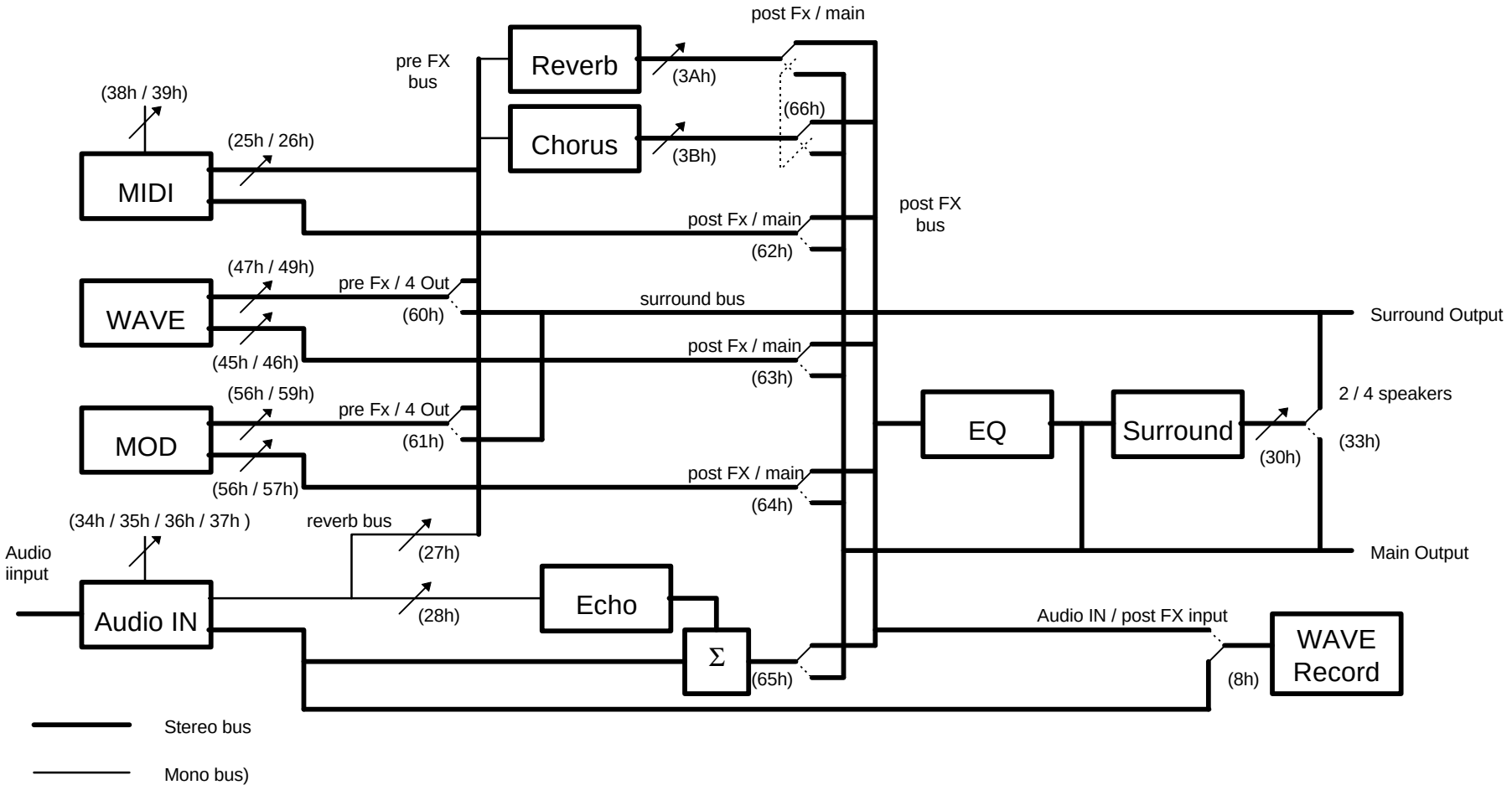
SURROUND DEVICE

Surround processing enable to expand stereo image of a stereo signal or to create a pseudo stereo image from a monophonic source.
Surround signal can be routed either to the main left/right output or to the Auxiliary left/right output.

PITCH SHIFTER DEVICE

This device may not be implemented in some versions of firmware.
The pitch shifter allows to change the pitch of selected sources (Audio in, Midi, Streaming Audio Buffer, Static Audio Buffer) without affecting the tempo.

Signal Processing Synoptic



Chapter 1

WINDOWS API

DREAM94.DRV driver provides 16bits WINDOWS 3.1 API

DREAM95.DRV driver provides 16bits & 32bits WINDOWS 95 API
Actually 32bits API are 16bits API converted VIA a thunk.

DREAM95.VXD driver provides 32bits WINDOWS 95 API

Windows Multimedia Low-Level Audio Services

Dream drivers provides entry points functions to support the Windows audio services:

- wodMessages
- widMessages
- modMessages
- midMessages
- auxMessages (not currently supported)

Those functions support enable application to use the Windows multimedia services:

Auxiliary Audio Device

auxiliary audio device services are not supported in the current version

Midi Input Device

- midiInGetNumDevs supported (1 device)
- midiInGetDevCaps supported
- midiInOpen supported
- midiInClose supported
- midiInPrepareHeader supported
- midiInUnPrepareHeader supported

- midiInAddBuffer supported
- midiInReset supported
- midiInStart supported
- midiInStop supported
- midiInGetErrorText supported

Midi Output Device

- midiOutGetNumDevs supported (1 device)
- midiOutGetDevCaps supported
- midiOutOpen supported
- midiOutClose supported
- midiOutPrepareHeader supported
- midiOutUnPrepareHeader supported
- midiOutShortMsg supported
- midiOutLongMsg supported
- midiOutGetVolume supported
- midiOutSetVolume supported
- midiOutGetErrorText supported

Wave Input Device

- waveInGetNumDevs supported (1 device)
- waveInGetDevCaps supported
- waveInOpen supported
- waveInClose supported
- waveInPrepareHeader supported
- waveInUnPrepareHeader supported
- waveInGetErrorText supported
- waveInAddBuffer supported
- waveInReset supported
- waveInStart supported

- waveInStop supported

Wave Output Device

- waveOutGetNumDevs supported (8 devices)
- waveOutGetDevCaps supported
- waveOutOpen supported
- waveOutClose supported
- waveOutPrepareHeader supported
- waveOutUnPrepareHeader supported
- waveOutGetErrorText supported
- waveOutWrite supported
- waveOutReset supported
- waveOutStart supported
- waveOutRestart supported
- waveOutGetVolume supported
- waveOutSetVolume supported
- waveOutGetPitch see below
- waveOutSetPitch see below
- waveOutGetPlaybackRate not supported
- waveOutSetPlaybackRate not supported

The Pitch API are supported but instead of doing Pitch transpose as required by Windows Mmedia specification, It actually change the sampling frequency.

Pitch transpose	modify playback spectrum	keep playback duration
Playback Rate variation	keep playback spectrum	modify playback duration
Sampling Freq variation	modify playback spectrum	modify playback duration

The Pitch & PlaybackRate API required specific processing that actually is supported by SAM9407 with the Pitch shifter Device (see FIRMWARE API chapter).
These API can be correctly implemented if necessary.

SAM9407 WINDOWS API LIST

mpuMessage	Send MPU command
mpuMessageAc	Send MPU command with acknowledge
genControlRead	Read parameter value from Control Table.
surroundSetVolume	Set 3D effect level
equalizerSetBand	Set equalizer band parameters
memRead	Read a memory block from Dream card
memWrite	Write a memory block into Dream card
memGetMapAddress	Get Address of the Dream card Memory Mapping Table
memSetMapAddress	Set Address of the Dream card Memory Mapping Table
waveSetBufferSize	Set the size of the wave buffers for further wave open
waveSetMainVolume	Set Main volumes
waveSetAuxVolume	Set AUX volume
waveSetFilter	Set filter parameters
waveSetPitch	Set sample rate
modOpen	Open the static audio device
modGetVoiceNumber	Get the number of available voices
modOpenVoice	Open a static wave voice
modChangeParam	Change a parameter (volume, pitch, filter) for a static wave voice
modDefineVoiceMemory	Set the sample parameters (start, loop, end) for a static wave voice
modGetPos	Get the current reading position.
modSetNewPos	Set a new reading position
modCommand	Send command (Start, Pause, Stop) to play and stop a voice
modClose	Close the static audio device

The Following API are 32bits VxD API (not available in current 1.40 version)

bankLoad	Load Sound bank from .94b file to SAM9407 memory
bankUnload	Unload Sound bank from .SAM9407 memory
bankGetCaps	Change the priority of a sound bank
bankSetPriority	Fill a structure with SAM9407 memory bank information

mpuMessage

Syntax WORD **mpuMessage**(wCommand, lpParam, wCount)
Send MPU command via the driver

Parameters WORD wCommand
Dream expanded MPU401 command number
LPWORD lpParam
Pointer to a byte table holding the command parameter
WORD wCount
Parameter byte count

Return Value Returns zero if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments This API enable windows programmer to send MPU401 command which are not supported by Microsoft or Dream API.
Parameter table is byte type, word parameter are stored LSB first.
See FIRMWARE API chapter for details on MPU401 command.

mpuMessageAc

Syntax WORD **mpuMessageAc**(wCommand, lpParam, wCount,Ack)
Send MPU command with acknowledge via the driver

Parameters WORD wCommand
Dream expanded MPU401 command number
LPWORD lpParam
Pointer to a byte table holding the command parameter
WORD wCount
Parameter byte count
BYTE Ack
Acknowledge that should be waited by the driver

Return Value Returns 0 if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments This API enable windows programmer to send MPU401 command which are not supported by Microsoft or Dream API.
Parameter table is byte type, word parameter are stored LSB first.
See FIRMWARE API chapter for details on MPU401 command.

genControlRead

Syntax WORD **genControlRead**(lpValue,bCommand)
Read parameter value from Control Table

Parameters LPBYTE lpValue
Pointer on a location filled by the returned control value
one BYTE if bCommand<7f
128 BYTE if bCommand=0xff
BYTE bCommand
0-7f: control nb for which the value is returned
0xff, Value is assumed to be a 128 BYTE table and is filled with 128 control values.
Value[n] is the value for the control nb n.

Return Value Returns zero if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments Control table is described in FIRMWARE Structure Chapter

surroundSetVolume

Syntax WORD **surroundSetVolume**(wVolume)
Set surround effect intensity.

Parameters WORD wVolume
Surround volume: 0 no surround, 0x7f maximum surround effect

Return Value Returns zero if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments Surround effect provides additional parameters: delay, input format, output format. Those parameters could be set using **mpuMessage** function.

equalizerSetBand

Syntax WORD **equalizerSetBand**(wBand,wChannel,wVolume)
Set equalizer Band Volume

Parameters WORD wBand
Band selected: EQ_BASS, EQ_MID1, EQ_MID2, EQ_TREBLE
WORD wChannel
Channels select: EQ_STEREO, EQ_LEFT, EQ_RIGHT
WORD wVolume
New Volume dB scale, 0 -12dB ,0x40 0dB, 0x7f +12dB

Return Value Returns zero if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments Equalizer is a serial effect. It process mixed audio signal, this includes Midi, Wave, Mod and Audio In.

