Implementing Legacy Audio on the PCI Bus

Revision 1.2

Written by

Gary Solomon
Sr. Staff Engineer
Platform Architecture Lab
gary_solomon@ccm.jf.intel.com

Intel Corporation

1. Introduction / Scope

This paper is targeted at IHVs and OEMs who have detailed working knowledge of the current PC audio architecture. It is also recommended that the reader be familiar with the Audio Codec '97 Component ("AC '97") Specification available on the Intel Web server at http://www.intel.com/pc-supp/platform/ac97/.

Companion white papers address two related subjects:

- "Digital audio" and the 1997 desktop PC
- Hardware Acceleration Models and Re-direction of Audio Streams

The scope of this document is not intended to fully detail each mechanism described herein. It is also not the intent of this paper to debate or judge whether or not it makes sense to migrate legacy compatible audio to the PCI bus. This paper is intended to provide guidance for those audio vendors who have chosen to do so. The reader will first be introduced to the technical issues that need to be resolved. Key design and support considerations for each possible solution will then be explored. In its conclusion this paper will provide a series of recommendations on how to implement legacy compatible PCI-based audio for 1997.

System audio is migrating to the PCI bus in the volume PC platform market segment. Lower BOM (Bill-of-Materials) cost due to higher levels of feature integration, coupled with the ability to make more efficient use of new found headroom provided by the PCI bus are two good reasons why. The Audio Codec '97 audio architecture was designed to comprehend and embrace this migration.

However, the issue of backwards compatibility with the multitude of DOS games presents a significant design challenge for PCI bus based AC'97 audio controllers in 1997. In a fully backwards compatible system, a software title that communicates directly with "legacy" I/O peripherals such as the 8237 DMAC registers, Sound Blaster* registers, and FM Synthesis registers need be able to find these registers, and their underlying functionality intact without requiring any additional software besides the legacy software itself.

PCI agents on the primary PCI bus segment always get first right of refusal in claiming cycles that emerge on the PCI bus before they are bridged to the ISA bus[1]. Therefore a PCI audio agent can easily be designed to claim the legacy Sound Blaster compatible I/O references. Issues relating to the combination of "Plug and Play" and Sound Blaster compatibility do however exist. These issues will be explored in a later section of this document.

Legacy (8237) DMA controller I/O references, along with the underlying functionality, present significant issues as well. Through no accident, the PCI bus specification does not provide for 8237 DMA functionality, but rather supports generic PCI bus mastering with robust plug and play capabilities.

[1] Assuming a fully subtractive decoding ISA bridge.

1.1. Related Documents

For more detailed descriptions of the hardware mechanisms introduced within this document, the reader is referred to the following supporting documents:

1. *PCI Local Bus Specification Revision 2.1* (available from the PCI SIG)
2. *PC/PCI DMA Arbitration and Protocols* (available at ftp://ftp.intel.com/pub/IAL/PCI/dma22.doc)
3. *Distributed DMA Support for PCI Systems*
4. *Serialized IRQ Support for PCI Systems*

2. Legacy DMA Support for the PCI Bus

In response to the OEM market requirements feedback for 1997, which continues to show a demand for legacy compatibility, two system specific solutions for extending 8237 compatible DMA functionality out onto the PCI bus have been devised. These are the mechanisms defined, implemented, and validated by Intel Corp. known as "PC/PCI", and a second, newly defined mechanism known as Distributed DMA*, or "DDMA*".

For practical reasons, which will be described later in this document, it is strongly recommended that both of these techniques limit their implementations to motherboard integrated PCI agents (no PCI add-in cards).

2.1. PC/PCI

PC/PCI is a mechanism that was defined and developed by Intel's Mobile/Handheld Products Group (MHPG) as a mobile docking solution which allows ISA slots to exist in docking stations connected to the notebook's PCI bus. This scheme is now being applied to the desktop PC as well.

