CtSetMixerLevelForFMLeft

CtSetMixerLevelForFMRight
CtSetMixerLevelForLeftSamplePb
CtSetMixerLevelForRightSamplePb
CtSetMixerLevelForAuxLeft
CtSetMixerLevelForAuxRight
CtSetMixerLevelForMicrophone
CtSetMixerLevelForTelephone

Syntax

WORD **CtSetMixerLevelForFMLeft**(WORD value)
WORD **CtSetMixerLevelForFMRight**(WORD value)
WORD **CtSetMixerLevelForLeftSamplePb**(WORD value)
WORD **CtSetMixerLevelForRightSamplePb**(WORD value)
WORD **CtSetMixerLevelForAuxLeft**(WORD value)
WORD **CtSetMixerLevelForAuxRight**(WORD value)
WORD **CtSetMixerLevelForMicrophone**(WORD value)
WORD **CtSetMixerLevelForTelephone**(WORD value)

Sets the volume for the specified device

Parameters

WORD value

Volume level from 128 to 255 whereis 128 is the minimum, 255 the maximum.

Return Value

1 if ok.

Comments

Writing a value less than 128 will result in a signal with negative polarity and should be avoided because the resulting signal may cancel out another signal of opposite polarity.

CtGetMixerLevelForFMLeft

CtGetMixerLevelForFMRight
CtGetMixerLevelForLeftSamplePb
CtGetMixerLevelForRightSamplePb
CtGetMixerLevelForAuxLeft
CtGetMixerLevelForAuxRight
CtGetMixerLevelForMicrophone
CtGetMixerLevelForTelephone

Syntax

WORD CtGetMixerLevelForFMLeft()
WORD CtGetMixerLevelForFMRight()
WORD CtGetMixerLevelForLeftSamplePb()
WORD CtGetMixerLevelForRightSamplePb()
WORD CtGetMixerLevelForAuxLeft()
WORD CtGetMixerLevelForAuxRight()
WORD CtGetMixerLevelForMicrophone()
WORD CtGetMixerLevelForTelephone()

Returns the volume of the specified device.

Parameters

None

Return Value

Volume level from 128 to 255 whereis 128 is the minimum, 255 the maximum.

Comments

None

CtSetOutputVolumeLeft

CtSetOutputVolumeRight

Syntax

WORD **CtSetOutputVolumeLeft**(WORD value)
WORD **CtSetOutputVolumeRight**(WORD value)

Sets the final output volume

Parameters

WORD value

Volume level from 0 to 255

Return Value

1 if ok.

Comments

There are actually 64 final volume levels. The driver divides the specified value by 4.

CtGetOutputVolumeLeft

CtGetOutputVolumeRight

Syntax

WORD CtGetOutputVolumeLeft()
WORD CtGetOutputVolumeRight()

Returns the the final output volume

Parameters

None

Return Value

Final output volumefrom 0 to 255

Comments

There are actually 64 final volume levels. The driver multiplies the specified value by 4 in the return value.the return value may not correspond exactly to the value specified with CTSetOutputVolumeXXX().

CtSetOutputBassLevel

CtSetOutputTrebleLevel

Syntax

WORD **CtSetOutputBassLevel**(WORD value)
WORD **CtSetOutputTrebleLevel**(WORD value)

Sets the output bass and treble level.

Parameters

WORD value

Range from -128 to 127.

Return Value

1 if ok.

Comments

Negative values decreases trebleor bass, positive numbers, increase treble or bass. 0 does not alter sound.

CtGetOutputBassLevel

CtGetOutputTrebleLevel

Syntax

WORD CtGetOutputBassLevel()
WORD CtGetOutputTrebleLevel()

Returns the bass or treble level setting.

Parameters

None

Return Value

Bass or treble setting, from -127 to 127

Comments

Since only 4 bits are actually used in the control Chip, the result obtained can differ with the value written using the CtSetOutputBassLevel() and CtSetOutputTrebleLevel function, due to rounding errors.

CtEnabDisabOutputMuting

Syntax

WORD **CtEnabDisabOutputMuting**(value)

Disables or enables output muting.

Parameters

WORD value

0 = disable, 1 = enable

Return Value

1 if ok.