Bass band is lowpass type, Medium bands are band pass type, High band is high pass type
Band filter frequency are 450Hz, 900Hz, 4khz, 9KHz.
Equalizer is parametric type, band frequency could be adjust using **mpuMessage** function.

memRead

Syntax WORD **memRead**(lpDst,dwSrcAddress,wCount)
Read a memory bloc from Dream card

Parameters LPVOID lpDst
Pointer to memory bloc to receive data read from Dream card.
DWORD dwSrcAddress
32bit Dream card memory Address
WORD wCount
Count of word to read

Return Value return zero if the function was successful. Otherwise, it returns an error number, possible returns are:

DREAMERR_ABORT	Dream card memory unit busy, read impossible, retry later.
DREAMERR_INVCOUNT	wCount > 32K word
DREAMERR_INVADD	block not in 64K page
DREAMERR_OUTOFMEM	block out of SAM9407 memory
DREAMERR_UNREF	any other error

Comments The memory block to read should fit in 64K word page and its size can't exceed 32K word.

memWrite

Syntax WORD **memWrite**(lpSrc,dwDestAddress,wCount)
Write a memory bloc into Dream card

Parameters LPVOID lpSrc
Pointer to memory bloc to write to Dream card.
DWORD dwDestAddress
32bit Dream card memory Address
WORD wCount
Number of word to write

Return Value return zero if the function was successful. Otherwise, it returns an error number, possible returns are:

DREAMERR_ABORT	Dream card memory unit busy, write impossible, retry later.
DREAMERR_INVCOUNT	wCount > 32K word
DREAMERR_INVADD	block not in 64K page
DREAMERR_OUTOFMEM	block out of SAM9407 mem
DREAMERR_UNREF	any other error

Comments The memory block to write should fit in 64K word page and its size can't exceed 32K word.

memGetMapAddress

Syntax DWORD **memGetMapAddress**()
Get Address of the Dream card Memory Mapping Table

Parameters none

Return Value return the 32bit Address of Memory Mapping Table. Otherwise, it returns DREAMERR_UNREF.

Comments The Memory Mapping Table is described in chapter OVERVIEW - SAM9407 Memory Management

memSetMapAddress

Syntax WORD **memSetMapAddress**(dwAddress)
Set Address of the Dream card Memory Mapping Table

Parameters DWORD dwAddress
32bit Address of Memory Mapping Table in Dream card memory.

Return Value return the 32bit Address of Memory Mapping Table. Otherwise, it returns DREAMERR_UNREF.

Comments The Memory Mapping Table is described in chapter OVERVIEW - SAM9407 Memory Management

waveSetBufferSize

Syntax WORD **waveSetBufferSize**(wSize,bForPlay)
Set the size of the wave buffers for further opens
if bForPlay=TRUE, the play buffers size is set, else the record buffer size is set.

Parameters WORD wSize
Size in word. Maximum size enabled by memWrite and memRead is 32K.
BOOL bForPlay
TRUE if modification is for play buffers, FALSE if it's for record buffer.

Return Value return zero if the function was successful. Otherwise, it returns DREAMERR_UNREF.

Comments The waveSetBufferSize doesn't affect current wave but only future open waves.

waveSetMainVolume

Syntax WORD **waveSetMainVolume**(Id, wVolLeft, wVolRight)
Set wave main output volume

Parameters WORD wVoice
Voice Number
WORD wVolLeft
Left Volume
WORD wVolRight
Right Volume

Return Value return zero if the function was successful. Otherwise, it returns DREAMERR_UNREF.

Comments This command is mainly used to control volume in multi-wave mode.

waveSetAuxVolume

Syntax WORD **waveSetAuxVolume**(Id, wVolLeft, wVolRight)
Set wave auxiliary output volume

Parameters WORD wVoice
Voice Number
WORD wVolLeft
Left Volume
WORD wVolRight
Right Volume

Return Value return zero if the function was successful. Otherwise, it returns DREAMERR_UNREF.

Comments Aux output can be affected to rear speakers or reverb/chorus effect send. see MPU401 EXPANDED PROTOCOL for more details.

waveSetFilter

Syntax WORD **waveSetFilter**(Id, wControl, wValue)
Set filter parameters

Parameters WORD wControl
Control type: WAVE_FC , WAVE_Q
WORD wValue
Value 0x0-0x7f: filter open (0x7f), filter close (0x0), no resonance (0x7f), maximum resonance (0x0)

Return Value return zero if the function was successful. Otherwise, it returns DREAMERR_UNREF.

Comments The filter used for wave are 12dB low pass filters. The WAVE_Q parameter is active for mono wave only.

waveSetPitch

Syntax WORD **waveSetPitch**(Id, wAbsolutePitch)
Set the sample rate

Parameters WORD wAbsolutePitch
Absolute value of the new sample rate

Return Value return zero if the function was successful. Otherwise, it returns DREAMERR_UNREF.

modOpen

Syntax WORD **modOpen**()
Open the static audio device

Parameters none

Return Value return zero if the function was successful. Otherwise, it returns MMSYSERR_ALLOCATED.

Comments This API must be called before any other static wave API (mod...).

modGetVoiceNumber

Syntax WORD **modGetVoiceNumber**()
Get the number of available voices.

Parameters none

Return Value return the number of available voices.

Comments This API should be called before openning a static wave voice (modOpenVoice) to check if the voice can be played.

modOpenVoice

Syntax WORD **modOpenVoice** (wVoice, bVol, wVolMain, wVolAux, wPitch, wFilt)
Open a static wave voice and set volumes, pitch, filter parameters.

Parameters WORD wVoice
Voice number. Should be smaller than the number of available voice returned by **modGetVoiceNumber**
If the bit 7 is set, the voice is assumed to be « cross bank ». This allow to play samples which arenot necessary inside 256kW bank. However, such a voice actually reserves two internal voices. (See §1, Static Audio Device).

BYTE bVol
Value 0x0-0xff: General volume for the static wave

WORD wVolMain
LSB-->Right main volume (0-ff)
MSB-->Left main volume (0-ff)

WORD wVolAux
LSB-->Right aux volume (0-ff)
MSB-->Left aux volume (0-ff)

WORD wPitch
400h=nominal frequency (depending on quartz frequency) and linearly scalled (200h = 1/2 nom. freq.)

WORD wFilt
MSB-> Fc 0(close)-ff(open)
LSB->Q 0(resonnance max.)-ff(no resonnance)

Return Value return zero if the function was successful.

modCommand

Syntax DWORD **modCommand**(wVoice, bCommand)
Send a command to static wave (start, stop, pause)

Parameters
WORD wVoice : voice number

BYTE bCommand	=	MOD_START
	or	MOD_STOP
	or	MOD_CLOSE

Return Value 0 if succed, error messages if failed.

modChangeParam

Syntax WORD **modChangeParam**(wVoice, wType, wValue)
Change a parameter (volume, pitch, filter) for a static wave voice.

Parameters WORD wVoice : voice number

According to wType the wValue is :

if wType=VOI_VOL	General volume for the static wave (0-ff)	
if wType=VOI_MAIN	LSB-->Right main volume (0-ff)	
	MSB-->Left main volume (0-ff)	
if wType=VOI_AUX	LSB-->Right aux volume (0-ff)	
	MSB-->Left aux volume (0-ff)	
if wType=PITCH	Pitch : 400h=nominal frequency and linearly scalled (200h = 1/2 nom. freq.)	
if wType=VOI_FILT	Filt	MSB-> Fc 0(close)-ff(open)
		LSB->Q 0(resonnance max.)-ff(no resonnance)

Return Value return zero if the function was successful.
Comments This API could be called when the voice is playing.

modDefineVoiceMemory

Syntax WORD **modDefineVoiceMemory**(wVoice, wFormat, dwStart, dwLoop, dwEnd)
Define start,loop, and end sample address and define the sample format

Parameters

WORD wVoice : voice number

WORD Format bit 7 : 0->16 bits ; 1->8 bits
bit 6 : used if bit 7=1.
0-> 8 bits samples are in LSB of word memory
1->Sample in MSB of word memory
bit 0-5 : loop type
0 : forward loop
1 : reverse loop
2 : reverse loop with sign inversion

Only loop type 0 is available if a crossbank sample has been defined in **modOpenVoice**. (See §1, Static Audio Device)

DWORD dwStart, dwLoop, dwEnd, : absolute address (32 bits)
The sample is first read from dwStart to dwEnd, then it's looping between dwLoop an dwEnd.
If the voice hasn't been openned for « cross bank » sample, the sample must be inside a 256 kWords bank.
To define a *one shot* sample, the dwLoop and dwEnd should be set on 8 zero samples added after the sample .

Return Value return zero.

modGetPos

Syntax DWORD **modGetPos**(wVoice)
Return the current reading position for the voice

Parameters

WORD wVoice : voice number

Return Value return absolute address of the current reading position.

Comments This API could be called to check if the static wave is looping.

modSetNewPos

Syntax DWORD **modSetNewPos**(wVoice, dwOffset)
Add an offset to current reading position

Parameters

WORD wVoice : voice number

DWORD dwOffset : 32 bits offset to add.

Return Value 0 if succed, error messages if failed.

modClose

Syntax void **modClose**()
Close the static audio device.

Parameters none

Comments This function must be called after having used static audio device.

(The following API are not available in current 1.40 version)

bankLoad

Syntax WORD **bankLoad**(lpszFileName, lpszBankName)
Load Sound bank from .94b file to SAM9407 memory

Parameters

LPCSTR lpszFileName
Points to a null-terminated string that specifies the name (without extension) of the .94b sound bank file.

LPCSTR lpszBankName
Points to a null-terminated string that specifies the name of the sound bank. A null string assumes that the BankName is the same as the FileName.

Return Value return bank number if the function was successful. Otherwise, it returns 0FFFFH.

Comments

bankUnload

Syntax WORD **bankUnload**(wBankNumber)
Unload Sound bank from .SAM9407 memory

Parameters

WORD wBankNumber
0-7 number of the bank to be deleted
0FFFFH Unload all banks

Return Value returns zero if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments

bankGetCaps

Syntax WORD **bankGetCaps**(lpStructure)

Fill a structure with SAM9407 memory bank information

Parameters LPVOID lpStructure
Points to the structure to be filled

Return Value returns zero if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments The structure has the following format:

WORD	total available memory for sound bank
WORD	free memory for sound bank

8 times the following record, one per sound bank:

8 * CHAR	sound bank name
WORD	priority level of the sound bank
WORD	Size of the Sound bank

bankSetPriority

Syntax WORD **bankSetPriority**(wBankNumber, wBankPriority)
Change the priority of a sound bank

Parameters WORD wBankNumber
0-7 number of the bank to be deleted
WORD wBankPriority
Bank priority level (0 to 0FFFEH)
0FFFFH priority assumes that the bank is disable but still in memory

Return Value returns zero if the function was successful. Otherwise, returns DREAMERR_UNREF.

Comments The bank set priority affects the midi mapping table (see FIRMWARE Structure chapter)

Chapter 2

FIRMWARE API

I / O Interface

The I/O Interface is composed of two byte registers, one word register and one IRQ:

I/O address	Write from PC (OUT)	Read to PC (IN)
MPU_base + 0	DATA8	DATA8
MPU_base +1	CONTROL	STATUS
MPU_base +2/3	DATA16	DATA16

The byte registers provides backwards compatibility with the standard MPU401 UART mode.
The control message is sent on CONTROL register with one or several data on DATA8 register (word data are send as two bytes). The read back values (if any) are available on DATA8 register.

The word register provides high rate data transfer in burst mode.
It is used to download / upload SAM9407 memory Data.
It should always be used with 16 bits I/Os instruction (OUT DX,AX IN AX,DX OUTSW INSW).
Using 8 bit I/Os at these addresses will give unpredictable results.

