AGP

A C C E L E R A T E D
G R A P H I C S   P O R T

DRAFT AGP V3.0
Interface Specification

Revision: 0.95
Original Date: May 2001

intel.

Revision History

Revision #	Date	Description
V0.9	Nov. 2000	First public release of the draft specification
V0.91	Dec. 2000	Several corrections made after initial review. Major items listed below: • FW made optional for both target and master – Sections: 2.7.4, 2.7.5, 5.8.4, 5.8.5 • **SBA** requests may start on any SB_STBx edge – Section: 2.1.2 • Clarification on "sticky" Type requests on **SBA** – Section: 2.3.2c • Tval changed to 5.5ns – Tables: 21 & 29 • Outbound DAC transaction support from core-logic made optional. – Sections: 2.7.4, 5.8.4 • Several editorial corrections made throughout specification
V0.91R	Apr. 2001	Table 7 corrected to make AGP Fast Writes optional for Core Logic
V0.95	May 2001	Major changes include: • Calibration Cycle transaction protocol changed. • DBI made mandatory for AGP Transmitter • DBI pin assignment on AGP connector changed • CAP ID changes for AGP Target • Updates to AC and DC timing specifications • 64b AGP8X section removed from Appendix A • Several minor inaccuracies corrected throughout document • Editorial corrections throughout document. • 4X speed operation in AGP3.0 signaling mode added • AGP3.0 introduced as the new name for the AGP8X Interface Specification

Table of Contents

List of Figures

List of Tables

1 Introduction and Overview

The AGP V3.0 Interface Specification (or AGP3.0) describes enhancements to the Accelerated Graphics Port (AGP) Interface. AGP3.0 offers a significant increase in performance along with feature enhancements to AGP2.0. This interface represents the natural evolution from the existing AGP to meet the ever-increasing demands placed on the graphic interfaces within the workstation and desktop environments.

While AGP3.0 clearly represents an evolutionary step in the AGP roadmap, it is not a superset of all previous interface versions. Certain AGP2.0 features are no longer supported by AGP3.0. Table 1 shows the evolution of the interface in terms of major specification releases.

Table 1: AGP Specification Releases

	AGP1.0	AGP2.0	AGP3.0
Signaling	3.3V signaling	1.5V Signaling	New 0.8V Signaling
Protocol	Pipelined transactions + Source synchronous clocking	AGP1.0 + Fast Writes	AGP2.0 + Some enhancements – some deletions (See Sec 1.5.1)
Speeds	2X, 1X	4X, 2X, 1X	8X, 4X
Connector	3.3V keyed	1.5V keyed, Universal	1.5V keyed

Throughout this document, the terms AGP1X, AGP2X, AGP4X, and AGP8X refer to the speed of data transfers on the AGP Interface while AGP1.0, AGP2.0, and AGP3.0 refer to specific releases of the AGP Interface specifications.

1.1 Organization of This Document

This document is to be used in conjunction with the AGP2.0 Interface Specification. Therefore, the focus will be on changes and additions made to that specification. The organization is as follows:

1. **Chapter 1:** Provides an introduction to the changes and enhancements made to the existing AGP interface specification.
2. **Chapter 2:** Describes changes and enhancements that apply to both desktop and workstation class platforms. Topics covering all changes include protocol and transactions, signal interface, platform dependencies, and programming.
3. **Chapter 3:** Provides the electrical and physical design specifications.

These three chapters complete the main body of the specification and are also referred to as the *core specification*. Several appendices describe additional changes and/or enhancements that generally target Workstation platforms. These are described as follows:

1. **Appendix A:** Describes optional features such as isochronous streaming protocol, further enhancements to flow control, etc.
2. **Appendix B:** Includes the optional programming enhancements that would be needed to support the features discussed in **Appendix A: Workstation Enhancements.**

Finally, **Appendix C** contains a glossary of terms used throughout this document.

1.2 Reference Documents

The following documents should be used as supporting reference material:

1. *Accelerated Graphics Port Interface Specification, V2.0*
2. *AGP System Design Guide, V1.5*
3. *PCI Local Bus Specification, V2.2*
4.

1.3 Need for AGP3.0

In the past, AGP bandwidth has been scaled up at regular intervals. AGP1X and 2X were introduced simultaneously, and AGP2.0 was introduced two years later. Now, AGP3.0 is recognized as the natural progression beyond AGP2.0. It is expected to satisfy the bandwidth demands on the graphics interface for at least two generations.

The need to increase bandwidth beyond AGP2.0 is based on the following trends:

- Workload content and real-time performance expectations are continuing to scale at a significant rate every year. These are measured by the complexity of the scene being visualized and the real-time scene changes and inter-activity that is expected.
- The capabilities of graphics sub-systems to render complex images are scaling at a tremendous rate due to significant architectural and technological improvements.
- The host platform's capabilities to support the graphics subsystem and workload requirements are also continually increasing. These capabilities include processor speeds, system memory capacity and bandwidth, and multiprocessing.

