AWE32/EMU8000 Programmer's Guide

Revision 1.00

By Dave Rossum

CONTENTS

License Agreement/Limitation And Disclaimer Of Warranties

IMPORTANT NOTE : by downloading and/or using the software and/or manual accompanying this license agreement, you are hereby agreeing to the following terms and conditions:

The software and related written materials, including any instructions for use, are provided on an "AS IS" basis, without warranty of any kind, express or implied. This disclaimer of warranty expressly includes, but is not limited to, any implied warranties of merchantability and/or of fitness for a particular purpose. No oral or written information given by Creative Technology Ltd., its suppliers, distributors, dealers, employees, or agents, shall create or otherwise enlarge the scope of any warranty hereunder. Licensee assumes the entire risk as to the quality and the performance of such software and licensee application. Should the software, and/or Licensee application prove defective, you, as licensee (and not Creative Technology Ltd., its suppliers, distributors, dealers or agents), assume the entire cost of all necessary correction, servicing, or repair.

RESTRICTIONS ON USE

Creative Technology Ltd. retains title and ownership of the manual and software as well as ownership of the copyright in any subsequent copies of the manual and software, irrespective of the form of media on or in which the manual and software are recorded or fixed. By downloading and/or using this manual and software, Licensee agrees to be bound to the terms of this agreement and further agrees that :

(1) Creative's BBS/FTP/Compuserve are the only online sites where users may download electronic files containing the manual and/or software,

(2) Licensee shall use the manual and/or software only for the purpose of developing licensee applications compatible with Creative’s Sound Blaster AWE32 series of products, unless otherwise agreed to by further written agreement from Creative Technology Ltd.; And,

(3) Licensee shall not distribute or copy the manual for any reason or by any means (including in electronic form) or distribute, copy, modify, adapt, reverse engineer, translate or prepare any derivative work based on the manual or software or any element thereof other than for the above said purpose, without the express written consent of Creative Technology Ltd.. Creative Technology Ltd. Reserves all rights not expressly granted to licensee in this license agreement.

LIMITATION OF LIABILITY

In no event will Creative Technology Ltd., or anyone else involved in the creation, production, and/or delivery of this document be liable to licensee or any other person or entity for any direct or other damages, including, without limitation, any interruption of services, lost profits, lost savings, loss of data, or any other consequential, incidental, special, or punitive damages,

arising out of the purchase, use, inability to use, or operation of the software, and/or licensee application, even if Creative Technology Ltd. or any authorised Creative Technology Ltd. dealer has been advised of the possibility of such damages. Licensee accepts said disclaimer as the basis upon which the software is offered at the current price and acknowledges that the price of the software would be higher in lieu of said disclaimer. Some states do not allow the limitation or exclusion of liability for incidental or consequential damages so the above limitations and exclusions may not apply to you.

Information in this document is subject to change without notice. Creative Technology Ltd. shall have no obligation to update or otherwise correct any errors in the manual and/or software even if Creative Technology Ltd. is aware of such errors and Creative Technology Ltd. shall be under no obligation to provide to Licensee any updates, corrections or bug-fixes which Creative Technology Ltd. may elect to prepare.

Creative Technology Ltd. does not warrant that the functions contained in the manual will be uninterrupted or error free and Licensee is encouraged to test the software for Licensee's intended use prior to placing any reliance thereon.

1 Introduction

1.1 Scope and Intended Purpose of this Document

This document is intended to be an official release of the EMU8000 programming within the AWE32 environment. The descriptions herein should be sufficient to allow programming of the EMU8000 for all normal functions. The focus in this document is on highly technical aspects of the EMU8000 chip. We assume that you are experienced programmer and familiar with hardware level programming.

1.2 Overview

The EMU8000 is a 32 channel wavetable synthesis chip with extensive ability to modulate the sound. In the AWE32 environment, Sound Memory comprises a General MIDI Sound ROM of 1 MB size, and Sound DRAM of a minimum size of 0.5MB.

The EMU8000 has very little on-board intelligence. It requires extensive initialization on power-up before any use can be made of its facilities. This initialization phase configures it for the AWE32 environment, brings the chip to a known, silent state, and enables audio output.

Once the EMU8000 has been initialized, sounds can be played, or Sound DRAM can be loaded with additional sounds.

The EMU8000 comprises 32 audio channels, each of which has a large number of parameters. There are also a number of global parameters which apply to all channels. The extensive register bank of the EMU8000 is accessed by means of a specialized interface.

2 EMU8000 Interface

The EMU8000 in the AWE32 environment appears as one doubleword I/O port two I/O ports which may serve as either a doubleword or two separate word ports and two word only I/O ports located relative to the BLASTER environment variable. These ports are named as follows:

Location	Size	Name	Function
BLASTER+0x400	Doubleword	Data0	Read and write of doubleword data
BLASTER+0x800	Word or Doubleword	Data1	Read and write of word and doubleword data
BLASTER+0x802	Word	Data2	Read and write of word data

BLASTER+0xC00	Word	Data3	Read and write of word data
BLASTER+0xC02	Word	Pointer	Read and write of register pointer value

NOTE : This EMU8000 I/O ports documented in table above is only valid for legacy cards. With the introduction of Plug & Play cards, EMU8000 base port is no longer guaranteed to be at BLASTER+0x400. You should read the 'E' parameter of the BLASTER environment variable to get the EMU8000 base port. For example, if the BLASTER environment is "A220 I5 D1 H5 P330 E640 T6 " , then EMU8000 ports will be located at

- 0x640-0x643
- 0xA40-0xA43
- 0xE40-0xE43

All I/O transactions must be performed as word or “doubleword” I/O transactions; no byte I/O transactions are allowed. A “doubleword” I/O transaction consists of a transfer of the LS 16 bit word of data from the specified I/O address, followed immediately by a transfer of the MS 16 bits of data from the I/O address two bytes higher.

Most reads and writes of the EMU8000 begin by writing the Pointer register. The Pointer register is a word register whose LS five bits are the Channel Number (0-31), whose next 3 bits (bits 7-5) are the Register Number (0-7), and whose MS 8 bits are Don’t Care (but conventionally zero) for a write and are random (actually a VLSI test register) during reads.