The IRQ (PC compatible rising edge) is backward compatible with MPU401 interrupt.
It is floated until the MPU401 interface is switched to UART mode, to minimize potential IRQ conflicts.

IO Status Register:

TE	RF	ID1	ID0	X	X	X	X

TE : Transmit empty.
If 0, data from SAM9407 to PC is pending and IRQ is high
Reading the data at MPU_BASE+0 will set TE to 1 and clear IRQ.

RF : Receiver full.
If 0 then SAM9407 is ready to accept CONTROL or DATA from the PC.

TE and RF are MPU401 compliant. Two additional bits ID1 ID0 are provided. They allow to identify the logical SAM9407 device read DATA8 as follows :

ID1	ID0	Device
0	0	MIDI
0	1	Streaming audio
1	0	MOD player
1	1	General

Stand alone & UART modes

Stand alone mode:

After power-up, hardware reset or MPU reset control, the board is in **stand-alone mode**:

Stand alone mode enables only 2 controls:

- 3FH to switch to **UART mode**
  3FH control is acknowledged by receiving 0FEH as DATA8 with ID(1,0)=00 (Midi device).
- BEH to sent any control
  BEH enable to send only one control, this means that each control sent in stand alone mode should start with BEH control.

UART mode:

UART mode accepts any control.
Control 0FFH (MPU reset) switch back to **stand alone mode**.

CONTROL MESSAGES OVERVIEW

Control messages general format

A device control message consists of one CONTROL byte followed by one or several DATA8 bytes (parameters). A parameter can be byte or multiple bytes. The least significant byte is always sent first (little endian). «word » means two bytes, «double word » or «Dword » means four bytes.
The number of DATA8 bytes is fixed for a given CONTROL. After receiving the correct number of DATA8 bytes, operation resumes to the MIDI device.

Control messages list

Ctrl #	CONTROL NAME	Action
1h	WRT_MEM	Initialize PC to SAM9407 transfer
2h	RD_MEM	Initialize SAM9407 to PC transfer
3h	GET_MMT	Get Memory Mapping Table address
4h	SET_MMT	Set Memory Mapping Table address
7h	MASTER_VOL	Master volume
8h	REC_MODE	Select record mode
0Bh	TRANS_ONOFF	-Enable/disable pitch shifter function(1)
0Ch	TRANS_GMCH	-Enable/disable pitch shifter on General Midi channels
0Dh	TRANS_VAL	- Pitch shifter value
0Eh	TRANS_REVSEND	- Pitch shifter reverb send
0Fh	TRANS_CHRSEND	- Pitch shifter chorus send
10H	EQ_LBL	Equalizer low band left
11H	EQ_MLBL	Equalizer med low band left
12H	EQ_MHBL	Equalizer med high band left
13H	EQ_HBL	Equalizer high band left
14H	EQ_LBR	Equalizer low band right
15H	EQ_MLBR	Equalizer med low band right
16H	EQ_MHBR	Equalizer med high band right
17H	EQ_HBR	Equalizer high band right
18H	EQF_LB	Equalizer low band frequency
19H	EQF_MLB	Equalizer med low band frequency
1AH	EQF_MHB	Equalizer med high band frequency
1BH	EQF_HB	Equalizer high band frequency
20H	AUD_SEL	Audio1/2 input select (1)
21H	AUD_GAINL	Audio Left input gain (1)
22H	AUD_GAINR	Audio Right input gain (1)
25H	GMREV_SEND	General Midi Reverb Send
26H	GMCHR_SEND	General Midi Chorus Send
27H	AUDREV_SEND	Audio Reverb Send
28H	ECH_LEV	Echo level applied on audio in (2)
29H	ECH_TIM	Echo time applied on audio in (2)
2AH	ECH_FEED	Echo feedback applied on audio in (2)
30H	SUR_VOL	Surround effect volume
31H	SUR_DEL	Surround effect delay
32H	SUR_INP	Input mono/stereo select for surround
33H	SUR_24	2 or 4 speakers output select for surround
34H	AUDL_VOL	Left Channel Audio in volume (2)

35H	AUDR_VOL	Right Channel Audio in volume (2)
36H	AUDL_PAN	Left Channel Audio in pan (2)
37H	AUDR_PAN	Right Channel Audio in pan (2)
38H	GM_VOL	General Midi volume
39H	GM_PAN	General Midi pan
3AH	REV_VOL	Reverb general volume
3BH	CHR_VOL	Chorus general volume
3DH	EN_MIDOUT	Enable midi out
3FH	UART_MOD	Switch to UART mode
40h	W_OPEN	Open channel with specified format and sampling rate.
41h	W_CLOSE	Close channel. (3)
42h	W_START	Request SAM9407 to init its streaming audio data Emitter / Receiver
43h	END_XFER	Sent to SAM9407 after end of XFER.
44h	W_PITCH	Change the pitch of the wave during playback. (1)
45h	W_VOLLEFT	Change the current left volume (1)
46h	W_VOLRIGHT	Change the current right volume (1)
47h	W_VOLAUXLEFT	Change the left auxiliary level (1)
48h	GEN_INT	Generate an interrupt (used to check if 9407 board installed)
49h	W_VOLAUXRIGHT	Change the right auxiliary level. (1)
4AH	W_FILT_FC	Change filter cutoff frequency
4Bh	W_FILT_Q	Change filter resonance (2)
51h	GET_VOI	Get number of voices available
52h	VOI_OPEN	Open one voice
53h	VOI_CLOSE	Close one voice
54h	VOI_START	Start play voice
55h	VOI_STOP	Stop play voice
56h	VOI_VOL	Voice output volumes
57h	VOI_MAIN	Main channel send volumes
58h	VOI_PITCH	Voice pitch
59h	VOI_AUX	Aux channel send volumes
5Ah	VOI_FILT	Voice filter
5Bh	VOI_MEM	Defines memory used by voice
5Ch	GET_POS	Return position of current reading
5Dh	ADD_POS	Add an offset to current reading
60h	WAVE_ASS	Wave audio output assignment (3)
61h	MOD_ASS	MOD player audio output assignment (3)
62h	GM_POST	Post effects applied on general midi (4)
63h	WAVE_POST	Post effects applied on wave (4)
64h	MOD_POST	Post effects applied on MOD player (4)
65h	AUDECH_POST	Post effects applied on Audio in and echo(4)
66h	EFF_POST	Post effects applied on Reverb-chorus (4)
68H	ECH_ONOFF	Echo On/Off (3)
69H	REV_TYPE	Reverb program select
6AH	CHR_TYPE	Chorus program select
6BH	EQU_TYPE	Equalizer type (3)
6CH	REV_ONOFF	Reverb On/Off (3)
6DH	CHR_ONOFF	Chorus On/Off (3)
6EH	SUR_ONOFF	Surround On/Off (3)
6FH	AUD_ONOFF	Audio On/Off (3)
70H	HOT_RES	Hot reset
72H	POLY_64	Enable 64 voice polyphony (3)
74H	CHR_DEL	Chorus delay
75H	CHR_FEED	Chorus feedback
76H	CHR_RATE	Chorus rate

77H	CHR_DEPTH	Chorus depth
78H	REV_TIME	Reverb time
79H	REV_FEED	Reverb feedback
BEH	EN_CONTROL	Enable dream control in stand alone mode
FFh	RESET	Reset UART mode

GENERAL DEVICE

System messages

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
07h	MASTER_VOL	-Data (byte 0-FFh,FFh)	Master volume	
48h	GEN_INT	-Data=0 (byte)	Generate an interrupt	Id=11 Data=088h
70H	HOT_RES	-Data=011H	Hot reset	Id=10 Data=0
BEH	EN_CONTROL	None	Enable dream control in stand alone mode	
FFh	RESET	None	Reset UART mode	
3FH	UART_MOD	None	Switch to UART mode	Id=00 Data= 0FEh

- MASTER_VOL :
Master volume.
Data range : 0-FFh. Default=0FFh.

- GEN_INT :
This command is used to detect SAM9407 board.
It sends back 088H data and generates interrupt

- HOT_RES :
Equivalent to hardware reset. Firmware should be reloaded..

- EN_CONTROL:
This control has been implemented to enable to send any control even in **Stand alone mode**.
It enable to send only one control, this means that each control sent in stand alone mode should start with EN_CONTROL control.

- RESET:
Switch SAM9407 in stand alone mode

- UART_MODE:
Switch SAM9407 in UART mode

Memory messages

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
1h	WRT_MEM	-Start (Dword) -Count (word)	Initialize PC to SAM9407 transfer	Id=11 Data = ACh / ABh
2h	RD_MEM	-Start (Dword) -Count (word)	Initialize SAM9407 to PC transfer	Id=11 Data = ACh / ABh
3h	GET_MMT	-Data=0(byte)	Get Memory Mapping Table address	Id=00 Data=Address (Dword)
4h	SET_MMT	-Address (Dword)	Set Memory Mapping Table address	

- GET_MMT, SET_MMT:
The Memory Mapping Table is a fixed length table stored in SAM9407 memory.
This table is described 7in «Firmware Stucture» chapter..

- WRT_MEM, RD_MEM :
Initializes the basic transfer in burst mode between PC memory and SAM9407 memory.
All transfers must be 16 bit data transfers

Start address is given as a double word (4 bytes) :
Ad1,Ad2 offset inside a 64K page,
Ad3,Ad4 64K page number.

Count is number of words to transfer on 2 bytes (Lsb first) :
Cnt1,Cnt2.
Count is limited to 32K words maximum (cnt2|cnt1=4000h max.)

Warning !! : A transfer cannot cross a 64K words page boundary :
(ad2|ad1)+(cnt2|cnt1) < 10000H

When start address and count parameters have been sent, the PC must wait for the acknowledgment byte ACh before starting transfer. Under normal traffic conditions, this acknowledgment byte is received within less than 100µs.

If transfer in burst mode is already used by streaming audio (wave play/record), SAM9407 returns byte ABh and abort rd_mem or wrt_mem command. In that case, command rd_mem and wrt_mem must be resent after having ended transfer requested by streaming audio.

ABh byte can also be returned if one transfer is done with wrong count word (case of PC software mistake). In that case, next transfers cannot be done. The only solution is to correct PC software.

The actual transfer is done through port DATA16 using REP OUTSW (if WRT_MEM) or REP INSW (if RD_MEM) assembler instructions.

Example :

```
; segments and SI (resp.DI) already initialized to point to user buffer
            mov   dx,MPU_base+2
            mov   cx,count
            rep   outsw             or    rep   insw
```

Config messages

All the configuration controls reinitialize the SAM9407 firmware.
Therefore control 3Fh should be sent if UART mode is required.
All streaming and static wave channels are closed. General midi reset occurs.

Following commands are reset to their default values :
-UART_MODE
-AUD_SEL, AUD_GAIN, -ECH_LEV, ECH_TIM, ECH_FEED
-SUR_VOL, SUR_DEL, SUR_INP, SUR_24
-GM_VOL, GM_PAN, REV_VOL, CHR_VOL
-REV_TYPE, CHR_TYPE
-EQ_xxx, EQF_xxx

Only following commands are not reset :
-WAVE_ASS, MOD_ASS
-all xxx_POST commands
- ECH_TYPE, EQU_TYPE

Memory tables and sound banks are not modified.