1.4 Evolutionary Aspects of AGP3.0

In this document, AGP1X, AGP2X, and AGP2.0 interfaces, as defined in the Accelerated Graphics Port Interface Specification Revision 2.0, will be referred to as "AGP" or "AGP interfaces." AGP3.0 is the next step in the evolution of AGP2.0. This evolutionary interface maintains the same connector and, except for a few additional signals to support the new signaling scheme, the same interface signals as AGP. However, since the electrical signaling and clock rates are significantly different between AGP and AGP3.0, compatibility at the platform level is achieved using a "Universal" motherboard. A Universal motherboard is one that supports both AGP2.0 and AGP3.0 signaling. Chapter 2 describes this concept in greater detail.

✎ **NOTE**

Table 2 lists the AGP features that are unchanged in AGP3.0.

Table 2: V2.0- Compatibility

AGP3.0 Feature	Impact
Uses the same connector and signal pins as AGP4x with a few additional signals to support the new signaling.	1.5 V connector support only.
Common Clock remains at 66 MHz.	Same flow control and PCI speeds. Buffering needs for Source Synchronous transfers double relative to AGP4X.
Same protocol and flow control as AGP.	Maintains interface level backward compatibility.
Continued support for PCI master and target transactions.	Allows the use of the existing PCI cores.
Same power delivery scheme as AGP and AGP Pro.	Uses AGP 1.5 V and AGP Pro 1.5 V connectors as is.
Same device enumeration and configuration schemes.	Backward compatible with the existing AGP enumeration schemes.
Routing rules, board impedance requirements etc. compatible with AGP.	Maintains evolutionary approach. Allows a motherboard to support both AGP2.0 and AGP3.0 modes of operation.

1.5 Classification of Changes

Changes to AGP2.0 are classified as follows:

1. **Performance Enhancements:** Changes here target improving the performance of the interface.
2. **Feature Removals:** Changes here target removing unused features in AGP with the intention of simplifying the interface.
3. **Feature Additions or Enhancements:** These target new usage models that were not the focus of AGP. Almost all feature additions or enhancements are specific to workstation platforms and are described in the Appendices.

1.5.1 Changes to AGP

The tables that follow describe the changes to AGP2.0. Descriptions of these changes are in the chapters and appendices that follow.

Table 3: Changes to AGP2.0 in Core Specs

Change	Section	Classification	Core-logic Impact	Graphics Card Impact
8X (533MT/s) transfer rate for Data and Side-band Address (SBA)	2.1.2	Performance	Required	Required
Parallel terminated, low voltage signaling	2.1	Performance	Required	Required
Hardware enforced coherency outside GART range for all transactions	2.4.1	Feature change	Required	Optional
"Long" Transaction Types Removed	2.3.2	Feature Removal	Required	Required
No PIPE mode Addresses	2.3.1	Feature Removal	Need not support PIPE	Require d
High Priority Transaction Support Removed	2.3.2	Feature Removal	Required	Cannot use HP transactions
Some changes to ordering rules	2.3.4	Performance	Optional	Required
No support for 3.3 V AGP signaling.	2.4.2	Feature Removal 1.5V AGP signaling still supported in "Universal Mode"	Need not support 3.3V AGP	Can't use 3.3V AGP
Calibration Cycle	2.1.3	Performance	Required	Required
Core-logic AGP Resources in PCI-to-PCI Bridge	2.5	Feature Enhancement	Optional	No Impact
Dynamic Bus Inversion	2.1.3.b	Performance	Required	Required

Table 4: Changes to AGP2.0 in Appendices A and B

Change	Section	Classification	Core-logic Impact	Graphics Card Impact
Support for Isochronous Transactions.	4.1	Feature Addition	Optional	Optional
Hardware enforced coherency inside GART region selectable on Page basis.	5.4	Feature Enhancement	Optional	Optional
Flow Control Change on Fast Writes.	4.2.2	Feature Enhancement	No Impact	Required
Multiple AGP Ports support.	5.2.3	Feature Enhancement	Optional	No impact
Ability to support multiple-page sizes in GART.	5.3.1	Feature Enhancement	Optional	No impact
Specified set of AGP3.0 configuration registers.	5.6	Feature Enhancement	Low	Low

2 Description of New Features

This chapter describes the AGP3.0 features that differ from AGP2.0 and covers the following areas:

1. Signal Interface Differences
2. Transaction and Protocol Differences
3. Platform Dependencies
4. Programming Changes

The electrical specification that describes the signaling scheme is in Chapter 3. All other changes that apply to the core specification are included in this chapter. Changes specific to Workstation platforms are covered in Appendices A and B.

