marian

MARC 4 MIDI Hardware Documentation

Version 1.0
Last Update 05-28-2002

Introduction

This document contains all technical information which are required to develop driver software for digital audio processing based on the technical properties of the “MARC 4 Midi“.

Memory Ranges

The MARC 4 Midi provides 4 memory ranges, which are of interest for the audio functionality of the card.

The access to these memory ranges has to be done by DWORD accesses only.

Nr.	Length (Bytes)	Area	Use	Commentary
0	128	PLX-Register	Configuration of the PCI-Bridge	
1	131072	RAM	Play and Capture Data	
2	1024	FPGA WR-Register	Control Register	All Write Only
		FPGA RD-Register	Status Register	All Read Only
3	512	FPGA Load	Loading the FPGA	Only available after: Power Up or local Reset

PLX Register (Memory Range 0, Read + Write)

This memory range provides the access to the registers of the PLX PCI-Bridge-Controllers. The memory address of a register can be calculated by: *base address of the memory range + register number*.

The registers 0x4C and 0x50 are of main interest regarding interrupt and reset control. For further information about the functionality of these registers please see the manufacturers documentation ‘plx.pdf'.

RAM (Memory Range 1, Read + Write)

The RAM occupies a memory range of 0x20000 bytes (128k) and serves for the exchange of the audio data between card and driver software.

Address	Hardware Audio Buffer
0x00000000	Analog Play Channel 1
0x00004000	Analog Play Channel 2
0x00008000	Analog Play Channel 3
0x0000C000	Analog Play Channel 4
0x00010000	Analog Record Channel 1
0x00014000	Analog Record Channel 2
0x00018000	Analog Record Channel 3
0x0001C000	Analog Record Channel 4

The hardware audio buffer of each channel has a length of 16 Kbytes and is 32 bit mono sample aligned.
Only the Bits 0..23 of any 32 Bit mono sample can be used for the audio data.
Samples are always to write and read MSB justified (highest bit = D23).

You will find sample code for audio data transfer in file „transfer.pas“.

FPGA WR Register (Memory Range 2, Write Only)

All signals are general H-active. After the FPGA initialization all registers are set to 0.

Address (Base + n)	Register
0x00	Control
0x04	Internal Clock Generator
0x08	Codec Control
0x0C	Monitor Control
0x10	MIDI Out 1
0x14	MIDI Out 2
0x18	SyncBus Start/Stop Mode

Control (Base + 0x00)

Bit	0	Analog Play	Enable	1)
	1	Analog Capture	"	
	2	Analog Play	Counter Reset	2)
	3	Analog Capture	"	
	4	Clock Generator	Range	3)
	5	"	"	
	6	-		
	7	Clear Interrupt		4)
	8	SyncBus	Clock Mode	5)
	9		"	
	10	Analog Play	Run	6)
	11	Analog Capture	"	
	12	Codec	Select	7)
	13	"	"	
	14	-		
	15	-		

1) Play/Capture Enable

Setting the respective bits causes the activation of the appropriate devices. Since the *SyncBus*–Feature is available, Play/Capture Enable will NOT start recording or playing of the devices. The device will remain in a kind of „standby“ mode and gets started only if the Run-Bit is set or a start is signaled by the SyncBus (when in SyncBus-Slave-Mode.)

Resetting the respective bits disables the appropriate devices. A running recording or playback stops even if there is no stop signaled via „Run“ or SyncBus.

„Enable“ and „Run“/“Start of the SyncBus“ are always logical AND combined.

For sample code see the file „control.pas“ under „_EnablePlay“, „_EnableRec“.

2) Counter Reset

To set the sample Counter of the respective devices to zero, the appropriate bit has to be set.

The counter receives the value 0 and will be blocked until the appropriate bit is set again.

For sample code see the file „control.pas“ under „_Reset“.

3) Clock Generator Range

Depending on the required sample rate, first the appropriate operating range has to be set for the internal clock generator.