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
6FH	AUD_ONOFF	-Data (byte 0/7Fh,7Fh)	Audio On/Off	Id=11 Data=0
68H	ECH_ONOFF	-Data(byte 0/7Fh,7Fh)	Echo On/Off	Id=11 Data=0
6CH	REV_ONOFF	-Data (byte 0/7Fh,7Fh)	Reverb On/Off	Id=11 Data=0
6DH	CHR_ONOFF	-Data (byte 0/7Fh,7Fh)	Chorus On/Off	Id=11 Data=0
6BH	EQU_TYPE	-Data(byte 0-2,0)	Equalizer type	Id=11 Data=0
6EH	SUR_ONOFF	-Data (byte 0/7Fh,7Fh)	Surround On/Off	Id=11 Data=0
72H	POLY_64	-Data(byte 0/7Fh,0)	Enable 64 voice polyphony	ID=11 Data=0
60h	WAVE_ASS	-Data(byte 0/7Fh,0)	Wave audio output assignment	Id=11 Data=0
61h	MOD_ASS	-Data(byte 0/7Fh,0)	MOD player audio output assignment	Id=11 Data=0

- AUD_ONOFF :
0 Audio in off
7FH (Default) Audio in on, requires 1 extra slot

- ECH_ONOFF :
0 echo applied on Audio in off
07FH (Default) echo applied on Audio in on, requires 2 extra slots

- REV_ONOFF :
0 reverb off
7FH (Default) reverb on, requires 13 extra slots

- CHR_ONOFF :
0 chorus off
7FH (Default) chorus on, requires 3 extra slots

- EQU_TYPE : Select type of equalizer.
0(Default) 4 band equalizer Requires 8 extra slots.

1 : 2 band equalizer (low, high). Requires 4 extra slots.
2 : no equalizer.

- SUR_ONOFF :

0 : surround effect off
7FH (Default) surround effect on, requires 1 extra slot

POLY_64 : (Optional)

0 (Default) off
7FH on, 64 voices of polyphony

Available only if firmware is downloaded into SRAM instead of DRAM (because refresh is disabled when using this command).
Set polyphony to 64 voices for midi, wave and mod player modules
When POLY_64 is on, equalizer, surround, reverb, chorus, audio in/echo, record functions are off. All commands concerning these functions are not active. The only way to active them again is to send POLY_64 off.

When POLY_64 is on, only midi, wave play and mod player are available Midi is output only on Main speaker output. Wave and mod player are always configured in 4 speaker output (wave_ass=7Fh, mod_ass=7Fh).

- WAVE_ASS :

Data = 0 : Wave module is configured for 2 speakers output, reverb and chorus can be applied to each wave :
- Command 45h, W_VOLLEFT : main left volume
- Command 46h, W_VOLRIGHT : main right volume
- Command 47h, W_VOLAUXLEFT : reverb send volume
- Command 48h, W_VOLAUXRIGHT : chorus send volume

Data = 7Fh : Wave module is configured for 4 speakers output. Reverb and chorus are not applied on waves :
- Command 45h, W_VOLLEFT : main left volume
- Command 46h, W_VOLRIGHT : main right volume
- Command 47h, W_VOLAUXLEFT : auxiliary left volume
- Command 48h, W_VOLAUXRIGHT : auxiliary right volume

Default =0

- MOD_ASS :

Data=0 : MOD player module is configured for 2 speakers output, reverb and chorus are available :
- Command 57h, VOI_MAIN : main send levels
- Command 59h, VOI_AUX : effect send (left=reverb send, right=chorus send)

Data=7Fh : MOD player module is configured for 4 speakers output, reverb and chorus are disabled :
- Command 57h, VOI_MAIN : main send levels
- Command 59h, VOI_AUX : auxiliary send levels.

Default=0

Routing messages

Post effects are surround + equalizer

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
62h	GM_POST	-Data(byte 0/7Fh,7Fh)	Post effects applied on general midi	
63h	WAVE_POST	-Data(byte 0/7Fh,7Fh)	Post effects applied on wave	
64h	MOD_POST	-Data(byte 0/7Fh,7Fh)	Post effects applied on MOD player	
65h	AUDECH_POST	-Data(byte 0/7Fh,7Fh)	Post effects applied on Audio in and echo	
66h	EFF_POST	-Data(byte 0/7Fh,7Fh)	Post effects applied on Reverb-chorus	

- xxx_POST :
Post effects are surround and equalizer.
Post effects can be separately applied on each module. However general settings of post effects (EQ_xxx, EQF_xxx, EQU_TYPE, SUR_VOL, SUR_DEL, SUR_INP and SUR_24) are common for all modules.
Data=0 : post effects not applied on module.
Data=7Fh : post effects applied on module.
Default value = 07Fh for all modules.

MIDI DEVICE

The current implementation specifies 32 MIDI channels for internal Wave table synthesis Device and a single 16 channels MIDI device for external MIDI device.

MIDI device is enable only in UART mode.
MIDI device is MPU401 UART mode compliant

This mode involved a specific protocol for MIDI device:

MIDI messages don't required any CONTROL but only DATA.
Writing DATA8 will send MIDI data to MIDI OUT and the wavetable synthesis.
MIDI data received from MIDI IN can be read as DATA8 with ID0-1=00

The detailed MIDI implementation is detailed in chapter 3

GM_VOL, GM_PAN & EN_MIDOUT are not MPU401 compliant messages, unlike most of midi messages they required CONTROL.

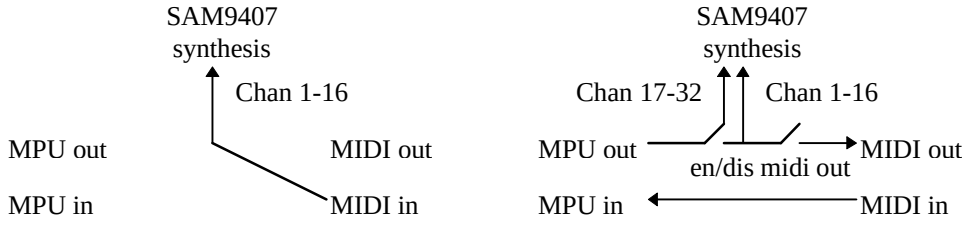
In **stand-alone mode,** the SAM9407 plays whatever is received on the MIDI IN serial line, MIDI OUT is disabled.

MIDI OUT (external cable) is enabled after 3FH (**UART mode**) and 3DH (enable midi out)

MIDI OUT is disabled after any other control than 3FH & 3DH . This means that any dream control should be followed by 3DH control to restore midi out. This has been done to avoid to send the data of control messages into MIDI OUT.

Only data from channels 1 to 16 are output on external MIDI OUT.
The channel selector is controled with B0H (channel 0-15) and B1H (channel 16-31).

Midi message path:



Stand alone mode **UART mode**

MIDI messages

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
38H	GM_VOL	-Data(byte 0-FFh,FFh)	General Midi volume	
39H	GM_PAN	-Data(byte 0-7Fh,40h)	General Midi pan	
3DH	EN_MIDOUT	None	Enable midi out	
B0H	MID_PORT0	None	Select chan 0 to 15	
B1H	MID_PORT1	None	Select chan 16 to 31	

- EN_MIDOUT :
Each general device control (except 3FH=UART_MOD) disables MIDI out. To enable again MIDI out, EN_MIDOUT must be sent before sending MIDI data to port MPU_base.

- GM_VOL
Range 0-FFh, linear scale.
Default value : GM_VOL=0FFh

- GM_PAN
0=hard left, 40h=center, 7Fh=hard right.
Pseudo logarithmic scale.
Same as GM system exclusive message «40h 00h 06h»
Default value :GM_PAN=040h

- MID_PORT0
Select midi port 0 (channel 0 to 15)
All the next midi messages are sent to midi port 0

- MID_PORT1
Select midi port 1 (channel 16 to 31)
All the next midi messages are sent to midi port 0

STREAMING AUDIO BUFFER DEVICE

I/O MESSAGES from PC -> SAM9407

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
40h	W_OPEN	-Channel (Byte) -Format (Byte) -Sampling rate (Word)	Open channel with specified format and sampling rate.	
41h	W_CLOSE	-Channel (Byte)	Close channel.	Id=01 (3) Byte=0Cx
42h	W_START	-Channel (Byte)	Request SAM9407 to initialize its streaming audio data Emitter / Receiver	
43h	END_XFER	-Channel (Byte)	Sent to SAM9407 after end of XFER.	
44h	W_PITCH	-Channel (Byte) -Sampling rate (Word)	Change the pitch of the wave during playback. (1)	
45h	W_VOLLEFT	-Channel (Byte) -Volume (Word)	Change the current left volume (1)	
46h	W_VOLRIGHT	-Channel (Byte) -Volume (Word)	Change the current right volume (1)	
47h	W_VOLAUXLEFT	-Channel (Byte) -Value (Word)	Change the left auxiliary level (1)	
49h	W_VOLAUXRIGHT	-Channel (Byte) -Value (Word)	Change the right auxiliary level. (1)	
4AH	W_FILT_FC	-Channel (Byte) -Value (Word)	Change filter cutoff frequency	
4Bh	W_FILT_Q	-Channel (Byte) -Value (Word)	Change filter resonance (2)	
8h	REC_MODE	-Data (byte 0/7Eh/7Fh,0)	Select record mode	

Notes
(1) : Play mode only
(2) : Mono mode.only
(3) :Play mode.only Byte returned is 0C0h + channel number (0-7). Acknowledge byte is sent when final half buffer read is completed.

I/O MESSAGES from SAM9407 -> PC

IRQ is used by SAM9407 to start streaming audio data block transfers between the PC and the SAM9407.
STATUS [5,4] = [ID1,ID0] with ID1=0 and ID0=1
DATA8[7..0]= Channel number (0 to 8)

The SAM9407 streaming audio buffer size is allocated by the PC using the memory mapping table (MMT) resources, (see memory table).

DETAILED PARAMETERS

For each command Channel is coded as follows:
From 0 to 7 for PLAY, 8 for RECORD

W_OPEN

Format	byte :	0000 00 ST 8B 8B : 0 if 16 bits, 1 if 8 bits. ST : 0 if mono, 1 if stereo.
Sampling Rate	word:	unsigned value from 0 to 48000 Hz.

W_VOLLEFT, W_VOLRIGHT, W_VOLAUXLEFT, W_VOLAUXRIGHT

Volume	word:	linear unsigned value from 0 to 65535

W_PITCH

Sampling Rate	word	unsigned value from 0 to 48000 Hz.

W_FILT_FC

FC	word	0 = filter closed, 0FFFFH filter open

W_FILT_Q

Q	word	0 = maximum resonance, 0FFFFH no resonance

- REC_MODE :

Data=0 : record only Audio In

Data=07Fh : record all signals going into Equalizer-Surround module (see Signal Processing synoptic in « overview » chapter)

Default = Data=0

Note : When in mode 07Fh, all commands xxx_POST have a double function : they are selecting if module is going into Equalizer-Surround module but also if module has to be record or not.

Data=07Eh : record all signals from surround bus (see Signal Processing synoptic)

STATIC AUDIO BUFFER DEVICE

The Static audio device can be used to provide hardware accelerator for Windows 95 Direct Sound.
It can also be used to implement specific MOD player Windows drivers.