Once the pointer register has been set to the appropriate Channel and Register Numbers, the corresponding EMU8000 register can be written or read at the appropriate Data I/O port. The EMU8000 makes use of the I/O WAIT function of the bus to prevent changing the Pointer Register before a previous write transaction is complete, and to allow for reading the data from the EMU8000 internal registers before allowing completion of a read transaction.

3 EMU8000 Registers

3.1 Register Map

The following table shows the conventional register names for all the EMU8000 registers:

Name	Reg #	Cnl #	I/O Port	Data Size	Function
CPF	0	all	Data 0	DW	Current Pitch and Fractional Address[cnl]
PTRX	1	all	Data 0	DW	Pitch Target, Rvb Send, and Aux Byte[cnl]
CVCF	2	all	Data 0	DW	Current Volume and Filter Cutoff [cnl]
VTFT	3	all	Data 0	DW	Volume and Filter Cutoff Targets [cnl]
PSST	6	all	Data 0	DW	Pan Send and Loop Start Address [cnl]
CSL	7	all	Data 0	DW	Chorus Send and Loop End Address [cnl]
CCCA	0	all	Data 1	DW	Q, Control Bits, and Current Address[cnl]

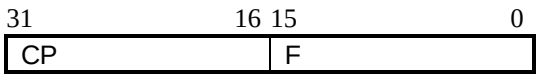
HWCF4	1	9	Data 1	DW	Configuration DoubleWord 4
HWCF5	1	10	Data 1	DW	Configuration DoubleWord 5
HWCF6	1	13	Data 1	DW	Configuration DoubleWord 6
SMALR	1	20	Data 1	DW	Sound Memory Address for “Left” SM Reads
SMARR	1	21	Data 1	DW	Sound Memory Address for “Right” SM Reads
SMALW	1	22	Data 1	DW	Sound Memory Address for “Left” SM Writes
SMARW	1	23	Data 1	DW	Sound Memory Address for “Right” SM Writes
SMLD	1	26	Data 1	Word	Sound Memory “Left” Data
SMRD	1	26	Data 2	Word	Sound Memory “Right” Data
WC	1	27	Data 2	Word	Sample Counter
HWCF1	1	29	Data 1	Word	Configuration Word 1
HWCF2	1	30	Data 1	Word	Configuration Word 2
HWCF3	1	31	Data 1	Word	Configuration Word 3
INIT1	2	all	Data 1	Word	Initialization Array 1
INIT2	2	all	Data 2	Word	Initialization Array 2
INIT3	3	all	Data 1	Word	Initialization Array 3
INIT4	3	all	Data 2	Word	Initialization Array 4
ENVVOL	4	all	Data 1	Word	Volume Envelope Delay [cnl]
DCYSUSV	5	all	Data 1	Word	Volume Envelope Sustain and Decay [cnl]
ENVVAL	6	all	Data 1	Word	Modulation Envelope Delay [cnl]
DCYSUS	7	all	Data 1	Word	Modulation Envelope Sustain and Decay [cnl]
ATKHLDV	4	all	Data 2	Word	Volume Envelope Hold and Attack [cnl]
LFO1VAL	5	all	Data 2	Word	LFO #1 Delay [cnl]
ATKHLD	6	all	Data 2	Word	Modulation Envelope Hold and Attack [cnl]
LFO2VAL	7	all	Data 2	Word	LFO #2 Delay [cnl]
IP	0	all	Data 3	Word	Initial Pitch [cnl]
IFATN	1	all	Data 3	Word	Initial Filter Cutoff and Attenuation [cnl]
PEFE	2	all	Data 3	Word	Pitch and Filter Envelope Heights [cnl]
FMMOD	3	all	Data 3	Word	Vibrato and Filter Modulation from LFO #1[cnl]
TREMFRQ	4	all	Data 3	Word	LFO #1 Tremolo Amount and Frequency [cnl]
FM2FRQ2	5	all	Data 3	Word	LFO #2 Vibrato Amount and Frequency [cnl]

3.2 Register Descriptions

These registers are functionally defined in detail as follows:

Register : CPF

Description : Current Pitch and Fractional Address [cnl]

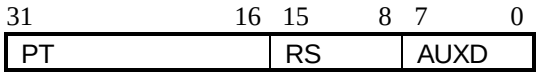


Bits 31-16 of this doubleword register will show the current pitch shift of the channel’s oscillator, with 0x4000 being no pitch shift. The current pitch is specified in linear increment.

Bits 15-0 of the register will show the current fractional address. This register is constantly being overwritten with new data, so writing to it is generally pointless.

Register : PTRX

Description : Pitch Target, Rvb Send, and Aux Byte [cnl]



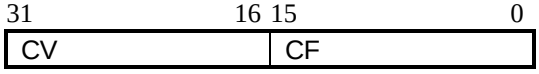
Bits 31-16 of this doubleword register determine the value to which the current pitch shift of the channel's oscillator will slew.

Bits 15-8 of the register determine the amount of Reverb Send from the channel to the effects engine, with 0x00 being none and 0xff being maximum.

Bits 7-0 of the register are an auxilliary data byte generally unused. The upper 16 bits of this register are being constantly updated by the envelope engine. so it is wise to read this value and re-write the same value when changing the reverb send.

Register : CVCF

Description : Current Volume and Filter Cutoff [cnl]

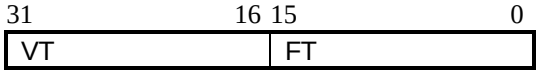


Bits 31-16 of this doubleword register show the current volume of the channel.

Bits 15-0 show the current filter cutoff value of the channel. This register is constantly being overwritten with new data, so writing to it is generally pointless.

Register : VTFT

Description : Volume and Filter Cutoff Targets [cnl]

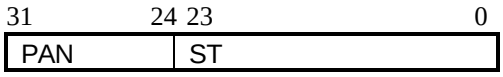


Bits 31-16 of this doubleword register determine the value to which the current volume of the channel will slew.

Bits 15-0 determine the current value to which the current filter cutoff will slew. This register is being constantly updated by the envelope engine, so it is generally not written.