Comments

None

CtGetEnabDisabOutputMuting

Syntax

WORD CtGetEnabDisabOutputMuting()

Returns a value indicating if output muting is disabled or enabled.

Parameters

None

Return Value

0: disabled, 1: enabled

Comments

None

CtSelectSCSIInterruptNumber

Syntax

WORD **CtSelectSCSIInterruptNumber**(WORD value)

Selects an interrupt request line for the SCSI hardware on the Goldcard.

Parameters

WORD value

0 = IRQ3
1 = IRQ4
2 = IRQ5
3 = IRQ7
4 = IRQ10
5 = IRQ11
6 = IRQ12
7 = IRQ15

Return Value

1 if ok.

Comments

Valid interrupt lines on an XT are IRQ3, IRQ4, IRQ5 and, IRQ7.
Valid interrupt lines on an AT are IRQ3, IRQ4, IRQ5, IRQ7, IRQ10, IRQ11, IRQ12 and IRQ15.

CtGetSCSIInterruptNumber

Syntax

WORD CtGetSCSIInterruptNumber()

Returns a number indicating the interrupt request line used by the SCSI hardware on the Gold card.

Parameters

None

Return Value

Interrupt request line:

0 = IRQ3
1 = IRQ4
2 = IRQ5
3 = IRQ7
4 = IRQ10
5 = IRQ11
6 = IRQ12
7 = IRQ15

Comments

None

CtEnabDisabSCSIInterrupt

Syntax

WORD **CtEnabDisabSCSIInterrupt**(value)

Disables or enables interrupt from SCSII.

Parameters

WORD value

0 = disable, 1 = enable

Return Value

1 if ok.

Comments

None

CtEnabDisabSCSIDMA

Syntax

WORD **CtEnabDisabSCSIDMA**(value)

Disables or enables DMA transfers on SCSI hardware.

Parameters

WORD value

0 = disable, 1 = enable

Return Value

1 if ok.

Comments

None

CtGetEnabDisabSCSIInterrupt

Syntax

WORD CtGetEnabDisabSCSIInterrupt()

Returns 1 if interrupts are enabled on the SCSI hardware.

Parameters

None

Return Value

0: Interrupts are disabled
1: Interrupts are enabled

Comments

None

CtGetEnabDisabSCSIDMA

Syntax

WORD CtGetEnabDisabSCSIDMA()

Returns 1 if DMA transfers are enabled on the SCSI hardware.

Parameters

None

Return Value

0: DMA is disabled
1: DMA is enabled

Comments

None

CtSelectSCSIDMAChannel

Syntax

WORD **CtSelectSCSIDMAChannel**(WORD value)

Assigns a DMA channel to the SCSI hardware of the Gold Card.

Parameters

WORD value

0 = DMA 0
1 = DMA 1
2 = DMA 2
3 = DMA 3

Return Value

1 if ok.

Comments

Valid DMA channels are 0 - 3. Other channel numbers are reserved for future extensions.

CtGetSCSIDMAChannel

Syntax

WORD CtGetSCSIDMAChannel()

Returns the number of the DMA channel Assigned to the SCSI hardware of the Gold card.

Parameters

None

Return Value

0 = DMA 0
1 = DMA 1
2 = DMA 2
3 = DMA 3

Comments

None

CtSetSCSIRelocationAddress

Syntax

WORD **CtSetSCSIRelocationAddress**(value)

Sets the base port address addresses for SCSI controller.

Parameters

WORD value

New base I/O address divided by 8.
Range from 0 to 127.

Return Value

1 if ok.

Comments

None

CtGetSCSIRelocationAddress

Syntax

WORD CtGetSCSIRelocationAddress()

Returns the base port address for SCSI controller.

Parameters

None

Return Value

New base I/O address divided by 8.
Range from 0 to 127.

Comments

None

CtSetHangUpPickUpTelephoneLine

Syntax

WORD **CtSetHangUpPickUpTelephoneLine**(WORD value)

Hangs up or picks up telephone.

Parameters

WORD value

0 = Disconnect telephone line,
1 = Connect telephone line

Return Value

1 if ok.

Comments

None

CtGetHangUpPickUpTelephoneLine

Syntax

WORD CtGetHangUpPickUpTelephoneLine()

Returns a value telling if the telephone line is on-hook or off-hook.