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
51h	GET_VOI	-Data=0 (byte)	Get number of voices available	Id=10 Voice count (1 byte)
52h	VOI_OPEN	-Voice (byte)	Open one voice	No ack
53h	VOI_CLOSE	-Voice (byte)	Close one voice	
54h	VOI_START	-Voice (byte)	Start play voice	
55h	VOI_STOP	-Voice (byte)	Stop play voice	
56h	VOI_VOL	-Voice (byte) - Volume (byte)	Voice output volumes	
57h	VOI_MAIN	-Voice (byte) - Main right (byte) - Main left (byte)	Main channel send volumes	
58h	VOI_PITCH	- Voice (byte) - Pitch (word)	Voice pitch	
59h	VOI_AUX	- Voice (byte) - Aux right (byte) - Aux left (byte)	Aux channel send volumes	
5Ah	VOI_FILT	- Voice (byte) - Q (byte) - Fc (byte)	Voice filter	
5Bh	VOI_MEM	- Voice (byte) - Format (byte) - Bank (byte) - Start (3 bytes) - Bank loop (byte) - Loop (3 bytes) - Bank end (byte) - End (3 bytes)	Defines memory used by voice	
5Ch	GET_POS	- Voice (byte)	Return position of current reading	Id=10 bank (byte) offset in bank (3 bytes)
5Dh	ADD_POS	- Voice (byte) - Bank (byte) - Offset (3 bytes)	Add an offset to current reading	

DETAILS

- GET_VOI:
The returned byte Voice_count is the maximum number of voices that can be opened at the same time in the application. It is the total number of voices available for MOD player, Wave player and MIDI.
MOD player and Wave player have higher priority than MIDI. However MOD player has same priority as Wave player.
For example, suppose Voice_count = 32. If no voices are already opened for Wave player, MOD player can used 32 voices (but in that case no voices will be available for MIDI playing). If n voices are used for Wave player, MOD player has only 32-n voices available. Because Wave player and MOD player is usually included in the same driver it is the driver programmer responsibility to memorize how many voices are opened for Wave and for MOD.
MOD voices always start from 0 until maximum allowed (Voice_count-n-1), even if Wave player is active. Voices are dynamically allocated and do not correspond to physical slots of the SAM9407.
Note that crossbank type waves request two voices for each note played.

- VOI_OPEN :
Open one voice.

Voice byte :

D7	D6	D5	D4	D3	D2	D1	D0
C	0	0	v4	v3	v2	v1	v0

v4|v0 : voice to open
C : 0- « in bank » voice, 1- « cross bank » voice

Cross bank voice actually reserves two internal voices.
Note : if pitch shifter device is implemented in firmware, this cross bank mode is not available.

- VOI_CLOSE
VOI_CLOSE is immediately closing the voice.
Stops the voice softly in less than 5ms.
VOI_OPEN must be sent to use the voice again.

- VOI_STOP
VOI_STOP stops the voice softly in less than 5 ms.
This voice can be used again immediately without clicking for another wave (VOI_START without sending again VOI_OPEN).

- VOI_START
Sending VOI_START supposes that VOI_OPEN control has been sent before but also that VOI_MEM, VOI_VOL, VOI_PAN ... has been sent to initialize voice parameters.
VOI_CLOSE and VOI_STOP reset all parameters to undefined values.
Only VOI_FILT is automatically initialized to : Q=0FFh, Fc=0FFh (filter open)

- VOI_VOL :
Volume for main audio output.
Slide the volume to new value in less than 5ms.
Linear volume. Range 0-FFh

- VOI_MAIN :
Send for main audio output. Linear scale.
Main right : range 0-FFH, main right send level.
Main left : range 0-FFH, main left send level.

- VOI_PITCH :
Pitch given in 2 bytes, low byte first.
400h (low byte=0, high byte=4) correspond to nominal frequency (Current version of program runs with a sampling rate of 37.5 KHz, due to current SIMM DRAM cycle times).
Pitch is linearly scaled (200h = half nominal frequency).

- VOI_AUX :
Send level for auxiliary audio output or effect.
Depending on command MOD_ASS (61h), auxiliary output can be also effect send (see 5. General device command)
Linear scale.
Aux right : range 0-FFh, auxiliary right send level or chorus send.
Aux left : range 0-FFh, auxiliary left send level or effect send.

- VOI_FILT :
First byte : Q (resonance) : range 0-0ffh (0ffh=no resonance, 0=maximum of resonance)
2nd byte : Fc (cut frequency) : range 0-0ffh (0ffh=filter open, 0=filter closed)

- VOI_MEM :
Defines addresses of wave used by voice.
SAM9407 access to external memory is made through banks of 256K samples. Waves can be « in bank » or « cross bank « . Cross bank waves take two polyphony partials to be played. Therefore it is advisable to use « in bank » waves for optimum polyphony performance.

Waves can be stored in dram into 16 bit or 8 bit format. Both formats can be used at same time (for example, one 16 bit wave can be stored from address M till N-1 and two 8 bit waves can be stored from address N till P-1, etc...)
- Format byte :

D7	D6	D5	D4	D3	D2	D1	D0
f7	f6	f5	f4	f3	f2	f1	f0

f7 0 :16 bit sample, 1 : 8 bit sample
f6 used only if f7=1 (8 bit sample)
0 : sample in low byte of word, 1: sample in high byte of word

f5-f0 loop type 0 : forward loop (Play to loop end and jump to loop start)
1 : reverse loop (Play to loop end, back to loop start, forward to loop end ...)
2 : reverse loop with sign inversion (Play to loop end, invert sign and back to loop start, invert sign and forward to loop end ...)
Only loop type 0 is available if a crossbank sample has been defined for VOI_OPEN.

- Bank(byte)|Start(3bytes) : defines the sample start point
- Bank loop(byte)|Loop(3bytes) : defines the loop point of the sample
- Bank end(byte)|End(3 bytes) : defines the end point of the sample

3 byte offset inside a bank is coded LSB first :

Byte #	D7	D6	D5	D4	D3	D2	D1	D0
1	a7	a6	a5	a4	a3	a2	a1	a0
2	a15	a14	a13	a12	a11	a10	a9	a8
3	0	0	0	0	0	0	a17	a16

« In bank » samples have Bank = Bank loop = Bank end

Looping feature is necessary for all waves. « One-shot » waves must be done by adding n zero samples to the wave and setting loop point to end-n (n=8 recommended so that to allow drastic pitch change controls).
For cross bank samples loop and/or end points in the last eight locations of a bank should be avoided.

See also appendix A « MEMORY MAPPING TABLE » to know which part of memory can be used for MOD player.

- GET_POS
Return current reading position of the voice.
Useful to know if wave has reached final loop or not.
Return position value on 4 bytes :
byt1 = bank
bytes 2 to 4 = offset inside bank
byt2 = bit7-0

byt3 = bit15-8
byt4 = bit17-16

- ADD_POS
Add to current reading position an offset.
Given on 4 bytes :
byt1 = bank offset
bytes 2 to 4 = offset
byt2 = bit7-0
byt3 = bit15-8
byt4 = bit17-16
If current position + offset is going behind end point, position is set to end point.

- Writing and reading dram
Writing and reading into dram is done using general device controls 1 (WRT_MEM) and 2 (RD_MEM).
These controls are using a start address defined on 4 bytes ad1, ad2, ad3, ad4 :

```
ad1 = | a7  a6  a5  a4  a3  a2  a1  a0  |
ad2 = | a15 a14 a13 a12 a11 a10 a9  a8  |
ad3 = | a23 a22 a21 a20 a19 a18 a17 a16 |
ad4 = | a31 a30 a29 a28 a27 a26 a25 a24 |

ad4|ad3 = 64K page number.
ad2|ad1= offset inside a 64K page
```

The correlation with 256K bank structure used for MOD driver is :
bank number = bits a25-a18
offset inside bank = bits a17-a0
bit a31-a26 = 0.

AUDIO IN & ECHO DEVICE

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
20H	AUD_SEL	-Data(byte 0/7Fh,7Fh)	Audio1/2 input select (1)	
21H	AUD_GAINL	-Data(byte 0-7Fh,0)	Audio Left input gain (1)	
22H	AUD_GAINR	-Data(byte 0-7Fh,0)	Audio Right input gain (1)	
27H	AUDREV_SEND	-Data(byte 0-7Fh,0)	Audio Reverb Send	
34H	AUDL_VOL	-Data(byte 0-FFh,FFh)	Left Channel Audio in volume (2)	
35H	AUDR_VOL	-Data(byte 0-FFh,FFh)	Right Channel Audio in volume (2)	
36H	AUDL_PAN	-Data(byte 0-7Fh,0)	Left Channel Audio in pan (2)	
37H	AUDR_PAN	-Data(byte 0-7Fh,7Fh)	Right Channel Audio in pan (2)	
28H	ECH_LEV	-Data(byte 0-7Fh,0)	Echo level applied on audio in (2)	
29H	ECH_TIM	-Data(byte 0-7Fh,2Bh)	Echo time applied on audio in (2)	
2AH	ECH_FEED	-Data(byte 0-7Fh,40h)	Echo feedback applied on audio in (2)	

(1) Only if CS4216/CS4218 type Codec used (2 stereo audio inputs Codec)
(2) Assumes ADC used (at least 1 stereo audio input Codec)

- AUD_xxx :
These 3 commands can be used only if a Codec with 2 audio stereo inputs and adjustable gain on audio input is used (typically CS4216 or CS4218)
AUD_SEL : 0= select audio stereo input 1, 07Fh=select audio stereo input 2 (default 7Fh)
AUD_GAINL : 0=+0dB, 07Fh=+22.5dB on audio left input (default 0)
AUD_GAINR : 0=+0dB, 07Fh=+22.5dB on audio right input (default 0)
On 94PC32 board, audio input 2 is J4 connector and audio input 1 is J6 connector

- ECH_xxx :
Controls for echo applied on audio input.
Available only if echo set on with command ECH_ONOFF
ECH_LEV: 0 to 07Fh (Default 0)
ECH_TIM (if ech_type=07Fh only) : 0 =shortest to 7Fh=longest (default 2Bh)
ECH_FEED (if ech_type=07Fh only) : 0=no feedback, 7Fh=maximum feedback (default 40h)

- AUDREV_LEV :
Reverb send level for audio in
Data=0 to 07Fh (Default=0)
Reverb used is same as reverb used for General Midi

REVERB DEVICE

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
69H	REV_TYPE	-Data(byte 0-7,4)	Reverb program select	
3AH	REV_VOL	-Data(byte 0-FFh)	Reverb general volume	
78H	REV_TIME	-Data(byte 0-7Fh)	Reverb time	
79H	REV_FEED	-Data(byte 0-7Fh)	Reverb feedback	
25H	GMREV_SEND	-Data(byte 0-FFh,80h)	General Midi Reverb Send	

- REV_TYPE : Reverb program.
Same as GM system exclusive message « 40h 01h 30h » or GM control 80.

room1	room2	room3	hall1	hall2	plate	delay	pan delay
0H	1H	2H	3H	4H	5H	6H	7H

Default=4 (hall2)

REV_VOL: Reverb volume
Same as GM system exclusive message « 40h 01h 33h »
Default values:

room1	room2	room3	hall1	hall2	plate	delay	pan delay
90H	90H	90H	C0H	90H	90H	FFH	FFH

- REV_TIME : Reverb time.
Same as GM system exclusive message « 40h 01h 34h »
Default values:

room1	room2	room3	hall1	hall2	plate	delay	pan delay
7FH	7FH	7FH	7FH	7FH	7FH	18H	7FH

- REV_FEED : Reverb delay feedback.
Only if reverb number=6 or 7 (delays)
This command is same as GM system exclusive message « 40h 01h 35h »
Default values:

delay	pan delay
22H	26H

-GMREV_SEND: Modify reverb send level for General Midi.
80H: original reverb send levels of midi sequence not modified
0 to 7FH : original reverb send levels decreased
81h to FFH : original reverb send levels increased
Default=80h