By providing a new arbitration construct, consisting of a serialization protocol for encoding and decoding DMA requests/grants, a request/grant pair, distinct from the PCI bus pair, is used to bundle requests for any combination of 8237 supported DMA channel(s) for each device needing DMA support. This encoded mapping on the PC/PCI agent's request/grant pair provides the pathway that enables a PCI resident agent to deliver 8237 style DMA requests to the system without requiring separate and distinct DREQ/DACK# pins for each DMA channel that is used by the PC/PCI agent.

Some future Intel products are currently expected to support PC/PCI functionality[2].

There are a couple of reasons why PC/PCI legacy agents be deployed as motherboard devices. One reason is that the PC/PCI DMA request serialization decoder is not a function of the standard PCI bus arbiter, and therefore there is no guaranteed connection between the PCI expansion slots and the serialization logic. Also AT* compatible interrupt requests (IRQs) are still provided directly to an 8259 compatible interrupt controller via sideband connection from the PCI device. These sideband signals are not supported in the PCI slot definition.

[2] Please address product specific inquiries to an Intel Sales Office near you directly.

PC/PCI features include:

1. Software transparent; no TSRs, VXDs, etc..
2. Does not require the PCI agent which implements it to be a PCI master, thus lowering the cost and complexity of an implementation on the part of peripheral vendors.
3. Technical support can be provided by a single source.

2.2. Distributed DMA (DDMA)

DDMA is an alternative legacy DMA solution. This technique effectively "distributes" the 8237 DMA controller into physically separate PCI resident components on a per DMA channel basis.

A "master DMA agent" central resource is located somewhere in the system which orchestrates the legacy I/O traffic to and from the distributed DMA elements of the system. The DMA master intercepts all 8237 I/O register references and redirects, via a software configured lookup table, these accesses to distributed agents throughout the system. Since the 8237 DMAC programming model, for control/status registers, combines data for all of its DMA channels within single I/O registers, the DMA master typically scatters and gathers several bus transactions to complete a single legacy I/O register read or write.

DDMA, as for PC/PCI, is also suggested to be a motherboard device solution. While the bus mastership interface could well be handled by a standard PCI slot, there is no guaranteed means for connection via the standard PCI slot of the legacy interrupt request signal lines. DDMA devices most likely will appear with a new serialized interrupt request scheme which calls for serially encoding IRQ information onto PCI bus standard PIRQ(A:D) signal lines (similar to encoding for PC/PCI DMA requests). However, PCI slot add-in DDMA agents could exhibit interoperability issues, since the connection of its PCI slot interrupt request signal to a serialization protocol capable receiver on the motherboard is not guaranteed.

Additionally, when considering DDMA PCI slot add-ins and their associated drivers, the determination and management of DMA channels that have been assigned to specific DDMA agents vs. standard ISA DMA assignments vs. unassigned DMA channels becomes more complex[3].

DDMA features include:

1. "Flyby" DMA transfers
2. Extended, non-legacy DMA support

3. Sound Blaster Compatibility on the Primary PCI Bus

As mentioned briefly in the introductory section of this paper, PCI agents on the primary PCI bus segment get first right of refusal on all PCI bus cycles before they are forwarded to the ISA bus, provided that the PCI to ISA bridge is a fully subtractive decoder. This is a key enabler for PCI devices that wish to integrate legacy audio.

While it is technically feasible to incorporate full Sound Blaster compatible legacy I/O mapping and the underlying functionality, the issue of Plug and Play needs to be addressed.

Additionally, the issues of I/O address aliasing, and the prevention of audio driver replacement need to be addressed.

[3] Complexity associated with the coordination of multiple pieces of software each of which is coming from a different source, i.e., BIOS, ISA Plug and Play, DDMA device drivers.

It is important to note that while the primary PCI bus segment has the necessary "flow through" quality which allows for cycles destined for the ISA bus to be intercepted along the way, other PCI segments or PCI-like segments such as AGP (Accelerated Graphics Port) do not have this quality. This limits the implementation of PCI based legacy audio to the primary bus segment of the system.