Register : PSST

Description : Pan Send and Loop Start Address [cnl]

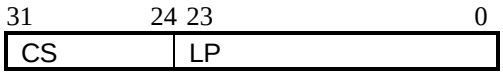


Bits 31-24 of this doubleword register determine the left-right pan of the channel. Range from 0 to 0x0FF, where 0=extreme right , 0x0FF =extreme left.

Bits 23-0 determine the start address of the loop for the channel. Note that due to interpolator offset, the actual loop point is one greater than the start address.

Register : CSL

Description : Chorus Send and Loop End Address [cnl]

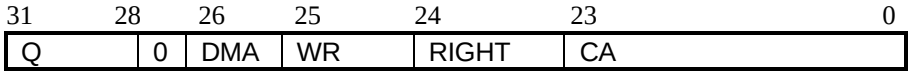


Bits 31-24 of this doubleword register determine the amount of Chorus Send from the channel to the effects engine, with 0x00 being none and 0xff being maximum..

Bits 23-0 determine the end address of the loop for the channel. Note that due to interpolator offset, the actual loop point will end at an address one greater than the loop address.

Register : CCCA

Description : Q, Control Bits, and Current Address [cnl]



Bits 31-28 of this doubleword register determine the filter Q of the channel, with 0 being no resonance, and 15 being about 24 dB of resonance.

Bit 27 should always be zero.

Bit 26 is the DMA bit for the channel, which should only be set when the channel is being used for sound memory access.

Bit 25 is the WR bit, which should only be set when bit 26 is set, and determines if the DMA channel is read or write. 1=write, 0 = read.

Bit 24 is the RIGHT bit for the channel which should be only set when bit 26 is set, and determines if the DMA channel uses the “left” or “right” DMA stream. 1=right, 0=left.

Bits 23-0 are the current sound memory address for the channel. Note that the actual audio location is the point 1 word higher than this value due to interpolator offset.

Register : HWCF4
Description : Configuration DoubleWord 4

Zero should be written to this doubleword register during the initialization process. After this, it can be ignored.

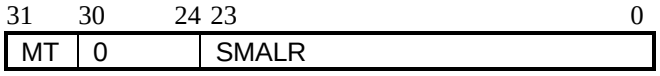
Register : HWCF5
Description : Configuration DoubleWord 5

Value 0x00000083 should be written to this doubleword register during the initialization process. After this, it can be ignored.

Register : HWCF6
Description : Configuration DoubleWord 6

Value 0x00008000 should be written to this doubleword during the initialization process. After this, it can be ignored.

Register : SMALR
Description : Sound Memory Address for “Left” SM Reads



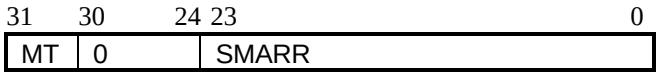
Bit 31 of this doubleword register is an “empty” bit, which indicates whether register SMLD is empty or full of data for reading. When cleared indicates the DMA data has been read from the sound memory into the SMLD register and can be read. Read only.

Bits 31-24 are Don’t Care on write, and bits 30-24 are zero on read.

Bits 23-0 are the sound memory address which will be used for “left” DMA stream data fetch the next time the SMLD read register becomes empty and a DMA channel is available to fill it.

Register : SMARR

Description : Sound Memory Address for "Right" SM Reads



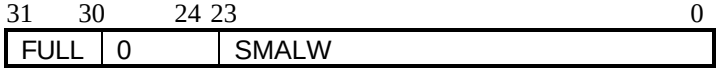
Bit 31 of this doubleword register is an "empty" bit, which indicates whether register SMRD is empty or full of data for reading. When low, the DMA data is ready to be read from the sound memory into the SMRD register. Read only.

Bits 31-24 are Don't Care on write, and bits 30-24 are zero on read.

Bits 23-0 are the sound memory address which will be used for "right" DMA stream data fetch the next time the SMRD read register becomes empty and a DMA channel is available to fill it.

Register : SMALW

Description : Sound Memory Address for "Left" SM Writes



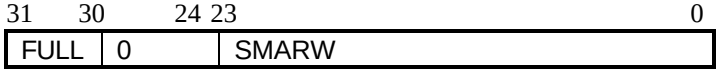
Bit 31 of this doubleword register is a "full" bit, which indicates whether register SMLD is empty or full of data for writing. When low, indicates the DMA data can be written to the SMRD register. Read only.

Bits 31-24 are Don't Care on write, and bits 30-24 are zero on read.

Bits 23-0 are the sound memory address which will be used for "left" DMA stream data write if the SMLD write register is full and a DMA channel is available to write it.

Register : SMARW

Description : Sound Memory Address for "Right" SM Writes



Bit 31 of this doubleword register is a "full" bit, which indicates whether register SMRD is empty or full of data for writing. When low, indicates the DMA data can be written to the SMRD register. Read only.

Bits 31-24 are Don't Care on write, and bits 30-24 are zero on read.

Bits 23-0 are the sound memory address which will be used for “right” DMA stream data write if the SMRD write register is full and a DMA channel is available to write it.

Register : SMLD
Description : Sound Memory “Left” Data

This word register is used for transfering data to and from sound memory using the “left” DMA stream. If one or more channels are programmed for DMA write using the left stream, a write of a word of data to this register will cause that data to be written to the address currently in SMALW. SMALW will be updated after the data is written. SMALW’s “full” bit will be set from the time the data is written in SMLD until it is transfered to sound memory. If a second write is attempted to SMLD while it is full, the I/O WAIT mechanism will hold off transfer until either the data is written or it is aborted because no channels are currently programmed for writing the left DMA stream.

If one or more channels are programmed for DMA read using the left stream, a read of a word of data from this register will transfer the current “read” contents of SMLD, and cause SMALR’s “empty” bit to be set until data can be transfered from sound memory to the SMLD read register. Data will be read from the address currently in SMALR, and SMALR will be updated after the data is read from sound memory. If a second read is attempted from SMLD while it is empty, the I/O WAIT mechanism will hold off transfer until either the data is read from sound memory or it is aborted because no channels are currently programmed for reading the left DMA stream.

Note that SMLD really consists of separate read and write registers, based on fully independent DMA streams. Note also that the read function involves a pre-fetch of data. This implies that under normal operation, one must read a word of “stale” data from the SMLD register before initiating a read transfer.