Parameters

None

Return Value

0: telephone line is on-hook (not connected)
1: telephone line is off-hook (connected)

Comments

None

CtSelectOutputSources

Syntax

WORD **CtSelectOutputSources**(value)

Selects final output mixing redirection.

Parameters

WORD value

0 = left mixer channel to left output & right mixer channel to right output,
1 = left mixer channel to both left and right outputs,
2 = right mixer channel to both left and right outputs.

Return Value

1 if ok.

Comments

On the Adlib Gold cards, mixing and volume control is performed in two stages. First, all sources are sent to a stereo mixer. Then, the stereo output of the mixer is fed into the final volume control circuitry. The final left and right outputs can be mixed in the fashion described above.

CtGetOutputSources

Syntax

WORD CtGetOutputSources()

Returns the final mixer redirection mode.

Parameters

None

Return Value

0 = left mixer channel to left output & right mixer channel to right output,
1 = left mixer channel to both left and right outputs,
2 = right mixer channel to both left and right outputs.

Comments

None

CtSelectOutputMode

Syntax

WORD **CtSelectOutputMode**(value)

Controls the effect applied to the final output .

Parameters

WORD value

0 = Forced mono,
1 = linear stereo,
2 = pseudo stereo,
3 = spatial stereo.

Return value

1 if ok.

Comments

Linear stereo is ordinary, with no effects added. The spatial and pseudo-stereo effects will be useful primarily when the original source is monophonic.

CtGetOutputMode

Syntax

WORD CtGetOutputMode()

Returns the effect applied to the final output .

Parameters

None

Return value

0 = Forced mono,
1 = linear stereo,
2 = pseudo stereo,
3 = spatial stereo.

Comments

None

GetControlRegister

Syntax

WORD **GetControlRegister**(reg)

Returns value stored on register 'reg' of Ad Lib Control Chip.

Parameters

int reg

Which register to read from.

Return value

Returns the WORD at the register position.

Comments

None

CtGetBoardIdentificationCode

Syntax

WORD CtGetBoardIdentificationCode()

Returns the board identification code.

Parameters

None

Return value

Board identification code:

0 - Gold 2000,
1 - Gold 1000,
2 - Gold 2000 MC.

Comments

None

CtGetBoardOptions

Syntax

WORD CtGetBoardOptions()

Returns a bit pattern indicating the options present on boardpresent

Parameters

None

Return value

Bit 0-3 (0 = not present, 1 = installed)

bit 0 - Telephone,
bit 1 - Surround,
bit 2 - SCSI,
bit 3 - Currently unused

Comments

None

CtGetControllerStatus

Syntax

WORD CtGetControllerStatus()

Returns the interrupt controller status.

Parameters

None

Return value

bit 0 -	equals 1 when an OPL3 interrupt is pending,
bit 1 -	equals 1 when an MMA interrupt is pending,
bit 2 -	equals 1 when an telephone interrupt is pending,
bit 3 -	equals 1 when a SCSI interrupt is pending,
bit 6 -	equals 1 when the Control Chip is currently, occupied writing a value to the Mixer Chip or the Volume Control Chip.
bit 7	Set to 1 when the Control Chip is busy writing its internal registers to the external EEPROM chip. This bit must be polled after activating the "Store configuration" sequence to make sure that the Control Chip is free to proceed with another operation.

Comments

Bit 7 and Bit 6 are polled by all set functions, prior to writing to the registers, to make sure that the Control Chip is free to proceed with another operation.

CtGetRingTelephoneStatus

Syntax

WORD CtGetRingTelephoneStatus()

Gets telephone status.

Parameters

None

Return value

bit 0: "Ring signal" (0 = no ring, 1 = ring)

Comments

None

CtGetInterruptRoutine

Syntax

WORD CtGetInterruptRoutine()

This routine returns the corresponding interrupt number associated with the interrupt request line used by the audio section.

Parameters

None

Return value

Corresponding interrupt number

Comments

Useful utility mostly used when setting interrupt vectors.

CtGetGoldCardPresence

Syntax

WORD CtGetGoldCardPresence()

Checks for Gold card presence.

