POMMAX2 — Programing Interface rev. 0.0

Krzysztof Mazur

1 Introduction

This document describes the programing interface of POMMAX2 analog inputs controllers.

The POMMAX2 interface supports up to 2 ADCs. The number of channels per ADC is not limited by the interface. Samples from ADCs are accessible from the host by two simple ring buffers. The POMMAX2 interface also allows for simple communication interface with ADCs.

The POMMAX2 programming interface is designed to use with PCI-compatible buses, including PCI, PCI-X, PCI Express, PCI/104 and ARBus.

1.1 New revisions

This is the revision 0.0 of this specification. The first number is the major revision number. The major revision number identifies the hardware interface version. Two documentations with the same major revision define the same hardware interface. The second number is the minor revision. This number identifies the version of the documentation of the same hardware interface.

Latest revision of this document is available at `http://www.microster.pl/doc/pommax2`.

1.2 Copyright

2 Configuration Registers

The POMMAX2 controller have 256 bytes of the PCI-compatible Configuration Address Space (see Figure 1). Access for this address space is usually provided by the bus interface driver in the operating system. If not, see the bus interface specification for information about accessing the Configuration Address Space on your system.

2.1 Vendor ID

Offset	0x00
Width	16 bit
Type	RO
Reset value	0xff00

The Vendor ID identifies the manufacturer of the device. For this device this register is equal to 0xff00.

2.2 Device ID

Offset	0x02
Width	16 bit
Type	RO
Reset value	0x0003

The Device ID identifies the device model. For this device this register is equal to 0x0003.

[1] This signature at 0xf0 is used only devices using ARBus bus interface. For devices using other interfaces this area is reserved.

Offset	3	2	1	0
0x00	Device ID (0x0003)		Vendor ID (0xff00)	
0x04	Status		Command	
0x08	Base Class (0x11)	Sub-class (0x80)	ProgIF (0x00)	Revision ID
0x0c	Reserved	Header Type	Reserved	Reserved
0x10	Base Address Register 0			
0x14	Base Address Register 1			
0x18	Base Address Register 2			
0x1c–0x2b	Reserved			
0x2c	Subsystem Device ID		Subsystem Vendor ID	
0x30–0xef	Reserved			
0xf0[1]	0x53	0x42	0x52	0x41
0xf4-0xff	Reserved			

Figure 1: The POMMAX2 Configuration Address Space.

2.3 Command

Offset	0x04
Width	16 bit
Type	RW
Reset value	0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESV														MEM	IO
RO														RW	RO

Command Register Bit Descriptions

Bit	Name	Description
15–2	RESV	**Reserved**.
1	MEM	**Memory Space**. Set to enable decoding of Memory Regions
0	IO	**IO Space**. Unused, always equal to zero. Memory Space

2.4 Status

Offset	0x06
Width	16 bit
Type	RO
Reset value	0x0000

The Status register is always equal to 0.

2.5 Revision ID

Offset	0x08
Width	8 bit
Type	RO
Reset value	Implementation specific

The Revision ID identifies the revision of the device.

2.6 ProgIF

Offset	0x09
Width	8 bit
Type	RO
Reset value	0x00

The ProgIF register identifies the programming interface in the specified sub-class of devices. This device reports the programming interface as 0x00.

2.7 Sub-class

Offset	0x0a
Width	8 bit
Type	RO
Reset value	0x80

The sub-class register identifies the sub-class of the device. This device uses class 0x11 and sub-class 0x80 — Signal Processing Controller.

2.8 Base Class

Offset	0x0b
Width	8 bit
Type	RO
Reset value	0x11

The Base Class register identifies the class of the device. This device uses class 0x11 — Signal Processing Controller.

2.9 Header Type

Offset	0x0e
Width	8 bit
Type	RO
Reset value	0x00 or 0x80

The Header Type identifies the type of the Configuration Space header. This field is equal to 0x00 in single function devices or 0x80 in multifunction devices.

2.10 Base Address Register 0

Note: This subsection describes a 32-bit Base Address Register. Some implementations may support 64-bit Base Address Registers and such register will occupy two words. See PCI Specification for detailed description of Base Address Registers.