Register : SMRD
Description : Sound Memory “Right” Data

This word register is used for transfering data to and from sound memory using the “right” DMA stream. If one or more channels are programmed for DMA write using the right stream, a write of a word of data to this register will cause that data to be written to the address currently in SMARW. SMARW will be updated after the data is written. SMARW’s “full” bit will be set from the time the data is written in SMRD until it is transfered to sound memory. If a second write is attempted to SMRD while it is full, the I/O WAIT mechanism will hold off transfer until either the data is written or it is aborted because no channels are currently programmed for writing the right DMA stream.

If one or more channels are programmed for DMA read using the right stream, a read of a word of data from this register will transfer the current "read" contents of SMRD, and cause SMARR's "empty" bit to be set until data can be transfered from sound memory to the SMRD read register. Data will be read from the address currently in SMARR, and SMARR will be updated after the data is read from sound memory. If a second read is attempted from SMRD while it is empty, the I/O WAIT mechanism will hold off transfer until either the data is read from sound memory or it is aborted because no channels are currently programmed for reading the right DMA stream.

Note that SMRD really consists of separate read and write registers, based on fully independent DMA streams. Note also that the read function involves a pre-fetch of data. This implies that under normal operation, one must read a word of "stale" data from the SMRD register before initiating a read transfer.

Register : WC
Description : Sample Counter

This word register provides a counter continuously incrementing at the sample rate. There is no mechanism to reset this counter, which cycles through 65536 value every 1.486 seconds.

Register : HWCF1
Description : Configuration Word 1

Value 0x0059 should be written to this word register after power-up. After this, it can be ignored. Note: Due to a VLSI error, this register will not be correctly read by the processor.

Register : HWCF2
Description : Configuration Word 2

Value 0x0020 should be written to this word register after power-up. After this, it can be ignored. Note: Due to a VLSI error, this register will not be correctly read by the processor.

Register : HWCF3
Description : Configuration Word 3

Value 0x0004 should be written to this word register after all other initialization is complete. After this, it can be ignored. Note: Due to a VLSI error, this register will not be correctly read by the processor.

Register : INIT1
Description : Initialization Array 1

During initialization, values will be written to this word array. After this process is complete, the register can be ignored.

Register : INIT2
Description : Initialization Array 2

During initialization, values will be written to this word array. After this process is complete, the register can be ignored.

Register : INIT3
Description : Initialization Array 3

During initialization, values will be written to this word array. After this process is complete, the register can be ignored.

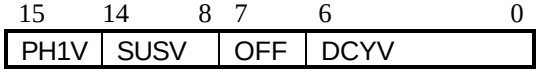
Register : INIT4
Description : Initialization Array 4

During initialization, values will be written to this word array. After this process is complete, the register can be ignored.

Register : ENVVOL
Description : Volume Envelope Delay [cnl]

This word register is programmed with the channel's volume envelope delay value at the beginning of the volume envelope. A value of 0x8000 indicates no delay; values below 0x8000 indicate increasing delay in 725 usec units.

Register : DCYSUSV
Description : Volume Envelope Sustain and Decay [cnl]



Bit 15 of this word register determines if the values written to this channel register are decay (0) or release (1).

Bits 14-8 are the volume envelope sustain level in 0.75dB increments, with 0x7f being no attenuation and 0 being silence.

Bit 7 indicates that this channel's envelope generator is to be turned off; this prevents the envelope engine from updating the channel's envelopes and LFO's, and also from writing to the channel's pitch, volume, and filter target registers.

Bits 6-0 are an encoded volume envelope's decay or release rate, with 0x7f being the mimimum time of 240 usec/dB, 0x01 being the maximum time of 470 msec/dB, and 0x00 being no decay.

Register : ENVVAL

Description : Modulation Envelope Delay [cnl]

This word register is programmed with the channel's modulation envelope delay value at the beginning of the modulation envelope. A value of 0x8000 indicates no delay; values below 0x8000 indicate increasing delay in 725 usec units.

Register : DCYSUS

Description : Modulation Envelope Sustain and Decay [cnl]

15	14 – 8	7	6 – 0
PH1	SUS	0	DCY

Bit 15 of this word register determines if the values written to this channel register are decay (0) or release (1).

Bits 14-8 are the modulation envelope sustain level in 0.75dB increments, with 0x7f being no attenuation and 0 being zero level.

Bit 7 should be written and read as zero.

Bits 6-0 are an encoded modulation envelope's decay or release rate, with 0x7f being the mimimum time of 240 usec/dB, 0x01 being the maximum time of 470 msec/dB, and 0x00 being no decay.

Register : ATKHLDV

Description : Volume Envelope Hold and Attack [cnl]

Bit 15 of this word register should be written as 0 to cause an envelope attack. Bits 14-8 are the volume envelope hold time in 92 msec increments, with 0x7f being no hold time and 0x00 being 11.68 seconds. Bit 7 should be written and read as zero. Bits 6-0 are the encoded volume envelope attack time, with 0x00 being never attack, 0x01 being 11.88 seconds, and 0x7f being 6 msec.

Register : LFO1VAL
Description : LFO #1 Delay [cnl]

This word register is programmed with the channel's LFO #1 delay value at the beginning of the note. A value of 0x8000 indicates no delay; values below 0x8000 indicate increasing delay in 725 usec units.

Register : ATKHLD
Description : Modulation Envelope Hold and Attack [cnl]

Bit 15 of this word register should be written as 0 to cause an envelope attack. Bits 14-8 are the modulation envelope hold time in 92 msec increments, with 0x7f being no hold time and 0x00 being 11.68 seconds. Bit 7 should be written and read as zero. Bits 6-0 are the encoded modulation envelope attack time, with 0x00 being never attack, 0x01 being 11.88 seconds, and 0x7f being 6 msec.

Register : LFO2VAL
Description : LFO #2 Delay [cnl]

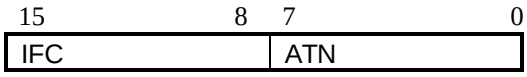
This word register is programmed with the channel's LFO #2 delay value at the beginning of the note. A value of 0x8000 indicates no delay; values below 0x8000 indicate increasing delay in 725 usec units.