CHORUS DEVICE

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
6AH	CHR_TYPE	-Data(byte 0-7,2)	Chorus program select	
3BH	CHR_VOL	-Data(byte 0-FFh)	Chorus general volume	
74H	CHR_DEL	-Data(byte 0-7Fh)	Chorus delay	
75H	CHR_FEED	-Data(byte 0-7Fh)	Chorus feedback	
76H	CHR_RATE	-Data(byte 0-7Fh)	Chorus rate	
77H	CHR_DEPTH	-Data(byte 0-7Fh)	Chorus depth	
26H	GMCHR_SEND	-Data(byte 0-FFh,80h)	General Midi Chorus Send	

- CHR_TYPE : Chorus program.
Same as GM system exclusive message « 40h 01h 38h » or GM control 81.

chorus1	chorus2	chorus3	chorus4	FB chorus	flanger	short del	FB delay
00H	01H	02H	03H	04H	05H	06H	07H

Default= 2 (chorus3)

- CHR_VOL : Chorus Volume
Same as GM system exclusive message « 40h 01h 3Ah »

- CHR_DEL : Chorus delay
Same as GM system exclusive message « 40h 01h 3Ch »

- CHR_VOL : Chorus Volume
Same as GM system exclusive message « 40h 01h 3Ah »

- CHR_FEED : Chorus feedback
Same as GM system exclusive message « 40h 01h 3Bh »

- CHR_RATE : Chorus rate
Same as GM system exclusive message « 40h 01h 3Dh »

- CHR_DEPTH : Chorus depth
Same as GM system exclusive message « 40h 01h 3Eh »

- CHR_VOL : Chorus Volume
Same as GM system exclusive message « 40h 01h 3Ah »

GMCHR_SEND : Modify chorus send level for General Midi.
Data=080h : original chorus send levels of midi sequence not modified
Data=0 to 07Fh : original chorus send levels decreased
Data=081h to 0ffh : original chorus send levels increased
Default=80h

Default values:

	chorus1	chorus2	chorus3	chorus4	FB chorus	flanger	short del	FB delay
CHR_VOL	90H	90H	90H	90H	90H	90H	FFH	FFH
CHR_DEL	4BH	40H	40H	2BH	7FH	56H	7FH	7FH
CHR_FEED	00H	07H	09H	0CH	48H	7FH	00H	50H
CHR_RATE	03H	09H	03H	09H	02H	01H	00H	00H
CHR_DEPTH	05H	13H	13H	10H	0CH	03H	00H	00H

EQUALIZER DEVICE

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
10H	EQ_LBL	-Level (byte 0-7Fh,60h)	Equalizer low band left	
11H	EQ_MLBL	-Level (byte 0-7Fh,40h)	Equalizer med low band left	
12H	EQ_MHBL	-Level (byte 0-7Fh,40h)	Equalizer med high band left	
13H	EQ_HBL	-Level (byte 0-7Fh,60h)	Equalizer high band left	
14H	EQ_LBR	-Level (byte 0-7Fh,60h)	Equalizer low band right	
15H	EQ_MLBR	-Level (byte 0-7Fh,40h)	Equalizer med low band right	
16H	EQ_MHBR	-Level (byte 0-7Fh,40h)	Equalizer med high band right	
17H	EQ_HBR	-Level (byte 0-7Fh,60h)	Equalizer high band right	
18H	EQF_LB	-Data (byte 0-7Fh,0Ch)	Equalizer low band frequency	
19H	EQF_MLB	-Data (byte 0-7Fh,1Bh)	Equalizer med low band frequency	
1AH	EQF_MHB	-Data (byte 0-7Fh,72h)	Equalizer med high band frequency	
1BH	EQF_HB	-Data (byte 0-7Fh,40h)	Equalizer high band frequency	

EQ_xxx Band level

00H	20H	40H	60H	7FH
-12dB	-6dB	0dB	+6dB	+12dB

Default =060h (+6dB) for LB-HB, =040h(0dB) for MLB-MHB

, EQF_xxx : Band frequency (0-7Fh), linear scale

Band	Range	Default
LB	0-4.7Khz	0CH
MLB	0-4.2Khz	1BH
MHB	0-4.2Khz	72H
HB	0-18.75Khz	40H

SURROUND DEVICE

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
30H	SUR_VOL	-Data(byte 0-FFh,0)	Surround effect volume	
31H	SUR_DEL	-Data(byte 0-7Fh,2)	Surround effect delay	
32H	SUR_INP	-Data(byte 0/7Fh,0)	Input mono/stereo select for surround	
33H	SUR_24	-Data(byte 0/7Fh,0)	2 or 4 speakers output select for surround	

- SUR_VOL : Surround effect volume.
Default=0

- SUR_DEL : Delay line length
Default=2

- SUR_INP : Input type select

0	Stereo (default),	Stereo wide,	Input to delay line is left - right.
7FH	Mono,	Pseudo stereo	Input to delay line is left + right.

- SUR_24 : Output type select

0	2 speakers(default)	Surround output on main outputs.
7FH	4 speakers	Surround output on auxiliary output.

PITCH SHIFTER DEVICE

This device is optional. May require some extended SIMM module.

Ctrl #	CONTROL NAME	Parameters (Data)	Action	Answer
0Bh	TRANS_ONOFF	-Data (byte) (0-0Fh)	-Enable/disable pitch shifter function(1)	Id=11 Data = 0
0Ch	TRANS_GMCH	-Data (word)	-Enable/disable pitch shifter on General Midi channels	
0Dh	TRANS_VAL	-Data (word)	- Pitch shifter value	
0Eh	TRANS_REVSEND	-Data (byte) (0-FFh)	- Pitch shifter reverb send	
0Fh	TRANS_CHRSEND	-Data (byte) (0-FFh)	- Pitch shifter chorus send	

note : (1) This control reinitializes the sam9407 firmware

8-2 - DETAILS

-TRANS_ONOFF :
Enable/disable pitch shifter function. When enable, requires 6 extra slots. This function can be independently selected on 4 different modules (Audio IN/Echo, Mod Player, Streaming Audio, Midi).
Some limitations of pitch shifting mode :
- Value of pitch shifting is the same for all modules.
- When setting a module in pitch shifting mode, effect (or auxiliary) send function of corresponding module is disabled. User can only access to a global reverb and chorus send value which will be the same for all modules being in pitch shifting mode (see command trans_revsend, trans_chrsend).
- Setting a module in pitch shifting mode will automatically set xxx_POST command ON for this module. Sending xxx_POST command OFF will cancel pitch shifting mode.
- If Streaming audio module or Mod Player module is set in pitch shifting mode, all channels inside module are pitch shifted.
Only for Midi module, user can select which channels are or are not pitch shifted (by using command TRANS_GMCHAN).
- If Streaming audio module or Mod Player module is set in pitch shifting mode, command xxx_ASS (60h, 61h) is disabled (always 0).

Data (1 byte), bit D7-D0 with :
* D7-D4 : Don’ t care
* D3 : Audio In/Echo pitch shifter On/Off. D3=1 ON , D3=0 OFF
If D3=1 :
    command AUDECH_POST (65h)= 7Fh
    command AUDREV_SEND (27h)= 0
* D2 : Mod Player pitch shifter On/Off. D2=1 ON , D2=0 OFF
If D2=1 :
    command MOD_POST (64h)= 7Fh
    command VOI_AUX (59h)= 0
    command MOD_ASS (61h)=0

* D1 : Streaming Audio pitch shifter On/Off. D1=1 ON , D1=0 OFF
If D1=1 :
command WAVE_POST (63h)= 7Fh
command W_VOLAUX (47h,49h)= 0
command WAVE_ASS (60h)=0
* D0 : General Midi pitch shifter On/Off. D1=0 ON , D0=0 OFF
If D0=1 :
command GM_POST (62h)= 7Fh
Midi controls 91 (reverb send), 93 (chorus send)=0 for midi channels assigned to pitch shifting mode only. For other channels, controls 91,93 are working normally.
Note :
For general midi, pitch shifter is working only if corresponding instrument is made with a single slot algorithm. All pcm algorithms are single slot, only FM4Y4 algorithm (fm synthesis) is double slot algorithm.

- TRANS_GMCH :
Select which channels are pitch shifted.
Data on 1 word : D15-D0.
Di=1, GM channel i pitch shifted
Di=0, GM channel i not pitch shifted
Note :
Di=1 will be active only if command TRANS_ONOFF has been sent before with bit D0=1.
Sending TRANS_ONOFF with bit D0=0, reset TRANS_GMCH value.
Note :
When a midi channel is set in pitch shifting mode, « Note Sounding Priority » is set to maximum for this channel. Note being played will never be killed by other non pitch shifted channels.

- TRANS_VAL :
Value of pitch shifting.
Data on 1 word.
Default = 04000h (no shifting).
Linearly scaled. If F0 is original frequency, new frequency F is given by formula :
F=F0 * (TRANS_VAL) / 4000h.
Recommended range 3000h-5000h (higher values are distorting signal).
Note :
Pitch shifter can be used in time stretching application.
For general midi, this can be done in simultaneously sending a pitch bend information with the pitch shifting value. Dream can provide an utility program (pitshift.exe) giving value of pitch bend and pitch shifting to apply when changing tempo of midi sequence.
All GM instrument requiring time stretching must be defined on 94WINST editor with frequency in normal mode and not in fix mode (because pitch bend is not applied if frequency is in fix frequency mode).

- TRANS_REVSEND, TRANS_CHRSEND :
Reverb and chorus send levels applied on all pitch shifted signals.
Data range : 0 to 0ffh
Default=0

Chapter 3

FIRMWARE Structures

The SAM9407 data Structures in its memory. Those structures includes data shared by firmware and client applications.

The SAM9407 memory stores 3 structures:

- Memory Mapping Table: defines the memory mapping (block type, start address, length) & sound bank list (name, priority level)
- Midi Mapping Table: defines the midi program & variation + sound bank source for each instrument
- Control table: store the current values of all the device parameters that can be read back by client applications

---

Memory Mapping Table (MMT)

Memory mapping table stays resident in 9407 board memory.
This table is split in two parts, memory block definition and MIDI sound bank definition.

Memory mapping table:

Nb	Type	Content
64	3*word	memory block definition
8	5*word	MIDI sound bank definition

Memory Block definition:

Nb	Type	Content
1	word	Block type
1	long	32 bits memory block start address

Block Type:

data [15..8]	data [7]	data [6..0]	content
0000 0000	0	000 0000	memory size
0000 0000	0	000 0001	free block
0000 0000	r	000 0010	system reserved block
0000 0000	r	000 0011	Memory Mapping table
0000 0000	r	000 0100	MIDI Mapping Table
0000 0000	r	000 0101	MIDI Mapping Table extension
0000 0000	r	000 0110	CONTROL Table
0000 i i i i	r	001 0000	MIDI sound parameter
dddd i i i i	r	001 0001	MIDI sound PCM
xxxx xxxx	0	010 0000	STREAMING WAVE buffer
xxxx xxxx	0	011 0000	STATIC WAVE buffer
1111 1111	1	111 1111	end of memory

Memory size block should always be the first block of the Memory Block definition.
All others memory block definition are sorted by ascending order; the start address is given by the block itself the end address is given by the next block.
End of memory block should always be the last block.
Memory Mapping Table block, MIDI Mapping Table block and MIDI Mapping Table Extension block should always be consecutive.

Data[7]:
This bit is the RAM(0) / ROM(1) flag. It defined the block memory type. ROM blocks could not be modified by user.