Offset	0x10
Width	32 bit
Type	RW
Reset value	implementation specific

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
RW															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR												P	TYPE		IO
RW				RO								RO	RO		RO

Base Address Register 0 Bit Descriptions

Bit	Name	Description
31–4	ADDR	**Address**. The ADDR sets the bits 31–4 of the region 0 base address. Note that on some buses like ARBus the address may be limited and higher bits are always cleared. Bits 11-4 are always cleared to indicate 4096 byte region
3	P	**Prefetchable**. implementation specific
2-1	TYPE	**Type**. implementation specific, 0 (32-bit base address) is assumed in this documentation
0	IO	**IO Space indicator**. cleared to indicate Memory Space

2.11 Base Address Register 1

Note: This subsection describes a 32-bit Base Address Register. Some implementations may support 64-bit Base Address Registers and such register will occupy two words. See PCI Specification for detailed description of Base Address Registers.

Offset	0x14
Width	32 bit
Type	RW
Reset value	implementation specific

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
RW															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR												P	TYPE		IO
RW								RO				RO	RO		RO

Base Address Register 1 Bit Descriptions

Bit	Name	Description
31–4	ADDR	**Address**. The ADDR sets the bits 31–4 of the region 1 base address. Note that on some buses like ARBus the address may be limited and higher bits are always cleared. Bits 7-4 are always cleared to indicate 256 byte region
3	P	**Prefetchable**. cleared to indicate non-prefetchable region
2-1	TYPE	**Type**. implementation specific, 0 (32-bit base address) is assumed in this documentation
0	IO	**IO Space indicator**. cleared to indicate Memory Space

2.12 Base Address Register 2

Note: This subsection describes a 32-bit Base Address Register. Some implementations may support 64-bit Base Address Registers and such register will occupy two words. See PCI Specification for detailed description of Base Address Registers.

Note: Implementation of region 2 is optional. If implementation does not support region 2 this registers is hard-wired to zero.

Offset	0x18
Width	32 bit
Type	RW
Reset value	implementation specific

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
RW															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR												P	TYPE		IO
RW								RO				RO	RO		RO

Base Address Register 2 Bit Descriptions

Bit	Name	Description
31–4	ADDR	**Address**. The ADDR sets the bits 31–4 of the region 2 base address. Note that on some buses like ARBus the address may be limited and higher bits are always cleared. Bits 8-4 are always cleared to indicate 512 byte region
3	P	**Prefetchable**. implementation specific
2-1	TYPE	**Type**. implementation specific 0 (32-bit base address) is assumed in this documentation
0	IO	**IO Space indicator**. cleared to indicate Memory Space

2.13 Subsystem Vendor ID

Offset	0x2c
Width	16 bit
Type	RO
Reset value	Subsystem specific

The Subsystem Vendor ID is assigned by expansion board or subsystem vendor.

2.14 Subsystem ID

Offset	0x2e
Width	16 bit
Type	RO
Reset value	Subsystem specific

The Subsystem ID is assigned by expansion board or subsystem vendor.

3 Region 0 — Ring buffers

The POMMAX2 uses two sample ring buffers, one per ADC. Both buffers are visible as Memory Region 0. The buffer for ADC0 starts at offset 0. The buffer for ADC1 starts at offset equal to BAR0 size divided by 2.

Region 0 map for 4 KiB BAR0

Offset	Register	Description
0x000	ADC0 ring buffer	3.1
0x800	ADC1 ring buffer	3.1

3.1 Ring buffer

Ring buffers contains interleaved 16-bit little-endian signed PCM data. The Fig. 2 shows how data is stored in ring buffers. $c_i(j)$ represents the j-th sample of i-th channel. The reliable access for the ring buffer must be properly synchronized with writes to buffer. The current write pointer can be read from ADC Write Pointer Register (see 4.2.1). The value returned by read from sample that is currently being written is undefined.