Register : IP
Description : Initial Pitch [cnl]

This word register is programmed with the channel's pitch shift. A value of 0xE000 indicates no pitch shift. Values above 0xE000 are upward pitch shift, while values below are downward pitch shift. The value is encoded as octaves and fractions thereof, with 0x1000 being one octave.

Register : IFATN

Description : Initial Filter Cutoff and Attenuation [cnl]

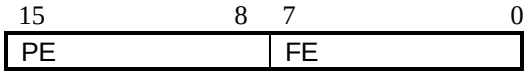


Bits 15-8 of this word register are programmed with the channel's initial filter cutoff value. The value is in quarter semitones, with 0x00 being 125 Hz and 0xFF being 8 kHz. If the Q of the channel is programmed to zero and the filter cutoff to 0xFF, the filter does not alter the signal.

Bits 7-0 of are programmed with the channel's attenuation in 0.375 dB steps, with 0x00 being no attenuation and 0xFF being 96 dB.

Register : PEFE

Description : Pitch and Filter Envelope Heights [cnl]

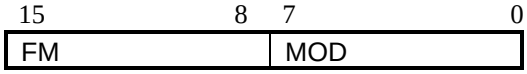


Bits 15-8 of this word register are programmed with the degree and sign with which the channel's modulation envelope affects its pitch. A value of 0x00 provides no effect, and positive values cause increasing pitch with peak deviation of +1 octave at value 0x7f and -1 octave at 0x80.

Bits 7-0 are programmed with the degree and sign with which the channel's modulation envelope affects its filter cutoff, with 0x00 providing no effect, and positive values increasing cutoff, with peak deviation of +6 octaves at 0x7f and -6 octaves at 0x80.

Register : FMMOD

Description : Vibrato and Filter Modulation from LFO #1 [cnl]



Bits 15-8 of this word register are programmed with the degree and sign with which the channel's LFO #1 provides vibrato. A value of 0x00 provides no effect, and positive values cause increasing vibrato with peak deviation of +/-1 octave at value 0x7f and -/+1 octave at 0x80.

Bits 7-0 are programmed with the degree and sign with which the channel's LFO #1 affects its filter cutoff, with 0x00 providing no effect, and positive values first increasing then decreasing cutoff, with peak deviation of +/-3 octaves at 0x7f and -/+3 octaves at 0x80.

Register : TREMFRQ

Description : LFO #1 Tremolo Amount and Frequency [cnl]

15 8	7 0
TREM	FRQ

Bits 15-8 of this word register are programmed with the degree and sign with which the channel's LFO #1 provides tremolo. A value of 0x00 provides no effect, and positive values cause increasing then decreasing amplitude with peak deviation of +/-12 dB at value 0x7f and -/+12 dB at 0x80.

Bits 7-0 are programmed to determine the channel's LFO #1 frequency in 0.042 Hz steps with 0xFF equalling 10.72 Hz.

Register : FM2FRQ2

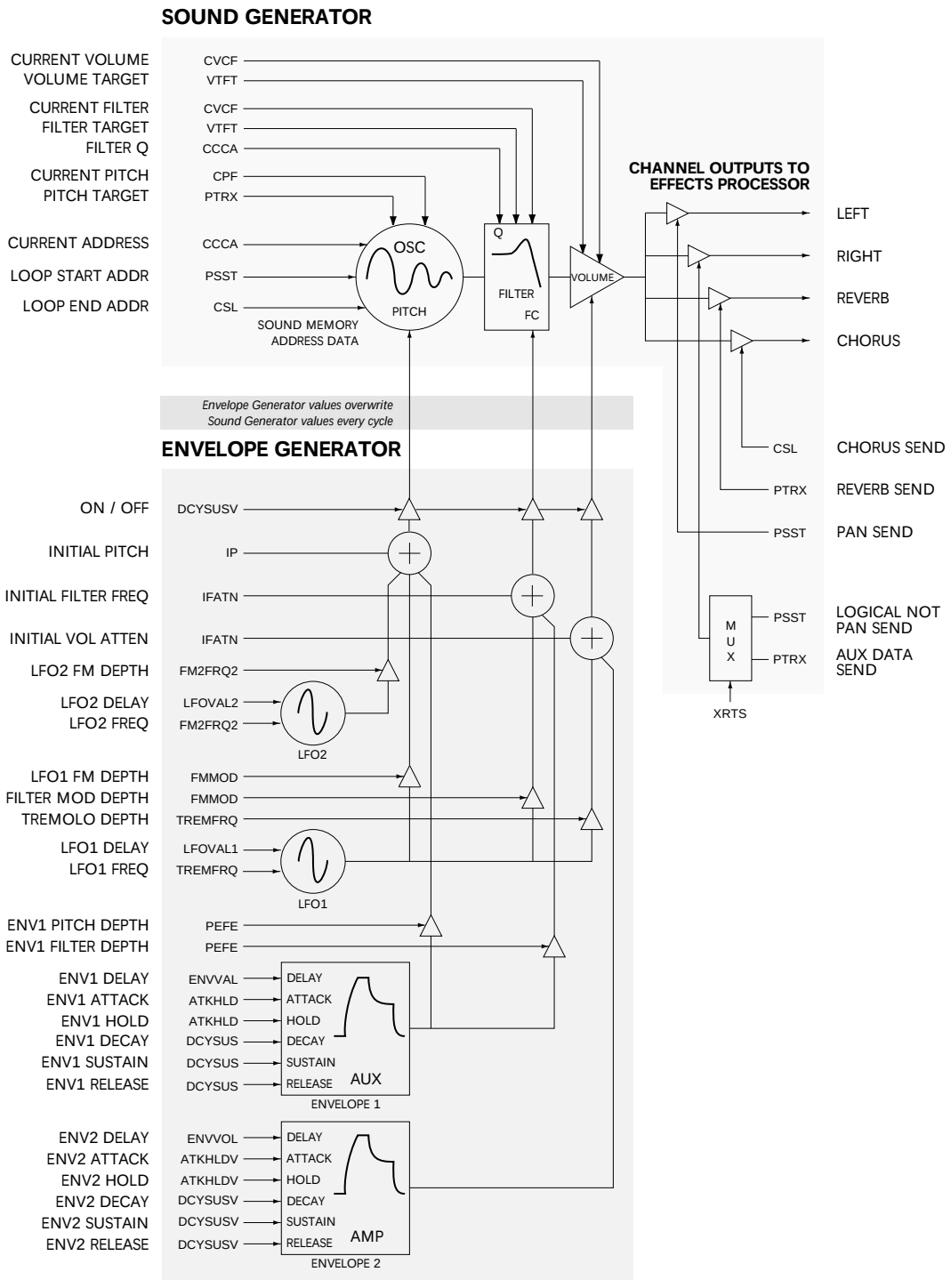
Description : LFO #2 Vibrato Amount and Frequency [cnl]