Memory Size 0000H
Total memory size including system reserved block.
This value is set by firmware and should not be modified by user.

Free Block 0001H
Free block is available for user. It could be filled or split by user.
Free block is always RAM type

System Reserved Block 0082H / 0002H
System reserved block can be program or data firmware block.
It can also be used for an address area without effective memory.
This block even RAM type can’t be modified by user.

Memory Mapping Table 0083H / 0003H
This block is the current Memory Mapping Table location.

The MMT start address can be modified by the user.
User should also modified MIDI mapping table start address (to be consecutive).
Thus MMT and MIDI mapping table should be relocate on memory by the user.
The user should also inform the firmware of the new MMT location using the SET_MMT MPU command (see FIRMWARE API)
MMT start address is initialize by the firmware.

MIDI Mapping Table 0084H / 0004H
The MIDI Mapping Table describes the program change and variation mapping according to the MIDI specification
This block can be initialized by the firmware if the card embedded a ROM MIDI sound bank.
To modify a MIDI Mapping Table in a ROM block, the user should relocate the MIDI mapping table in a free block and release the old block as a system reserved block.
To relocate the MIDI mapping table see MMT block relocation.

MIDI Mapping Table Extension 0085H / 0005H
This block provides an extension for MIDI Mapping table. It avoid to relocated the MMT and the MIDI Mapping table when inserting a new instrument in the MIDI mapping table.
The MIDI Mapping table Extension block is created and update by the user (512 word is recommended for initialization size)

Control Table 0006H
This block is Ram only. It is initialize and update by firmware and read only by user.
It can't be modified or relocate by user.

MIDI Sound Parameter 0000 iiix - 90H / 10H
This is the sound bank Parameter block.
« iii » is the sound bank ID (0 to 7), « iii » points to the sound bank definition
« x » is parameter block NB, each sound bank parameter can be mapped at best in two 64K word blocks.

MIDI Sound PCM dddd iii0 - 91H / 11H
This is the sound bank PCM block.
« iii » is the sound bank ID (0 to 7), « iii » points to the sound bank definition
« dddd » is PCM block NB, each sound bank PCM can be split in blocks.

Streaming Wave Buffer xx20H
This block is RAM only.
xx is the streaming wave NB 0-7 for play, 8 for record.

Static Wave Buffer xx30H
This block is RAM only.
xx is the static wave NB (play only).

Sound bank definition:

Nb	Type	Content
8	Char	MIDI sound bank name
1	word	MIDI sound bank priority level

Sound bank definition can be initialized by firmware in case of ROM MIDI sound bank. Sound bank definition can be modified by user.

MIDI Sound bank name
8 ASCII character corresponding to the MIDI sound bank file name *.94b
Unused MIDI sound bank are filled with 8*00H.

MIDI Sound priority level
Range 0 to 127

Midi Mapping Table

Although each bank has its own MIDI mapping a single MIDI mapping table is used by the 9407 board.
This single MIDI mapping table is created/updated by SBE and used by 9407 board software.
Sound bank exchanges with 9407 board always begin with downloading of MIDI mapping from 9407 board memory, then SBE updates MIDI mapping table and uploads the table back to 9407 board memory.

MIDI mapping table describes current MIDI mapping but also includes additional information to recover instruments that were replaced with an higher priority sound bank.

Those two kinds of sounds are referred in the instrument mapping table as

Current instrument/Drumset

Backup instrument/Drumset

MIDI mapping table:

Nb	Type	Content
1	word	total table length
16	word	***sound bank page***
128	word	16bits pointer to ***instrument program table***
	variable	***Drumset table***
	word	FFFF end of block

Sound bank page:
Sound bank parameter maximum size is 128Kword. If block size is superior to 64K, it should be split into two 64K pages.
SBE can manage up to 8 sound banks.
The 16 word sound bank page table actually holds 8 sound bank with 2 page parameter each. Even sound bank page refer to the first parameter block and odd to the second parameter block.

Instrument program table:

Nb	Type	Content
n	long	current ***instrument definition***
	word	FFFF end of block
p	long	backup ***instrument definition***
	word	FFFF end of block

Instrument definition:

Nb	Type	Content
	byte	variation number
	byte	sound bank number, actually sound bank nb *2 + 0 (for 1st param block) / 1 (for 2nb param block)
	word	16bit pointer to instrument parameter

Drumset table:

Nb	Type	Content
n	3*word	current ***Drumset definition***

	word	FFFF end of block
p	3*word	backup ***Drumset definition***
	word	FFFF end of block

Drumset definition:

Nb	Type	Content
	word	program number
	byte	variation number
	byte	sound bank number (up to 8 bank)
	word	16bit pointer to Drumset parameter

CONTROL TABLE

The control table stores the current values of all the device parameters that can be read back by client applications.

It is a 128 bytes long table, the address corresponds to the device parameter control number (see Firmware API), Only a few control number among the 128 possible values are defined and only a few of these defined parameters can be read back.

Ctrl #	CONTROL NAME	Action
07H	MASTER_VOL	Master volume
08H	REC_MODE	Select record mode
10H	EQ_LBL	Equalizer low band left
11H	EQ_MLBL	Equalizer med low band left
12H	EQ_MHBL	Equalizer med high band left
13H	EQ_HBL	Equalizer high band left
14H	EQ_LBR	Equalizer low band right
15H	EQ_MLBR	Equalizer med low band right
16H	EQ_MHBR	Equalizer med high band right
17H	EQ_HBR	Equalizer high band right
18H	EQF_LB	Equalizer low band frequency
19H	EQF_MLB	Equalizer med low band frequency
1AH	EQF_MHB	Equalizer med high band frequency
1BH	EQF_HB	Equalizer high band frequency
20H	AUD_SEL	Audio1/2 input select (1)
21H	AUD_GAINL	Audio Left input gain (1)
22H	AUD_GAINR	Audio Right input gain (1)
25H	GMREV_SEND	General Midi Reverb Send
26H	GMCHR_SEND	General Midi Chorus Send
27H	AUDREV_SEND	Audio Reverb Send
28H	ECH_LEV	Echo level applied on audio in (2)
29H	ECH_TIM	Echo time applied on audio in (2)
2AH	ECH_FEED	Echo feedback applied on audio in (2)
30H	SUR_VOL	Surround effect volume
31H	SUR_DEL	Surround effect delay
32H	SUR_INP	Input mono/stereo select for surround
33H	SUR_24	2 or 4 speakers output select for surround
34H	AUDL_VOL	Left Channel Audio in volume (2)
35H	AUDR_VOL	Right Channel Audio in volume (2)
36H	AUDL_PAN	Left Channel Audio in pan (2)
37H	AUDR_PAN	Right Channel Audio in pan (2)
38H	GM_VOL	General Midi volume
39H	GM_PAN	General Midi pan
3AH	REV_VOL	Reverb general volume
3BH	CHR_VOL	Chorus general volume
3FH	UART_MOD	Switch to UART mode
61h	MOD_ASS	MOD player audio output assignment (3)
62h	GM_POST	Post effects applied on general midi (4)
63h	WAVE_POST	Post effects applied on wave (4)

64h	MOD_POST	Post effects applied on MOD player (4)
65h	AUDECH_POST	Post effects applied on Audio in and echo(4)
66h	EFF_POST	Post effects applied on Reverb-chorus (4)
68H	ECH_ONOFF	Echo On/Off (3)
69H	REV_TYPE	Reverb program select
6AH	CHR_TYPE	Chorus program select
6BH	EQU_TYPE	Equalizer type (3)
6CH	REV_ONOFF	Reverb On/Off (3)
6DH	CHR_ONOFF	Chorus On/Off (3)
6EH	SUR_ONOFF	Surround On/Off (3)
6FH	AUD_ONOFF	Audio On/Off (3)
72H	POLY_64	Enable 64 voice polyphony (3)
74H	CHR_DEL	Chorus delay
75H	CHR_FEED	Chorus feedback
76H	CHR_RATE	Chorus rate
77H	CHR_DEPTH	Chorus depth
78H	REV_TIME	Reverb time
79H	REV_FEED	Reverb feedback

Chapter 4

MIDI Implementation

MIDI MESSAGE	HEX CODE	DESCRIPTION	COMPATIBI-LITY
NOTE ON	9nH kk vv	Midi channel n(0-15) note ON #kk(1-127), velocity vv(1-127). vv=0 means NOTE OFF	MIDI
NOTE OFF	8nH kk vv	Midi channel n(0-15) note OFF #kk(1-127), vv is don' t care.	MIDI
PITCH BEND	EnH bl bh	Pitch bend as specified by bh|bl (14 bits) Maximum swing is +/- 1 tone (power-up). Can be changed using « pitch bend sensitivity ». Center position is 00H 40H.	GM
PROGRAM CHANGE	CnH pp	Program (patch) change. Specific action on channel 10 (n=9) : select drumset. Refer to sounds / drumset list. Drumsets can be assigned to other channels (see SYSEX MIDI channel to part assign and part to rhythm allocation)	GM/GS
CHANNEL AFTERTOUCH	DnH vv	vv pressure value. Effect set using Sys. Ex. 40H 2nH 20H-26H	MIDI
MIDI RESET	FFH	Reset to power-up condition	
CTRL 00	BnH 00H cc	Bank select : Refer to sounds list. No action on drumset. cc=64 reserved for dream sound editor	GS/ DREAM
CTRL 01	BnH 01H cc	Modulation wheel. Rate and maximum depth can be set using SYSEX	MIDI
CTRL 05	BnH 05H cc	Portamento time.	MIDI
CTRL 06	BnH 06H cc	Data entry : provides data to RPN and NRPN	MIDI
CTRL 07	BnH 07H cc	Volume (default=100)	MIDI
CTRL 10	BnH 0AH cc	Pan (default=64 center)	MIDI
CTRL 11	BnH 0BH cc	Expression (default=127)	MIDI/GM
CTRL 64	BnH 40H cc	Sustain (damper) pedal	MIDI
CTRL 65	BnH 41H cc	Portamento ON/OFF	MIDI
CTRL 66	BnH 42H cc	Sostenuto pedal	MIDI
CTRL 67	BnH 43H cc	Soft pedal	MIDI
CTRL 80	BnH 50H vv	Reverb program vv=00H to 07H (default 04H) 00H : Room1 01H : Room2 02H : Room3 03H : Hall1 04H : Hall2 05H : Plate 06H : Delay 07H : Pan delay	DREAM
CTRL 81	BnH 51H vv	Chorus program vv=00H to 07H (default 02H) 00H : Chorus1 01H : Chorus2 02H : Chorus3 03H : Chorus4 04H : Feedback 05H : Flanger 06H : Short delay 07H : FB delay	DREAM
CTRL 91	BnH 5BH vv	Reverb send level vv=00H to 7FH	GS
CTRL 93	BnH 5DH vv	Chorus send level vv=00H to 7FH	GS
CTRL 120	BnH 78H 00H	All sound off (abrupt stop of sound on channel n)	MIDI
CTRL 121	BnH 79H 00H	Reset all controllers	MIDI
CTRL 123	BnH 7BH 00H	All notes off	MIDI
CTRL 126	BnH 7EH 00H	Mono on	MIDI
CTRL 127	BnH 7FH 00H	Poly on (default power-up)	MIDI
CTRL CC1	BnH ccH vvH	Assignable Controller 1. cc=Controller number (0-5Fh), vv=Control value (0-7Fh). Control number (ccH) can be set on CC1	GS