3.1. PCI bus Legacy Compatibility and ISA Plug and Play

Transplanting Sound Blaster compatible audio to the PCI bus means moving Sound Blaster's associated ISA bus resources along with it[4]. These ISA resources are currently managed by ISA Plug and Play algorithms.

The issue at hand is deciding what to do with respect to the ISA Plug and Play paradigm as these ISA resources move into a gray area situated somewhere between the ISA bus and the PCI bus.

The ISA Plug and Play objectives do not align in any way with the objective of maintaining legacy compatibility. In fact, they are diametrically opposed objectives. On the one hand there is ISA Plug and Play which requires the relocatibility of I/O resources. Yet on the other hand there are legacy functions that will not work correctly if their system resources are not located exactly where the legacy software title expects to find them.

Therefore, since hardware legacy compatibility is a requirement for those companies who have chosen to implement it, and Plug and Play relocatibility compromises these legacy features, the optimum solution should both maintain legacy compatibility while minimizing the design complexity and overhead otherwise needed to support the problematic reconfiguration of these features.

3.2. Legacy I/O address Aliasing

Another aspect of legacy compatibility relates to the issue of I/O address aliasing. The original IBM PC and IBM PC AT designs defined ISA add-in I/O space to be limited to 10 bits of addressing. This has become the de facto industry standard for legacy software compatibility.

Since ISA bus components used to incorporate the bus interface with 10 bits of address decode for I/O references, any alias of an incorporated base 10-bit address would also be claimed by this same component.

Recently there has been an industry initiative to eliminate this aliasing by requiring I/O decode of the full 16-bits of address on the ISA bus. While this would meet the initiative's objective of freeing these aliased addresses for subsequent I/O resource assignment to other *non-legacy* functions, it creates some degree of software compatibility exposure for the device. I/O cycles that used to get through as 10-bit aliases would now be filtered out by the full 16-bit decode. The difference in the cost of implementing full 16-bit vs. 10-bit decode is very small and the recommendation is that the component be capable of running in either mode of operation.

[4] ISA DMA channels, Interrupt requests and I/O space for example.

3.3. Audio Driver Replacement

If the audio vendor has included performance enhancements or other functions in the driver that are not supported by the set of 32-bit Protected Mode drivers that come with Windows* 95, then the system may wish to prevent the O/S from swapping out that driver.

Microsoft has implemented a mechanism that allows the system to identify which Real Mode drivers are safe to replace. This "Safe Drivers List" is found in the file, IOS.INI, which is located in the Windows 95 root directory. If any installed Real Mode driver is listed by name in IOS.INI then Windows 95 assumes that it is safe to replace the currently installed Real Mode driver with its corresponding 32-bit Protected Mode counterpart.

Therefore if the audio driver's filename matches the name of a driver currently listed in IOS.INI, deleting it from IOS.INI will prevent Windows 95 from replacing it.

4. Conclusion / Recommendations

Sound Blaster hardware compatible audio lives on through 1997 as an OEM requirement for the volume "games compatible" desktop PC market segment.

However there is clearly a transition in progress that is moving the base audio subsystem off of the ISA bus and onto the PCI bus. The challenge of the day is to manage this transition so that it occurs cleanly from an architectural point of view, while at the same time ensuring that current customer functional requirements are met and support costs are kept to a minimum. Providing relocatibility for legacy resources may add additional cost to the implementation of the legacy features, and may very well generate a net increase in support calls.

4.1. Recommended DMA on PCI Scheme

Regardless of whether the audio vendor implements PC/PCI or DDMA, the legacy PCI solution is believed to be practically feasible as a motherboard integrated device.