Range	0–1	2–3	4–5	6–7	8–9	10–11	12–13	14–15
0x000	$c_0(0)$	$c_1(0)$	$c_2(0)$	$c_3(0)$	$c_4(0)$	$c_5(0)$	$c_6(0)$	$c_7(0)$
0x010	$c_0(1)$	$c_1(1)$	$c_2(1)$	$c_3(1)$	$c_4(1)$	$c_5(1)$	$c_6(1)$	$c_7(1)$
0x020	$c_0(2)$	$c_1(2)$	$c_2(2)$	$c_3(2)$	$c_4(2)$	$c_5(2)$	$c_6(2)$	$c_7(2)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0x7e0	$c_0(126)$	$c_1(126)$	$c_2(126)$	$c_3(126)$	$c_4(126)$	$c_5(126)$	$c_6(126)$	$c_7(126)$
0x7f0	$c_0(127)$	$c_1(127)$	$c_2(127)$	$c_3(127)$	$c_4(127)$	$c_5(127)$	$c_6(127)$	$c_7(127)$

Figure 2: Ring buffer format 8 channels per ADC for card with two 2 KiB buffers.

4 Region 1 — Runtime registers

Runtime Register Summary

Offset	Type	Register	Reset value	Description
0x00	—	Global Registers	—	4.1
0x80	—	ADC0 Registers	—	4.2
0xc0	—	ADC1 Registers	—	4.2

4.1 Global registers

Global Register Summary

Offset	Type	Register	Reset value	Description
0x00	RW	ADC Reset	0x00	4.1.1

4.1.1 ADC Reset

Offset	0x00
Width	8 bit
Type	RW
Reset value	0x00

7	6	5	4	3	2	1	0
RESV						ADC1_RST	ADC0_RST
RO						RW	RW

ADC Reset Bit Descriptions

Bit	Name	Description
7–2	RESV	**Reserved**.
1	ADC1_RST	**ADC1 Reset**. Setting this bit to 1 resets the ADC1. The ADC1 stays in reset state as long this bit is equal to 1. For normal reset this bit should be cleared by user after at least 1 μs.
0	ADC0_RST	**ADC0 Reset**. Setting this bit to 1 resets the ADC0. The ADC0 stays in reset state as long this bit is equal to 1. For normal reset this bit should be cleared by user after at least 1 μs.

4.2 ADC Runtime registers

Runtime Register Summary

Offset	Type	Register	Reset value	Description
0x00	RO	ADC Write Pointer (ADC_PTR)	0x00000000	4.2.1
0x08	RO	ADC Command Status (ADC_CSTAT)	0x00	4.2.2
0x10	RO	ADC Command RX Buffer (ADC_RX)	0x00	4.2.5
0x20	WO	ADC Command Control (ADC_CCTRL)	0x00	4.2.5
0x30	WO	ADC Command TX Buffer (ADC_TX)	0x00	4.2.5

4.2.1 ADC Write Pointer (ADC_PTR)

ADC runtime offset	0x00
Width	32 bit
Type	RO
Reset value	0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTR															
RO															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTR															
RO															

ADC Write Pointer (ADC_PTR) Bit Descriptions

Bit	Name	Description
31-0	PTR	**ADC Write Pointer**. This register holds the pointer to the sample that it's currently being written. The pointer is in samples (not bytes). For cards with 8 channels per ADC with 16-bit resolution, sample size is equal to 16 bytes. The card is allowed to hard-wire to 0 any number of most significant bits of this register. It's guaranteed that reading 32-bit value from this register is atomic (see section A for details).

4.2.2 ADC Command Status (ADC_CSTAT)

ADC runtime offset	0x08
Width	8 bit
Type	RO
Reset value	0x00000000

7	6	5	4	3	2	1	0
RESV					SEQ	XMIT	PENDING
RO					RO	RO	RO

ADC Command Status (ADC_CSTAT) Bit Descriptions

Bit	Name	Description
7-3	RESV	**Reserved**.
2	SEQ	**Sequence Number**. The received response sequence number. This bit is toggled every message is received from ADC.
1	XMIT	**Command transmission in progress**. This bit is equal to 1 only if the transmission of command is in progress.
0	PENDING	**Command transmission pending**. This bit is equal to 1 only if the command transmission is started by user by setting of **Start Command** bit but command transmission was not started yet.

4.2.3 ADC Command Receive buffer (ADC_RX)

ADC runtime offset	0x10
Width	16 bytes
Type	RO

The ADC Command Receive buffer contains message received from ADC.