15 8	7 0
FM2	FRQ2

Bits 15-8 of this word register are programmed with the degree and sign with which the channel's LFO #2 provides vibrato. A value of 0x00 provides no effect, and positive values cause increasing then decreasing pitch with peak deviation of +/-1 octave at value 0x7f and -/+1 octave at 0x80.

Bits 7-0 are programmed to determine the channel's LFO #2 frequency in 00042 Hz steps with 0xFF equalling 10.72 Hz.

EMU8000 SIGNAL PATH DIAGRAM



4 EMU8000 Initialization

On power-up, the most EMU8000 registers contain random data. Register HWCF3 contains a bit which enables audio output which is cleared on reset. The proper procedure for initializing the EMU8000 is to:

1) Initialize the hardware configuration by writing registers HWCF1 and HWCF2.
2) Initialize the audio channels by writing default values to all channel registers.
3) Initialize the initialization arrays and the sound memory DMA address registers.
4) Enable audio by writing register HWCF3.

On power-up, register HWCF1 should be written with value 0x0059 and register HWCF2 with value 0x0020. Note that due to a VLSI error, these registers will not be correctly read by the processor, so the actual values programmed cannot be easily verified.

The channels should next be initialized. The order in which registers are initialized does matter, since the envelope engine registers can alter the sound engine registers, and the sound engine registers can alter each other.

Step 1: For all channels, set DCYSUSV to 0x0080. This turns off the envelope engine.
Step 2: For all channels, set the following registers to zero: ENVVOL, ENVVAL, DCYSUS, ATKHLDV, LFO1VAL, ATKHLD, LFO2VAL, IP, IFATN, PEFE, FMMOD, TREMFRQ, FM2FRQ2, PTRX, VTFT, PSST, CSL, CCCA. This intializes all envelope and sound engine registers except “current” registers, which must be intialized last.
Step 3: For all channels, set the following registers to zero: CPF, CVCF. This initializes the “current” registers.

The channels are now initialized, and the sound memory DMA registers can be initialized by writing zero to SMALR, SMARR, SMALW, SMARW. Next the initialization arrays are set, by the following sequence:

Step 1: Copy the first set of initialization arrays to INIT1 through INIT4.
Step 2: Wait 1024 sample periods (24 msec).
Step 3: Copy the second set of initialization arrays to INIT1 through INIT4.
Step 4: Copy the third set of initialization arrays to INIT1 through INIT4.
Step 5: Set the values of registers HWCF4, HWCF5 and HWCF6 to 0, 0x00000083, and 0x00008000.
Step 6: Copy the fourth set of initialization arrays to INIT1 through INIT4.

The arrays are now fully initialized. The output audio can now be enabled by writing 0x0004 to HWCF3.

5 Sound Memory Interface

The sound memory of the EMU8000 on the AWE32 is divided into two spaces, the sound ROM at addresses 0x000000 through 0x1FFFFF, and sound DRAM beginning at 0x200000, and potentially continuing as high as 0xFFFFDF. Sound memory from 0xFFFFE0 through 0xFFFFFF is reserved and unusable.

To transfer data to sound memory, one or more channels must be allocated to a DMA stream. The more channels allocated, the faster the transfer will take place. If multiple channels are allocated, it is best to space them evenly among the available channels.

To allocate a channel to a DMA stream (left or right):

1. Set its DCYSUSV register to 0x0080 to turn off its envelope generator.
2. Set its VTFT register to zero to force its volume to zero.
3. Set its CVCF register to zero to immediately cause its volume to become zero.
4. Set its PTRX register to 0x40000000 to cause it to step through memory one word at a time.
5. Set its CPF register to 0x40000000 to cause it to immediately become unity pitch shift.
6. Set its PSST register and its CSL register to zero to cause no loops to occur.
7. Set its CCCA register to:
   0x04000000 (left read)
   0x05000000 (right read)
   0x06000000 (left write)
   0x07000000 (right write)
   to enter DMA mode.

Once all the DMA channels you wish to allocate are complete, make sure the “empty” or “full” bit in the stream for which you have been allocating channels is clear. Then set the address register to the desired sound memory transfer address:

SMALR (left read)
SMARR (right read)
SMALW (left write)
SMARW (right write)

If you wish to do a write transfer, simply write the data words to be transfered to sequential sound memory addresses into sound memory to SMLD (left) or SMRD (right). The address will be automatically incremented. When you have completed the transfer, wait until the FULL bit in the SMAxW register goes false. You may then deallocate the channels from DMA by setting CCCA to zero, or change the SMAxW register to a new address and write data to a new location.

Note that the EMU8000 must ALWAYS loop on data. Hence if a sound is to be “single-shot” rather than looping, it should contain a loop of zero data at its end. Similarly, for perfect

reproduction of the attack, the sound should begin playing with zero data. Conventionally, a set of approximately forty sample of zero data is placed between each sound in sound memory.

If you wish to do a read transfer from sound memory, you must first read the "stale" data previously fetched from SMxD. This action actually causes the transfer of the first word in the desired sequential sound memory area into SMxD. Discard this stale data, then sequentially read the desired sound memory data from SMLD (left) or SMRD (right). The address will be automatically incremented. When you have completed the transfer, wait until the "empty" bit in the SMAxR register becomes clear before deallocating channels or changing the SMAxR register to read from a new location. This insures that the "stale" data has been read, so that data transfer will occur in the expected order.

6 Starting a Sound

Before a sound can be started, the channel must be silent and idle. This is the case when:

The DCYSUSV register has been set to 0x0080
The VTFT and CVCF registers have been zeroed.
The PTRX and CPF registers have been zeroed.