Bit 1	Bit 0	
1	1	6..14 kHz
1	0	12..29 kHz
0	1	24..58 kHz
0	0	47..115 kHz

For sample code see the file „control.pas“ under „SetDcoRate“.

4) Clear Interrupt

After an interrupt occurred, your interrupt service routine has to clear it on the card. It will be cleared after setting and subsequently resetting this bit. For sample code see the file „isr.asm“

5) SyncBus Clock Mode

The SyncBus ensures that several *MARIAN SyncBus* compatible audio cards run synchronously. These Bits allow the configuration of the cards clock behavior towards the SyncBus.

The software must ensure, that while operating several cards parallel, only one of them outputs a clock to the SyncBus. If there is at least one card configured as clock slave, one other card has to be configured as clock master.

Bit 1	Bit 0	
0	0	Autonomous (SyncBus clock independently)
0	1	Slave (clock is determined by the SyncBus clock)
1	1	Master (internal clock is the master clock for the SyncBus)

6) Run

These bits together with the Enable-Bit start the recording or playback of a device. Within the SyncBus-Master-Mode the start signal will be generated on the SyncBus, in such a way as to enable the pending slave devices with active Enable-Bit to start.

For sample code see the data „control.pas“ under „_Start“.

7) Codec Select

Before writing any control sequences to the codecs, the destination codec(s) has to be determined. That’s the task of these bits.

Bit 3	Bit 2	Bit 1	Bit 0	
0	0	0	1	Codec 1 (Channels 1-2)
0	0	1	0	Codec 2 (Channels 3-4)

Internal Clock Generator (Base + 0x08)

These settings determine the frequency of the internal clock generator. This generator produces the master clock for the given sample rate.

Bit 0..15 Master Frequency Value

This values has to be calculated:

Master Frequency Value = Sample Rate * 0.0524288

Additionally the appropriate operating range has to be set for internal clock generator. (see Clock Generator Range).

For sample code see the data „control.pas“ under „SetDcoRate“.

Codec Control (Base +0x08)

By use of this control register you have access to the internal registers of the codecs (AK4524). Writing into this register initiates a serialization and transmission process of the given value for a given register. The control sequence consists of a dword with the following structure:

Bit	0..7	Value for Codec Register
	8..11	Codec Register Number
	12	0
	13	1
	14	0
	15	1
	16..31	0

The destination codec(s) will be determined by the Codec Select Bits.
Between two writes accesses a pause of minimum 4µs must be ensured.
The documentation of all codec registers can be found in the manufacturers documentation „4524.pdf“.

Monitor Control (Base + 0x0C)

This register determines the signal source for the audio outputs.

Bit	0..1	Output 1-2	Input select	1)
	2	Output 1-2	Playback/Monitoring	2)
	3..4	Output 3-4	Input select	
	5	Output 3-4	Playback/Monitoring	

1) Input Select

These bits determine the input channel pair, which will be monitored at the appropriate output in case of monitoring is active.

Bit	0 1	
	0 0	Input 1-2
	0 1	Input 3-4

2) Playback/Monitoring

Setting this bit causes the appropriate output to monitor the chosen input channel pair. Otherwise, the playback signal can be heard at the output.

Midi Out 1 and Midi Out 2 (Base + 0x10 and Base + 0x14)

These two registers are the out ports for midi interface 1 (Base+0x10) and for midi interface 2 (Base+0x14).

Bit 7..0 Midi out data

All midi out data are buffered by the hardware through a FIFO. Before a byte is written to an out register, the state of the FIFO buffer should be checked.

SyncBus Start/Stop Mode

The SyncBus ensures that several *MARIAN SyncBus* compatible audio cards run synchronously. Through this register, the devices start/stop behavior towards the SyncBus can be configured. For further information about the configuration of the cards behavior towards the SyncBus please see SyncBus Clock/Mode.

Bit	0	Play Slave
	1	Record Slave
	2	Play Master
	3	Record Master

If a device is configured as Slave and the Enable-Bit is set, it will start only if the SyncBus detects a start signal.