Parameters

None

Return value

1 if any Gold card is found. 0 if no Gold card is found.

Comments

None

CtGetDriverPresence

Syntax

WORD CtGetDriverPresence()

Checks for Ad Lib Gold Control Driver.

Parameters

None

Return value

1 if the Gold Control driver is found. Returns 0 otherwise.

Comments

None

CtProgramSurroundPreset

Syntax

WORD **CtProgramSurroundPreset**(ptrData)

This routine will store a preset into the surround module.
The preset is defined by a 32 bytes array passed as argument.

Parameters

BYTE *ptrData

Pointer to the array of 32 bytes.

Return value

0 if no error, otherwise 1, no surround module.

Comments

The 1 bytes of the Surround Preset are a 1 to 1 image of the 32 registers of the Surround processor.

5.3 DOS FM Synthesis Driver

The Ad Lib Gold FM Synthesis Driver offers services to access features of the OPL3 FM Chip.

Voice Allocation Structure

The OPL3 chip contains 36 operators which can be combined in various ways to create 1-, 2- or 4-operator voices. (You may wish to refer to the "FM Driver Voices" table on the next page for the purposes of this discussion.)

The 4-operator voices offer the richest sound. Up to six 4-operator voices can be used simultaneously. In the FM Driver, the 4-operator voices are numbered 0, 2, 4, 6, 8 and 10. By default, all six 4-operator voices are enabled. They may be selectively disabled, thus creating two 2-operator voices.

In the FM Driver, when 4-operator voice *x* is disabled, the two 2-operator voices are numbered *x* and *x*+1. For example, if 4-operator voice #2 was disabled, the resulting 2-operator voices will be numbered 2 and 3.

Use Set4OpMaskOPL3() to determine the grouping of the units in either 2 operator or 4 operator voices.

Six of the chip's operators can only be used as three 2-operator voices. These three voices are numbered 12, 13 and 14.

The configuration of the remaining 6 operators depends on whether the card is in melodic or percussive mode. In melodic mode, these 6 operators are configured as three 2-operator voices: driver voice numbers 15, 16 and 18. In percussive mode, the 6 operators are used to create one 2-operator voice (the bass drum) and four 1-operator voices (the remaining drum sounds). The percussive voices are driver voice numbers 15 through 19.

Use SetPercModeOPL3() to configure this section in the melodic or percussive mode.

4 operator voice number	2 operator voice number	Percussive voice number
0	0, 1	-
2	2,3	-
4	4,5	-
6	6,7	-
8	8,9	-
10	10,11	-
-	12	-
-	13	-
-	14	-
-	15	15 (BD)
-	16	16 (HH)
-	-	17 (SD)
-	18	18 (TOM)
-	-	19 (CYMB)

FM Driver Voices

Function Directory

The following section is an alphabetically arranged definition of all the functions available in the FM Synthesis Driver.

InitOPL3

Syntax

void InitOPL3(address)

Initializes the FM Chip.

Parameters

WORD address

Port address of the FM chip.

Comments

After initialization, percussion voices are available and all 4 op-voices are enabled.

LeftRightOPL3

Syntax

void LeftRightOPL3(voiceNum, leftRight)

Modifies the stereo position of the voice.

Parameters

int voiceNums

VoiceNumber between 0 and 19.

int leftRight

Position of the specified voice:

0: Center.
1: Left.
2: Right.

LevelOPL3

Syntax

void LevelOPL3(voiceNum, level)

Specify the individual volume for a voice.

Parameters

int voiceNum

Voice number between 0 and 19

int level

Volume for the voice.
This in an integer number between 0 and 127.
Volume scaling is linear.

Comments

The volume is scaled linearly by the driver software.

NoteOffOPL3

Syntax

void NoteOffOPL3(voiceNum)

Starts the decay of the timbre currently playing on the voice.

Parameters

int voiceNum

VoiceNumber between 0 and 19.

NoteOnOPL3

Syntax

void NoteOnOPL3(voiceNum, note)

Starts playing a note on the specified voice.

Parameters

int voiceNum

VoiceNumber between 0 and 19.

int note

MIDI value for the note played, in the range 12-107.