4.2.4 ADC Command Control (ADC_CCTRL)

ADC runtime offset	0x20
Width	8 bit
Type	RW
Reset value	0x00

7	6	5	4	3	2	1	0
RESV				XMIT_SET	XMIT_CLEAR	RESV	START
WO				WO	WO	WO	WO

ADC Command Control (ADC_CCTRL) Bit Descriptions

Bit	Name	Description
7-4	RESV	**Reserved**.
3	XMIT_SET	**Xmit Set**. Writing 1 to this bit sets the **Command transmission in progress** bit and immediately starts transmission of command.
2	XMIT_CLEAR	**Xmit Clear**. Writing 0 to this bit clears the **Command transmission in progress** bit and immediately stops transmission of command.
1	RESV	**Reserved**.
0	START	**Start Command**. Writing 1 to this bit sets the **Command transmission pending** bit and schedules of start transmission of command. The command will be started after receiving of synchronization from ADC.

4.2.5 ADC Command Transmit buffer (ADC_TX)

ADC runtime offset	0x30
Width	16 bytes
Type	WO

The ADC Command Transmit buffer contains message that will be sent to ADC. The content of this register may be destroyed during transmission.

5 Region 2 — POMMAX-compatible Sample buffer

The POMMAX2 cards can optionally add support for POMMAX-compatible interface. The POMMAX cards use single 512 byte memory region. This region is exported as Memory Region 2. If implementation does not support region 2 the Base Address Register 2 is hard-wired to zero.

In POMMAX card ADC samples are stored in separate per-channel 8-entry FIFOs (Fig. 3). Each FIFO occupy 16 consecutive bytes. The mapping between POMMAX2 c_j channels and POMMAX channels p_j is implementation-specific.

Range	0–1	2–3	4–5	6–7	8–9	10–11	12–13	14–15
0x00	$p_0(i)$	$p_0(i-1)$	$p_0(i-2)$	$p_0(i-3)$	$p_0(i-4)$	$p_0(i-5)$	$p_0(i-6)$	$p_0(i-7)$
0x10	$p_1(i)$	$p_1(i-1)$	$p_1(i-2)$	$p_1(i-3)$	$p_1(i-4)$	$p_1(i-5)$	$p_1(i-6)$	$p_1(i-7)$
0x20	$p_2(i)$	$p_2(i-1)$	$p_2(i-2)$	$p_2(i-3)$	$p_2(i-4)$	$p_2(i-5)$	$p_2(i-6)$	$p_2(i-7)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0x1e0	$p_{30}(i)$	$p_{30}(i-1)$	$p_{30}(i-2)$	$p_{30}(i-3)$	$p_{30}(i-4)$	$p_{30}(i-5)$	$p_{30}(i-6)$	$p_{30}(i-7)$
0x1f0	$p_{31}(i)$	$p_{31}(i-1)$	$p_{31}(i-2)$	$p_{31}(i-3)$	$p_{31}(i-4)$	$p_{31}(i-5)$	$p_{31}(i-6)$	$p_{31}(i-7)$

Figure 3: POMMAX memory map.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	RESV			VALUE											
RO	RO			RO											

Each 12-bit sample is stored as 12 least significant bits of 16-bit word. The most significant bit is used to indicate that sample is currently being written and the result may be inconsistent. In POMMAX2 this bit is however always cleared and the hardware is required to implement 16-bit atomic reads (see section A for details).

Sample format in POMMAX-compatibility sample buffer Bit Descriptions

Bit	Name	Description
15	`BUSY`	**Busy**. The sample is currently being updated. The `VALUE` may be inconsistent.
14-12	`RESV`	**Reserved**.
11–0	`VALUE`	**Value**. This field contains measured value.

A Emulation of multibyte atomic reads

The POMMAX2 interface requires that reads of some 16-bit or 32-bit registers are atomic:

- ADC Write Pointer (`ADC_PTR`),
- POMMAX-compatible buffer samples.

If the bus interface does not support such atomic reads proper emulation must be implemented in hardware. The implementation details depends on bus interface. For instance on 8-bit bus where the 16-bit read of address $2k$, where k is integer value, is implemented by reading byte $2k$ followed by reading byte $2k + 1$ the hardware must guarantee that the value ok $2k + 1$ will be latched during first read and the second read will return latched value instead of current value which can be modified.