To begin programming the channel to sound:

First set the envelope engine parameters. The following are "default" values for simply playing back unarticulated audio at 44.1 kHz: ENVVOL=0x8000, ENVVAL=0x8000, DCYSUS=0x7F7F, ATKHLDV=0x7F7F, LFO1VAL=0x8000, ATKHLD=0x7F7F, LFO2VAL=0x8000, IP=0xE000, IFATN=0xFF00, PEFE=0x0000, FMMOD=0x0000, TREMFRQ=0x0010, FM2FRQ2=0x0010. Note that DCYSUSV is not programmed at this time.

Set PSST to the loop start address for the sound. For a single shot sound, the loop is in the trailing zero data. Set CSL to the loop end address for the sound. Set CCCA to the desired filter Q and audio start address of the sound. Note that all these addresses are one less than the actual audio location due to interpolator offset.

The following actions occur as close to simultaneously as possible. Set VTFT=0x0000FFFF, and then set CVCF=0x0000FFFF. Then write DCYSUSV, default value 0x7F7F. Set PTRX, default value 0x40000000. Then set CPF to 0x40000000. The note has now begun.

7 Ending a Sound

To end a sound, one can either abruptly terminate the audio (which potentially causes an audible click) or taper the sound gradually to zero, using the release time of the envelope. If the letter method is used, the channel continues to produce audio after the termination action until the envelope has decayed to zero gain.

Causing an envelope to enter "release" mode is very easy in the EMU8000. One simply programs the DCYSUSV with a value of 0x80rr, where rr represents the release rate. For example, for a release rate of 100 msec, use a value of 0x805C. To cause the modulation envelope to release, program it similarly.

Completely terminating a sound can be done by turning off the envelope engine and taking the volume to zero. This is done by writing 0x0080 to the DCYSUSV register, followed by writing 0x0000FFFF to VTFT and then 0x0000FFFF to CVCF. The audio is now terminated.

8 Modulating a Sound

Some of the EMU8000 parameters can be adjusted during the actual playback of a sound. In particular, the pitch, filtering, and volume of the sound can be changed. These parameters are often changed by means of MIDI continuous controllers.

Changing the pitch of a sound is accomplished by adjusting the IP register value. Changing the filter cutoff frequency is done by changing bits 15 through 8 of the IFATN register, and changing the volume is done by changing bits 7 through 0 of that register.

Envelope and LFO parameters can also be changed during the playback of a note causing the expected results.

Other parameters can also be changed during the playback of a note if some care is taken, and in some cases if limited audio distortion can be tolerated. The effects sends and pan can be changed, but will cause minute clicks in the sound. This distortion can be minimized if the changes are made in minimum steps.

The loop points of the sound being played can be moved to a later point in the sound if the loop end point is changed prior to changing the loop start point.

9 Initialization Arrays

The initialization arrays to be written to INIT1 through INIT4 contain parameters determining the function of the reverb, chorus and equalization effects performed by the EMU8000. The following values supply the standard default AWE32 “Hall 2” reverb and “Chorus 3” chorus programs, and with a flat equalization curve:

The initialization arrays are as follows:

First initialization set:

```
init1[32] = {  0x03ff, 0x0030, 0x07ff, 0x0130, 0x0bff, 0x0230, 0x0fff, 0x0330,
               0x13ff, 0x0430, 0x17ff, 0x0530, 0x1bff, 0x0630, 0x1fff, 0x0730,
               0x23ff, 0x0830, 0x27ff, 0x0930, 0x2bff, 0x0a30, 0x2fff, 0x0b30,
               0x33ff, 0x0c30, 0x37ff, 0x0d30, 0x3bff, 0x0e30, 0x3fff, 0x0f30};
init2[32] = {  0x43ff, 0x0030, 0x47ff, 0x0130, 0x4bff, 0x0230, 0x4fff, 0x0330,
               0x53ff, 0x0430, 0x57ff, 0x0530, 0x5bff, 0x0630, 0x5fff, 0x0730,
               0x63ff, 0x0830, 0x67ff, 0x0930, 0x6bff, 0x0a30, 0x6fff, 0x0b30,
               0x73ff, 0x0c30, 0x77ff, 0x0d30, 0x7bff, 0x0e30, 0x7fff, 0x0f30};
init3[32] = {  0x83ff, 0x0030, 0x87ff, 0x0130, 0x8bff, 0x0230, 0x8fff, 0x0330,
               0x93ff, 0x0430, 0x97ff, 0x0530, 0x9bff, 0x0630, 0x9fff, 0x0730,
               0xa3ff, 0x0830, 0xa7ff, 0x0930, 0xabff, 0x0a30, 0xafff, 0x0b30,
               0xb3ff, 0x0c30, 0xb7ff, 0x0d30, 0xbbff, 0x0e30, 0xbfff, 0x0f30};
init4[32] = {  0xc3ff, 0x0030, 0xc7ff, 0x0130, 0xcbff, 0x0230, 0xcfff, 0x0330,
               0xd3ff, 0x0430, 0xd7ff, 0x0530, 0xdbff, 0x0630, 0xdfff, 0x0730,
               0xe3ff, 0x0830, 0xe7ff, 0x0930, 0xebff, 0x0a30, 0xefff, 0x0b30,
               0xf3ff, 0x0c30, 0xf7ff, 0x0d30, 0xfbff, 0x0e30, 0xffff, 0x0f30};
```