Because of its centralized[5] nature and software transparency, PC/PCI is the recommended DMA solution. While it would be inappropriate to speak on behalf of other companies, what can be said is that some future Intel products are currently expected to support PC/PCI functionality. PC/PCI has been validated today within the context of a PCI-ISA docking station for ISA add-in cards, Cardbus host controllers and PCMCIA cards.[6]

4.2. Recommended Handling of 10-bit Address Aliases

The filtering out of 10-bit aliases introduces legacy software compatibility exposure. The level of this exposure is unknown.

It is recommended that the PCI legacy agent provide a means for either 10-bit decode or 16-bit decode with the boot up default being 10-bit decode. If the Plug and Play O/S asks the BIOS to configure 16-bit decode for audio and a legacy software title fails to operate correctly due to the aliasing compatibility

[5] As opposed to being distributed.
[6] Please address product specific inquiries to an Intel Sales Office near you directly.

issue, it would still be possible to get the software running correctly by rebooting DOS alone and re-launching the software. The reboot to DOS would have reset the I/O decode to 10-bit.

4.3. Recommended "Bi-Modal" Audio Controller Implementation

It is recommended that PCI bus resident ***legacy*** agents utilize the legacy compatible ISA I/O resources. In order to preclude any irreconcilable resource conflicts, it is also recommended that each of the legacy subfunctions have a disable bit that is visible to the BIOS in the agent's PCI configuration space register file (see Section A.2. of Appendix A) . It is also recommended that some relocatability be supported for ISA DMA and Interrupt resources. This will ensure compatibility with older ISA add-in adapters that may have "hardwired" resource conflicts with either DMA or IRQ resources. It is further recommended that the legacy audio subsystem upon activation by the BIOS, default to the legacy compatible I/O, DMA and IRQ settings.

However, in the interest of facilitating the cleanest migration of audio to the PCI bus, a "Native PCI Mode" audio interface also should be implemented. In this way the same audio controller could be used to preserve the end user's software investment (Legacy Mode), while at the same time be forward looking with a PCI 2.1 compliant solution (Native PCI Mode).

Native PCI Mode audio delivers enhanced performance and robust Plug and Play capabilities. New applications written for the Windows 95 operating environment should take advantage of Native PCI Mode.

Native PCI Mode audio should be compliant with PCI Spec. Revision 2.1, and should not use any ISA resources or legacy constructs such as PC/PCI DMA. All system resources should be assigned via industry standard PCI bus Plug and Play software, and generic PCI mastering should be used for audio sample transport.

An example register template for the recommended PCI Audio Function is provided in Appendix A. This example encompasses both Native PCI and Legacy audio modes.

Appendix A. Bi-Modal PCI Audio Controller

A.1. Legacy Mode Audio Controller

Given a motherboard integrated solution, the system BIOS should enumerate the legacy audio as a series of device nodes with no I/O address relocatibility. However any of the legacy audio's subfunctions could be independently disabled[7] if they need to be. In addition some flexibility in the selection of DMA and IRQ resources is recommended.

Since the PCI legacy audio agent is assumed to be a motherboard device and can be assigned legacy audio compatible I/O space first at boot time, I/O conflicts with other legacy audio add-in cards enumerated at a later time in the boot process shouldn't occur. New add-in peripherals, whether ISA-based, PCI-based, or USB-based are assumed to be Plug and Play compliant such that they can be mapped so as not to interfere with the legacy audio's I/O resource allocations. Even older ISA add-in adapters typically have a series of jumper blocks to allow for multiple selections for each of the supported subfunctions' I/O address ranges.

To avoid irreconcilable conflicts with older ISA add-in cards[8] that may be hardwired to a given IRQ or DMA resource, some amount of flexibility is recommended for these ISA resources.

In order to maintain the audio controller's legacy compatibility at low cost[9], it is recommended that the following legacy resources be supported with no I/O address relocatibility. Sample transport is recommended to be configurable between the three listed DMA channel options, and four choices for Sound Blaster and MIDI interrupts are suggested.