Comments

If a note is already playing on the specified voice, the frequency of the voice will be modified. However, the attack for the timbre will not be heard. To reattack the timbre on the specified voice, a NoteOffOPL3 must be issued.

PitchbendOPL3

Syntax

void PitchBendOPL3(voiceNum, pitchBend)

Modifies the pitch bend scaling factor for the melodic voice.

Parameters

int voiceNum

Melodic voiceNumber between 0 and 15.

WORD pitchBend

Pitch bend scaling factor within the range set in SetGlobalOPL3().

The pitch bend scaling factor is a 14 bit unsigned value. 0 is the maximum negative pitch bend, 0x2000 is no bend and 0x3FFF is the maximum positive pitch bend.

Comments

Percussive voices cannot be bent.

PresetOPL3

Syntax

void PresetOPL3(voiceNum, timbrePtr)

Assigns a patch to the specified voice.

Parameters

int voiceNum

voiceNumber between 0 and 19

struct TIMBRE *timbrePtr

pointer to a description (28 bytes) of the patch assigned to the voice.

Comments

If a 4 operator description is sent to a 2-op voice, only the first two operators are considered.

Appendix A: FM Patch format further describes the structure pointed to by timbrePtr.

QuitOPL3

Syntax

void QuitOPL3()

Resets the FM chip in the compatible mode.

Parameters

None.

Comments

This should be called by all applications prior to leaving, in order to put the OPL3 chip back in the Ad Lib compatible mode.

Set4OpMaskOPL3

Syntax

void Set4OpMaskOPL3(mask)

Enables or disables 4-op voices.

Parameters

WORD mask

Bit mask of enabled 4-op voices (in bits 0-5).

Bits 0-5 of mask specify whether the corresponding voice is in 4-op mode (bit set to 1) or in 2-op mode (bit cleared to 0).

Bit 0 corresponds to voice 0 (0-1 in 2 op), bit 1 to voice 2 (2-3 in 2 op) etc. (See to table 1 in the Voice Allocation section of this document).

Comments

There is a maximum of 6 4-op voices.

SetGlobalOPL3

Syntax

void SetGlobalOPL3 (noteSelectEnable, amplitudeModEnable, vibDepthEnable, pitchBendRange)

Modifies global operating parameters of the OPL3.

Parameters

BOOL noteSelectEnable

For future use. Set to 0 for now.

BOOL amplitudeModEnable

When non-zero, enables amplitude modulation for all timbres that have an amplitude modulation defined.

BOOL vibDepthEnable

When non-zero, enables vibrato for all timbres that have a vibrato depth defined.

int pitchBendRange

Range of the pitch bend in semitones.
Integer between 0-12.

SetPercModeOPL3

Syntax

void SetPercModeOPL3(newState)

Sets the OPL3 in melodic or percussive mode.

Parameters

BOOL newState

True for percussive mode, false for melodic mode.

Comments

If newState is true, disables melodic voices 15-18 and enables percussive voices 15-19 instead.

If newState is false, melodic voices 15-18 are enabled in place of percussive voices 15-19.

5.4 DOS Wave Driver

The Ad Lib Wave Driver is a high level software interface to the sampling hardware of the Gold Card. Its interface is inspired by the Microsoft Multimedia Wave Driver specifications. But in order to support the target hardware and software more efficiently, some adaptations were necessary. The main differences are:

- The support of ADPCM as well as PCM formats.
- The support of a stereo sample format.
- The control of multiple transfer modes from memory to hardware (polling, interrupt, DMA). (This implies an extension of the WaveFormat structure to include the new parameters.)
- The use of a callback function as a message-passing mechanism between the application and the driver during waveform recording and playback.

Some syntactical differences were introduced in the naming of functions and structures, in order to respect the Ad Lib naming conventions already in use. Please note that this specification is a preliminary document and is incomplete. More functions will be added to this preliminary specification.
The Wave Driver will first be available as a linkable library of functions. It will also be made available to developers in the form of a memory-resident driver, interacting with applications via an interrupt-driven protocol.

DOS Wave Driver Functions

The following section is an alphabetically arranged definition of all the functions available in the Wave Driver.

InitWaveDriver

Syntax

void InitWaveDriver()

Initializes the wave driver.
It is to be called only once by the application.

Parameters

None

Return value

None