Second initialization set:

```
init1[32] = {  0x03ff, 0x8030, 0x07ff, 0x8130, 0x0bff, 0x8230, 0x0fff, 0x8330,
               0x13ff, 0x8430, 0x17ff, 0x8530, 0x1bff, 0x8630, 0x1fff, 0x8730,
               0x23ff, 0x8830, 0x27ff, 0x8930, 0x2bff, 0x8a30, 0x2fff, 0x8b30,
               0x33ff, 0x8c30, 0x37ff, 0x8d30, 0x3bff, 0x8e30, 0x3fff, 0x8f30};
init2[32] = {  0x43ff, 0x8030, 0x47ff, 0x8130, 0x4bff, 0x8230, 0x4fff, 0x8330,
               0x53ff, 0x8430, 0x57ff, 0x8530, 0x5bff, 0x8630, 0x5fff, 0x8730,
               0x63ff, 0x8830, 0x67ff, 0x8930, 0x6bff, 0x8a30, 0x6fff, 0x8b30,
               0x73ff, 0x8c30, 0x77ff, 0x8d30, 0x7bff, 0x8e30, 0x7fff, 0x8f30};
init3[32] = {  0x83ff, 0x8030, 0x87ff, 0x8130, 0x8bff, 0x8230, 0x8fff, 0x8330,
               0x93ff, 0x8430, 0x97ff, 0x8530, 0x9bff, 0x8630, 0x9fff, 0x8730,
               0xa3ff, 0x8830, 0xa7ff, 0x8930, 0xabff, 0x8a30, 0xafff, 0x8b30,
               0xb3ff, 0x8c30, 0xb7ff, 0x8d30, 0xbbff, 0x8e30, 0xbfff, 0x8f30};
init4[32] = {  0xc3ff, 0x8030, 0xc7ff, 0x8130, 0xcbff, 0x8230, 0xcfff, 0x8330,
               0xd3ff, 0x8430, 0xd7ff, 0x8530, 0xdbff, 0x8630, 0xdfff, 0x8730,
               0xe3ff, 0x8830, 0xe7ff, 0x8930, 0xebff, 0x8a30, 0xefff, 0x8b30,
               0xf3ff, 0x8c30, 0xf7ff, 0x8d30, 0xfbff, 0x8e30, 0xffff, 0x8f30};
```

Third initialization set:

```
init1[32] = {  0x0C10, 0x8470, 0x14FE, 0xB488, 0x167F, 0xA470, 0x18E7, 0x84B5,
               0x1B6E, 0x842A, 0x1F1D, 0x852A, 0x0DA3, 0x9F7C, 0x167E, 0xF254,
               0x0000, 0x842A, 0x0001, 0x852A, 0x18E6, 0x9BAA, 0x1B6D, 0xF234,
               0x229F, 0x8429, 0x2746, 0x8529, 0x1F1C, 0x96E7, 0x229E, 0xF224};
init2[32] = {  0x0DA4, 0x8429, 0x2C29, 0x8529, 0x2745, 0x97F6, 0x2C28, 0xF254,
               0x383B, 0x8428, 0x320F, 0x8528, 0x320E, 0x9F02, 0x1341, 0xF264,
               0x3EB6, 0x8428, 0x3EB9, 0x8528, 0x383A, 0x9FA9, 0x3EB5, 0xF294,
```

```
                0x3EB7, 0x8474, 0x3EBA, 0x8575, 0x3EB8, 0xC4C3, 0x3EBB, 0xC5C3};
init3[32] = {   0x0000, 0xA404, 0x0001, 0xA504, 0x141F, 0x8671, 0x14FD, 0x8287,
                0x3EBC, 0xE610, 0x3EC8, 0x8C7B, 0x031A, 0x87E6, 0x3EC8, 0x86F7,
                0x3EC0, 0x821E, 0x3EBE, 0xD208, 0x3EBD, 0x821F, 0x3ECA, 0x8386,
                0x3EC1, 0x8C03, 0x3EC9, 0x831E, 0x3ECA, 0x8C4C, 0x3EBF, 0x8C55};
init4[32] = {   0x3EC9, 0xC208, 0x3EC4, 0xBC84, 0x3EC8, 0x8EAD, 0x3EC8, 0xD308,
                0x3EC2, 0x8F7E, 0x3ECB, 0x821E, 0x3ECB, 0xD208, 0x3EC5, 0x831F,
                0x3EC6, 0xC308, 0x3EC3, 0xB2FF, 0x3EC9, 0x8265, 0x3EC9, 0x831E,
                0x1342, 0xD308, 0x3EC7, 0xB3FF, 0x0000, 0x8365, 0x1420, 0x9570};
```

Fourth initialization set:

```
init1[32] = {   0x0C10, 0x8470, 0x14FE, 0xB488, 0x167F, 0xA470, 0x18E7, 0x84B5,
                0x1B6E, 0x842A, 0x1F1D, 0x852A, 0x0DA3, 0x0F7C, 0x167E, 0x7254,
                0x0000, 0x842A, 0x0001, 0x852A, 0x18E6, 0x0BAA, 0x1B6D, 0x7234,
                0x229F, 0x8429, 0x2746, 0x8529, 0x1F1C, 0x06E7, 0x229E, 0x7224};
init2[32] = {   0x0DA4, 0x8429, 0x2C29, 0x8529, 0x2745, 0x07F6, 0x2C28, 0x7254,
                0x383B, 0x8428, 0x320F, 0x8528, 0x320E, 0x0F02, 0x1341, 0x7264,
                0x3EB6, 0x8428, 0x3EB9, 0x8528, 0x383A, 0x0FA9, 0x3EB5, 0x7294,
                0x3EB7, 0x8474, 0x3EBA, 0x8575, 0x3EB8, 0x44C3, 0x3EBB, 0x45C3};
init3[32] = {   0x0000, 0xA404, 0x0001, 0xA504, 0x141F, 0x0671, 0x14FD, 0x0287,
                0x3EBC, 0xE610, 0x3EC8, 0x0C7B, 0x031A, 0x07E6, 0x3EC8, 0x86F7,
                0x3EC0, 0x821E, 0x3EBE, 0xD208, 0x3EBD, 0x021F, 0x3ECA, 0x0386,
                0x3EC1, 0x0C03, 0x3EC9, 0x031E, 0x3ECA, 0x8C4C, 0x3EBF, 0x0C55};
init4[32] = {   0x3EC9, 0xC208, 0x3EC4, 0xBC84, 0x3EC8, 0x0EAD, 0x3EC8, 0xD308,
                0x3EC2, 0x8F7E, 0x3ECB, 0x021E, 0x3ECB, 0xD208, 0x3EC5, 0x031F,
                0x3EC6, 0xC308, 0x3EC3, 0x32FF, 0x3EC9, 0x0265, 0x3EC9, 0x831E,
                0x1342, 0xD308, 0x3EC7, 0x33FF, 0x0000, 0x8365, 0x1420, 0x9570};
```