Legacy Audio Resource	Resource Assignment (base for I/O addresses)
Sound Blaster	220h
FM Synthesis	388h
Gameport (joystick)	200h
MPU-401 (MIDI UART)	330h
DMA	CH0, CH1, or CH3 (3 options)
IRQ (SB)	IRQ5, IRQ7, IRQ9, or IRQ11
IRQ (MIDI UART)	IRQ5, IRQ7, IRQ9, or IRQ11

DMA request/grant signaling is wired directly between the agent's PC/PCI request/grant pair and the PC/PCI arbiter, and the legacy IRQs are wired either directly to an 8259 compatible interrupt controller as separate IRQ signals or as a serialized stream over a single signal[10].

The audio subsystem should default to Legacy Mode, in an inactive state upon power up. Once the BIOS enables Legacy audio via a legacy audio control register, a hardware compatible legacy audio subsystem becomes activated.

[7] Plug and Play or CMOS Setup feature
[8] ISA add-ins that may be unrelated to audio
[9] Audio controller's related support cost as well as product cost
[10] Please address product specific inquiries to an Intel Sales Office near you directly

A.2. Native PCI Mode Audio Controller

The following PCI Configuration register map outlines the recommended programming model for the implementation of a dual mode PCI Audio Function .

Native PCI Mode for audio provides the avenue by which a broad spectrum of differentiation has been enabled with the advent of AC '97's architectural partitioning. This appendix is meant to set the direction for AC'97 Native PCI Mode audio implementations, and not to arbitrarily impose limits of any kind on the potential for future innovation and differentiation.

Configuration Offset	Register	Register Access
00h-01h	Vendor Identification (VID)	RO
02h-03h	Device Identification (DID)	RO
04h-05h	Command (COM)	R/W
06h-07h	Device Status (DS)	R/WC
08h	Revision Identification (RID)	RO
09h	Programming Interface (PI)	RO
0Ah	Sub Class Code (SCC)	RO
0Bh	Base Class Code (BCC)	RO
0Ch	*reserved*	-
0Dh	Master Latency Timer (MLT)	R/W
0Eh	Header Type (HEDT)	RO
0Fh	*reserved*	-
10h-13h	Native Mode Audio Base Address	R/W
14h-2Bh	*reserved*	-
2Ch-2Dh	Subsystem Vendor ID(SVID)	RO
2Eh-2Fh	Subsystem ID(SID)	RO
24h-3Fh	*reserved*	-
40h-41h	Legacy Audio Control	R/W
42h-FFh	*reserved*	-

A.2.1. VID—Vendor Identification Register

Address Offset: 01h-00h
Default Value: *Vendor Specific VID*
Attribute: Read Only
Size: 16 Bits

A.2.2. DID—Device Identification Register

Address Offset: 03h-02h
Default Value: *Vendor Specific DID*
Attribute: Read Only
Size: 16 Bits

A.2.3. COM—Command Register

Address Offset: 05h-04h
Default Value: 0000h
Attribute: Read/Write
Size: 16 bits

COM is a 16-bit control register. Refer to the PCI 2.1 specification for complete detail on each bit.

Bit	Description
15:10	**Reserved.** *Read 0.*
9	**FBE (Fast Back to Back Enable)**
8	**SEN (SERR# Enable)**
7	**WCC (Wait Cycle Control)**
6	**PER (Parity Error Response)**
5	**VPS (VGA Palette Snoop)**.
4	**MWI (Memory Write and Invalidate Enable)**
3	**SCE (Special Cycle Enable)**
2	**BME (Bus Master Enable)** *Read/Write* Bit(2) = "1" enables standard PCI bus mastering capabilities.
1	**MS (Memory Space)**
0	**IOS (I/O Space):** This bit controls access to the I/O space registers. If this bit is set to "1", access to the Native Mode Audio I/O interface is enabled. The Native PCI Mode Base Address register should be programmed prior to setting this bit.