If a device is configured as Master, the setting of the appropriate Run-Bits will lead to a start signal at the SyncBus.

A device has to be configured as Master and Slave, if you want to generate a SyncBus start signal and start the device simultaneously.

If a device is configured neither as Master nor as Slave, it will be autonomous. For start and stop only the bits Enable and Run will be used.

FPGA RD Register (Memory Range 2, Read Only)

Address (Base + n)	Register
0x00	Counter
0x04	Status
0x08	MIDI In 1
0x0C	MIDI In 2

Counter (Base + 0x00)

By reading these counters you can get the current playback respectively record positions in relation to the base memory address of the respective playback respectively record device.

Bit	9..0	Analog Play Counter
	19..10	Analog Capture Counter

The counters represent the upper 10 bit of the internal 12 bit wide dword-counter. One step of this internal counter represents a progress of one 32 bit mono sample. One step of the counters in this register represent a progress of four 32 Bit mono samples.

Status (Base + 0x08)

Bit	0	Midi 1 Out Buffer Full	1)
	1	-	
	2	Midi 2 Out Buffer Full	1)
	3	-	
	4	Midi In 1 Buffer Overflow	2)
	5	Midi In 1 Data Available	3)
	6	Midi In 2 Buffer Overflow	2)
	7	Midi In 2 Data Available	3)

1) Midi Out Buffer Full

This bit signals whether or not the output buffer is full. If it is set, then no midi output data should be written to the appropriate midi output register. This bit should be read twice before assuming that it is valid.

2) Midi In Buffer Overlow

This bit signals whether or not the midi input buffer has an overflow. If this bit is set, then it should be handled as a midi input error. This occurs, if the software is not able to read the midi input register as fast as the midi data come in. This bit becomes inactive if the next byte is read from the appropriate input buffer.

3) Midi In Data Available

This bit is set, if the midi input buffer is not empty. That means for the driver software, that midi in data can be read from the appropriate midi input register.

MIDI In 1 and Midi In 2 (Base + 0x08 und Base + 0x0C)

By reading these registers, the midi input data can be received from the midi interface 1 (Base+0x08) or midi interface 2 (Base+0x0C). All midi in data are buffered by the hardware through a FIFO. The state of this FIFO should be considered before reading these registers and to recognize an error situation.

FPGA Load (Memory Range 3, after Power Up or Local Reset)

To give the MARC 4 Midi a functionality besides PnP, you have to load a control design into the FPGA. The control design is stored in so-called BIT-Data files. The loading of a control design happens through a bit-serial data stream at D0 of the FPGA Load address. You will find the control design for the “Marc 4 Midi” in the file „marc4m.bit“.

For sample code to FPGA-Loading see the file “init.pas”

Initializing the Hardware

Subsequently you will find the steps to initialize the hardware successfully. Afterwards the card will be ready for recording and playback. For sample code see the file „init.pas“.

1. Initialize the PCI Controller (see „InitPciController“)
2. Release the reset of the hardware, initialize the FPGA (see „WriteConfigData“)
3. Verify the success of the initialization (see „FpgaInitialized“)
4. Set the Internal Clock Generator to a sample rate (i.e. 44100Hz)
5. Initialize the analog codecs (see „InitADDA“). A description of all codec registers contains the manufacturer documentation „4524.pdf“.
6. Set the Output Routing for the devices. The software should provide the possibility to determine the Output Routing. After restarting the system the software should restore these settings
7. Define the volume settings for the devices (see „volume.pas“). The software should provide the possibility to define the volume settings. After restarting the system the software should restore these settings.
8. Define the SyncBus-Clock and SyncBus-Start/Stop settings. The software should provide the possibility to define these settings. After restarting the system the software should restore these settings.

Contact the Developers

Should you have any problems or questions please do not hesitate to contact:

marian

Elektronik & Software Entwicklung

Eisenacher Strasse 72
04155 Leipzig

Email: developers@marian.de