		CONTROLLER NUMBER (Sys. Ex 40 1x 1F). The resulting effect is determined by CC1 controller function (Sys.Ex. 40 2x 40-4A)	
CTRL CC2	BnH ccH vvH	Assignable Controller 2. cc=Controller number (00h-5Fh), vv=control value (0-7Fh). Control number can be set on CC2 CONTROLLER NUMBER (Sys.Ex. 40 1x 20). The resulting effect is determined by CC2 controller function (Sys.Ex.40 2x 50-5A).	
RPN 0000H	BnH 65H 00H 64H 00H 06H vv	Pitch bend sensitivity in semitones (default=2)	MIDI/GM
RPN 0001H	BnH 65H 00H 64H 01H 06H vv	Fine tuning in cents (vv=00 -100, vv=40H 0, vv=7FH +100	MIDI
RPN 0002H	BnH 65H 00H 64H 02H 06H vv	Coarse tuning in half-tones (vv=00 -64, vv=40H 0, vv=7FH +64	MIDI
NRPN 0108H	BnH 63H 01H 62H 08H 06H vv	Vibrate rate modify (vv=40H -> no modif)	GS
NRPN 0109H	BnH 63H 01H 62H 09H 06H vv	Vibrate depth modify (vv=40H -> no modif)	GS
NRPN 010AH	BnN 63H 01H 62H 0AH 06H vv	Vibrate delay modify (vv=40H -> no modif)	GS
NRPN 0120H	Bnh 63H 01H 62H 20H 06H vv	TVF cutoff freq modify(vv=40H -> no modif)	GS
NRPN 0121H	BnH 63H 01H 62H 21H 06H vv	TVF resonance modify (vv=40H -> no modif)	GS
NRPN 0163H	Bnh 63H 01H 62H 63H 06H vv	Env. attack time modify(vv=40H ->no modif)	GS
NRPN 0164H	BnH 63H 01H 62H 64H 06H vv	Env. decay time modify(vv=40H -> no modif)	GS
NRPN 0166H	BnH 63H 01H 62H 66H 06H vv	Env. release time modif(vv=40H ->no modif)	GS
NRPN 18rrH	BnH 63H 18H 62H rr 06H vv	Pitch coarse of drum instr. note rr in semitones (vv=40H -> no modif)	GS
NRPN 1ArrH	BnH 63H 1AH 62H rr 06H vv	Level of drum instrument note rr (vv=00 to 7FH)	GS
NRPN 1CrrH	BnH 63H 1CH 62H rr 06H vv	Pan of drum instrument note rr (40H = middle)	GS
NRPN 1DrrH	BnH 63H 1DH 62H rr 06H vv	Reverb send level of drum instrument note rr (vv=00 to 7FH)	GS
NRPN 1ErrH	BnH 63H 1EH 62H rr 06H vv	Chorus send level of drum instrument note rr (vv=00 to 7FH)	GS
NRPN 37xxH	BnH 63H 37H 62H xx 06H vv	Special SAM9407 features controls (see below)	DREAM
Standard Sysex	F0H 7EH 7FH 09H 01H F7H	General MIDI reset	GM
Standard Sysex	F0H 7FH 7FH 04H 01H 00H ll F7H	Master volume (ll=0 to 127, default 127)	GM
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 00H 00H dd dd dd dd xx F7H	Master tune (default dd= 00H 04H 00H 00H) -100.0 to +100.0 cents. Nibblized data should be used (always four bytes). For example, to tune to +100.0 cents, sent data should be 00H 07H 0EH 08H	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 00H 04H vv xx F7H	Master volume (default vv=7FH)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 00H 05H vv xx F7H	Master key-shift (default vv=40H, no transpose)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 00H 06H vv xx F7H	Master pan (default vv=40H, center)	
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 00H 7FH 00H xx F7H	GS reset	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 30H vv xx F7H	Reverb type (vv=0 to 7), default = 04H 00H : Room1 01H : Room2 02H : Room3 03H : Hall1 04H : Hall2 05H : Plate 06H : Delay 07H : Pan delay	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 31H vv xx F7H	Reverb character, default 04H	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 33H vv xx F7H	Reverb master level, default = 64	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 34H vv xx F7H	Reverb time	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 35H vv xx F7H	Reverb delay feedback. Only if reverb number=6 or 7 (delays)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 38H vv xx F7H	Chorus type (vv=0 to 7), default = 02H 00H : Chorus1 01H : Chorus2 02H : Chorus3 03H : Chorus4 04H : Feedback 05H : Flanger 06H : Short delay 07H : FB delay	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 3AH vv xx F7H	Chorus master level, default = 64	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 3BH vv xx F7H	Chorus feedback	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 3CH vv xx F7H	Chorus delay	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 3DH vv xx F7H	Chorus rate	GS

ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 01H 3EH vv xx F7H	Chorus depth	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 1pH 02H nn xx F7H	MIDI channel to part assign, p is part (0 to 15), nn is MIDI channel (0 to 15, 16=OFF). This SYSEX allows to assign several parts to a single MIDI channel or to mute a part. Default assignment : part MIDI channel 0 9 (DRUMS) 1-9 0-8 10-15 10-15	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 1pH 15H vv xx F7H	Part to rhythm allocation, p is part (0 to 15), vv is 00 (sound part) or 01 (rhythm part). This SYSEX allows a part to play sound or drumset. There is no limitation of the number of parts playing drumset. Default assignment : part 0 plays drums (default MIDI channel 9) all other parts play sound.	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 1nH 40H v1 v2 ... v12 xx F7H	Scale tuning, n is MIDI channel (0 to 15), v1 to v12 are 12 semi-tones tuning values (C, C#, D, ... A#, B), in the range -64 (00H) 0 (40H) +63(7FH) cents. This SYSEX allows non chromatic tuning of the musical scale on a given MIDI channel. Default v1, v2, ... ,v12 = 40H, 40H,...,40H (chromatic tuning). Scale tuning has no effect if the part is assigned to a rhythm channel or if the sound played is not of chromatic type.	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 1nH 1AH vv xx F7H	Velocity slope from 00H to 7FH (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 1nH 1BH vv xx F7H	Velocity offset from 00H to 7FH (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 1nH 1FH vv xx F7H	CC1 Controller number (00-5FH) (default = 10H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 1nH 20H vv xx F7H	CC2 Controller number (00-5FH) (default = 11H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 00H vv xx F7H	Mod pitch control (-24,+24 semitone) (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 01H vv xx F7H	Mod tvf cutoff control (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 02H vv xx F7H	Mod Amplitude control (-100%-+100%) (default=40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 03H vv xx F7H	Mod lfo1 rate control (default = 40H). n is don' t care. Rate is common on all channels	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 04H vv xx F7H	Mod lfo1 pitch depth (0-600 cents) (default=0AH)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 05H vv xx F7H	Mod lfo1 tvf depth (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 06H vv xx F7H	Mod lfo1 tva depth (0-100%) (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 10H vv xx F7H	Bend pitch control (-24,+24 semitone) (default = 42H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 11H vv xx F7H	Bend tvf cutoff control (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 12H vv xx F7H	Bend Amplitude control (-100%-+100%) (default=40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 14H vv xx F7H	Bend lfo1 pitch depth (0-600 cents) (default=0AH)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 15H vv xx F7H	Bend lfo1 tvf depth (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 16H vv xx F7H	Bend lfo1 tva depth (0-100%) (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 20H vv xx F7H	CAF pitch control (-24,+24 semitone) (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 21H vv xx F7H	CAF tvf cutoff control (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 22H vv xx F7H	CAF Amplitude control (-100%-+100%) (default=40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 24H vv xx F7H	CAF lfo1 pitch depth (0-600 cents) (default=0AH)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 25H vv xx F7H	CAF lfo1 tvf depth (default = 0H)	GS

ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 26H vv xx F7H	CAF lfo1 tva depth (0-100%) (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 40H vv xx F7H	CC1 pitch control (-24,+24 semitone) (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 41H vv xx F7H	CC1 tvf cutoff control (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 42H vv xx F7H	CC1 Amplitude control (-100%-+100%) (default=40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 44H vv xx F7H	CC1 lfo1 pitch depth (0-600 cents) (default=0AH)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 45H vv xx F7H	CC1 lfo1 tvf depth (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 46H vv xx F7H	CC1 lfo1 tva depth (0-100%) (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 50H vv xx F7H	CC2 pitch control (-24,+24 semitone) (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 51H vv xx F7H	CC2 tvf cutoff control (default = 40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 52H vv xx F7H	CC2 Amplitude control (-100%-+100%) (default=40H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 54H vv xx F7H	CC2 lfo1 pitch depth (0-600 cents) (default=0AH)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 55H vv xx F7H	CC2 lfo1 tvf depth (default = 0H)	GS
ROLAND SYSEX	F0H 41H 00H 42H 12H 40H 2nH 56H vv xx F7H	CC2 lfo1 tva depth (0-100%) (default = 0H)	GS

notes :

NRPN sending method : CTRL#99=high byte, CTRL#98=low byte, CTRL#6=vv
Example : NRPN 0108H = 40H -> CTRL#99=1, CTRL#98=8, CTRL#6=64

x or xx means « don’t care »

DREAM SPECIAL NRPN CONTROLS

The various features of the SAM9407 are controlled by NRPN MIDI messages as follows :

NRPN # (High|Low)	Description	Power-up default
3700H	Equalizer Low band (bass) 0=-12dB, 40H=0dB, 7FH=+12dB	60H (+6dB)
3701H	Equalizer Med Low band 0=-12dB, 40H=0dB, 7FH=+12dB	40H (0dB)
3702H	Equalizer Med High band 0=-12dB, 40H=0dB, 7FH=+12dB	40H (0dB)
3703H	Equalizer High band (treble) 0=-12dB, 40H=0dB, 7FH=+12dB	60H (+6dB)
3708H	Equalizer Low cutoff freq 0=0Hz, 7FH=4.7 kHz	0CH
3709H	Equalizer Med Low cutoff freq 0=0Hz, 7FH=4.2 kHz	1BH
370AH	Equalizer Med High cutoff freq 0=0Hz, 7FH=4.2 kHz	72H
370BH	Equalizer High cutoff freq 0=0Hz, 7FH=18.75 kHz	40H
3710H	Input select 0=select microphones 7FH=select AUXL/ AUXR (note 1)	00H (mikes)
3711H	Mike1/AUXL input gain 0=0dB to 7FH=+22.5dB (note 1)	00H (0dB)
3712H	Mike2/AUXR input gain 0=0dB to 7FH=+22.5dB (note 1)	00H (0dB)
3720H	Surround effect volume 0= no effect, 7FH= maximum effect	00H
3724H	Mike1/AUXL volume 0 to 7FH	40H
3725H	Mike2/AUXR volume 0 to 7FH	40H
3726H	Mike 1/AUXL pan 0=hard left, 40H=center, 7FH=hard right	00H (left)
3727H	Mike 2/AUXR pan 0=hard left, 40H=center, 7FH=hard right	7FH (right)
3728H	Mike/AUX echo level 0 to 7FH	40H
3729H	Mike/AUX echo time 0=shortest to 7FH=longest	40H
372AH	Mike/AUX echo feed-back 0=no feed back to 7FH=maximum feedback	40H
372CH	Surround effect delay 0=shortest to 7Fh=longest	2H
372DH	Surround effect input 0=stereo, 7Fh=mono	0H
372EH	Surround effect output mode 0=2 speaker mode, 7Fh=4 speaker mode	0H

note 1 : Aux inputs are available only with CS4216/CS4218 type DAC/ADC.