A.2.4. DS—Device Status Register

Address Offset: 07h-06h
Default Value: *Vendor specific*
Attribute: Read/Write Clear
Size: 16 bits

DSR is a 16-bit status register. Refer to the PCI 2.1 specification for complete detail on each bit.

Bit	Description
15	**DPE (Detected Parity Error)**
14	**SERRS (SERR# Status)**
13	**MAS (Master-Abort Status)**
12	**RTA (Received Target-Abort Status)**
11	**STA (Signaled Target-Abort Status**
10:9	**DEVT (DEVSEL# Timing Status):** This 2-bit field defines the timing for DEVSEL# assertion. These read only bits indicate the audio controller's DEVSEL# timing when performing a positive decode.
8	**DPD (Data Parity Detected)**
7	**FBC (Fast Back to back Capable)**
6	**UDF Supported**
5	**66 MHz Capable**
4:0	**Reserved**. *Read as 0's.*

A.2.5. RID—Revision Identification Register

Address Offset: 08h
Default Value: *Component Specific RID*
Attribute: Read Only
Size: 8 Bits

A.2.6. PI—Programming Interface Register

Address Offset: 09h
Default Value: 00h
Attribute: Read Only
Size: 8 bits

A.2.7. SCC—Sub Class Code Register

Address Offset: 0Ah
Default Value: 01h
Attribute: Read Only
Size: 8 bits

This register indicates that the device is an audio device, in the context of a multimedia device (Base Class Code = 04h).

A.2.8. BCC—Base Class Code Register

Address Offset: 0Bh
Default Value: 04h
Attribute: Read Only
Size: 8 bits

This register indicates that the function implements a multimedia device.

A.2.9. MLT—Master Latency Timer Register

Address Offset: 0Dh
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

MLT is an 8-bit register that controls the amount of time the audio controller, as a bus master, can burst data on the PCI Bus. Programmed MLT count is timed in PCI clocks.

Bit	Description
7:4	**Master Latency Timer Count Value:** The MLT limits the duration of the PCI burst cycle to the number of PCI Bus clocks specified by this field. The MLT is used when the audio controller is operating in Native PCI Mode.
3:0	**Reserved.** *Read as 0's*

A.2.10. HEDT—Header Type Register

Address Offset: 0Eh
Default Value: 00h
Attribute: Read Only
Size: 8 bits

This register is always read as zero.

A.2.11. NMABAR—Native Mode Audio Base Address Register

Address Offset: 13h-10h
Default Value: 00000001h
Attribute: Read / Write
Size: 32 bits

The Native PCI Mode Audio function uses PCI Base Address register #1 to request a contiguous block of I/O space that is to be used for the Native Mode Audio software interface. Note that there are 6 total PCI base address registers that are available to a given PCI function. Only the first base address register is arbitrarily defined for this example. If the audio function requires more than 256 bytes of I/O space, or wishes to implement memory space as well, then some number of the remaining five available PCI base address registers could be implemented.

Bit	Description
31:8	Read/Write, base address. These bits are used in the I/O space decode of the Native Mode Audio interface registers. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. In this example the upper 24 bits are programmable which yields the maximum I/O block size of 256 bytes for this base address.
7:2	Hardwired to 0's
1	**Reserved.** *Read 0.*
0	**RTE (Resource Type Indicator):** This bit is set to one, indicating a request for I/O space.

A.2.12. SVID—Subsystem Vendor ID

Address Offset: 2Dh-2Ch
Default Value: *Vendor Specific SVID*
Attribute: Read Only
Size: 16 bits

This register should be implemented for any function that could be instantiated more than once in a given system, for example, a system with 2 audio subsystems, one down on the motherboard and the other plugged into a PCI expansion slot, should have the SVID register implemented. The SVID register, in combination with the Subsystem ID register, enable the operating environment to distinguish one audio subsystem from the other(s).

A.2.13. SID—Subsystem ID

Address Offset: 2Fh-2Eh
Default Value: *Vendor Specific SID*
Attribute: Read Only
Size: 16 bits

This register should be implemented for any function that could be instantiated more than once in a given system, for example, a system with 2 audio subsystems, one down on the motherboard and the other plugged into a PCI expansion slot. The SID register, in combination with the Subsystem Vendor ID register make it possible for the operating environment to distinguish one audio subsystem from the other(s).

A.2.14. LACR—Legacy Audio Control Register

Address Offset: 41h-40h
Default Value: 887Fh
Attribute: Read/Write
Size: 16 bits

This register provides control for independent enable/disable for each of the legacy audio subfunctions. Additionally, bit 15 defines a single soft switch for global legacy audio disable. A separate MIDI interrupt request enable/disable bit is provided so that the MIDI subsystem could be configured for either polled or interrupt driven MIDI I/O operation.

The audio controller powers up configured for legacy compatibility (Legacy Mode), however the global disable bit is set to fully disable the interface. To activate the legacy audio subsystem, the BIOS needs to flip the state of bit(15) to “0”.

Legacy Audio Control Register: Default = 887Fh

Bit	Description
15	**Legacy Audio Disable:** A "1" in this bit position acts as a global disable for legacy audio. When this bit is a "0", I/O transactions targeting any of the individually enabled legacy audio register blocks (see bits below) are positively decoded on the PCI bus.
14:12	**Reserved.** *Read as 0.*
11:0	**MIDI I/O IRQ Select:** This encoded field selects the ISA interrupt request to be used for the MIDI UART if configured for interrupt driven operation via bit(4) of this register. bit(9) bit(8) 0 0 - IRQ5 0 1 - IRQ7 1 0 - IRQ9 (Default) 1 1 - IRQ11
9:8	**SB IRQ Select:** This encoded field selects the ISA interrupt request to be used for the Sound Blaster legacy subsystem. bit(9) bit(8) 0 0 - IRQ5 (Default) 0 1 - IRQ7 1 0 - IRQ9 1 1 - IRQ11
7:6	**SB DMA Channel Select:** This encoded field selects the ISA DMA channel to be used for Sound Blaster audio sample transport. bit(7) bit(6) 0 0 - DMA CH0 0 1 - DMA CH1 (Default) 1 0 - Reserved 1 1 - DMA CH3
5	**I/O Address Aliasing Control:** A "1" in this bit position selects 10-bit I/O address decode which enables capture of aliases legacy I/O references. A "0" in this bit position selects 16-bit I/O address decode which eliminates all legacy I/O address aliasing. In either configuration the upper address bits, AD(31:16), should be sampled low before any legacy subfunctions will claim a PCI cycle
4	**MPU-401 IRQ Enable:** A "1" in this bit position activates the IRQ# specified in bits(11:10) for MPU-401 UART interrupt service requests. This bit enables either polled, or interrupt driven MIDI I/O support. This bit will be superseded by bit(15)=1
3	**MPU-401 I/O Enable:** A "1" in this bit position enables positive decode for all MPU-401 UART (MIDI) I/O references. This bit will be superseded by bit(15)=1.
2	**Game Port Enable:** A "1" in this bit position enables positive decode for all Game Port (joystick) register I/O references. This bit will be superseded by bit(15)=1.
1	**FM Synthesis Enable:** A "1" in this bit position enables positive decode for all FM Synthesis register I/O references. This bit will be superseded by bit(15)=1.
0	**Sound Blaster Enable:** A "1" in this bit position enables positive decode for all Sound Blaster register I/O references. If Sound Blaster decode is enabled via this bit, and the legacy subsystem is enabled via bit(15) = "0", then the DMA channel specified by bits(7:6) is activated for playback sample transport, and the IRQ# specified by bits(9:8) is activated for SB interrupt service requests.