2.1 Signal Interface Differences

AGP3.0 uses the same signal list (with some additions) and connector level pin assignment as AGP2.0. However, the signaling scheme is different. The new signaling scheme inverts the assertion levels of many signals. For details on the signal functions, refer to **AGP Interface Specification**. The primary motivation for doing this is to ensure that the unasserted logic state for any AGP3.0 signal is at "low voltage" which equates to no DC current flow hence a low power state. Throughout this document, the convention used for signal assertion is as follows:

- A signal name which ends with # has logic '1' = low voltage level (0V for AGP3.0) and logic '0' = high voltage (0.8V for AGP3.0).
- A signal name that does not end with # has logic '1' = high voltage (0.8V on AGP3.0) and logic '0' = low voltage (0V on AGP3.0).

Table 5: Signal List

AGP2.0 Signal	AGP3.0 Signal	Signaling Scheme in AGP3.0	Max Signaling Rate in AGP3.0	Assertion Level in AGP3.0
SBA	SBA#1	AGP3.0 signaling	533MT/s/Source Synch	1=Low; 0=High
SB_STB, SB_STB#	SB_STBF, SB_STBS	AGP3.0 Signaling	266MHz	1=High; 0=Low
AD	AD	AGP3.0 Signaling	533MT/s/Source Synch	1=High; 0=Low
AD_STB[1:0], AD_STB#[1:0]	AD_STBF[1:0], AD_STBS[1:0] DBI_HI, DBI_LO	AGP3.0 Signaling	266 MHz	1=High, 0=Low
C/BE#	C#/BE	AGP3.0 Signaling	533MT/s/Source Synch	C#: 1=Low; 0=High BE: 1=High; 0=Low
ST, PAR	ST, PAR	AGP3.0 Signaling	66MHz/Common Clock	1=High;0=Low
FRAME#, TRDY#, IRDY#, STOP#, GNT#, DEVSEL#, PERR#, SERR#, REQ#, IDSEL, RBF#, WBF#	FRAME, TRDY, IRDY, STOP, GNT, DEVSEL, PERR, SERR, REQ, IDSEL, RBF, WBF	AGP3.0 Signaling	66MHz/Common Clock	1=High;0=Low
CLK	CLK	Same as AGP2.0	66 MHz	Same as AGP
RST#, INTA#, INTB#, PME#, TYPEDET#	Same as AGP	Same as AGP2.0-	Asynch/Static	Same as AGP

2.1.1 New and Redefined Signals

AGP3.0 adds and redefines the following new signals to support the change in the signaling scheme. The new signals replace reserved pins on the current AGP connector.

Table 6: AGP3.0 New Signals

Name	Type of Change	Type	Description
AGP_Vrefcg	Redefined	Static	This pin is used by the motherboard (or core-logic) to supply AGP Vref or AGP3.0 Vref to the graphics card based on the configuration detected.
AGP_Vrefgc	Redefined	Static	This pin is used by the Graphics Card to supply AGP or AGP3.0 Vref to the motherboard based on the configuration detected.
GC_DET#	New	Static	This is pulled down to Vss by the AGP3.0 or Universal AGP3.0 Graphics Card. AGP Graphics Cards will leave this signal unconnected. The motherboard usage of this pin is implementation specific. This signal uses a currently "reserved" pin on the AGP connector.
MB_DET#	New	Static	This is pulled down to Vss by the AGP3.0 or Universal AGP3.0 motherboard. The graphics card usage of this pin is implementation specific. This signal uses a currently "reserved" pin on the AGP connector.
DBI_HI	New	Source Synchronous	This is a bit that goes along with **AD[31:16]** to indicate whether AD[31:16] needs to be inverted on the receiving end. **DBI_HI** = 0 **AD[31:16]** is not inverted, so receiver may use as is. **DBI_HI** = 1 **AD[31:16]** is inverted, so receiver must invert before use. On the AGP connector, **DBI_HI** is multiplexed with the signal, **PIPE#,** which is not used in AGP3.0.
DBI_LO	New	Source Synchronous	This is a bit that goes along with **AD[15:00]** to indicate whether **AD[15:00]** needs to be inverted on the receiving end. **DBI_LO** = 0 **AD[15:00]** is not inverted, so receiver may use as is. **DBI_LO** = 1 **AD[15:00]** is inverted, so receiver must invert before use. This signal uses a currently "reserved" pin on the AGP connector.

---

[1]It is important to note that SBA encodings for address and requests are inverted on AGP3.0 relative to AGP. This is because a NOP or idle state on this interface is all 1s, so in AGP3.0 this equates to a 0V level.

2.1.2 Clocking Changes

Table 5 describes the signaling rate and the clocking scheme associated with the signals on the AGP3.0 interface. The major change from AGP is that the source synchronous signals are "strobed" at 8X the common clock frequency of 66 MHz. All signals that are clocked using the common clock retain the signaling rate of AGP. The names of the strobes used to clock the Source Synchronous signals have been changed from those used in AGP, to more accurately specify their usage.

Figure 1 illustrates the effect of the 8X signaling on the Side-band Address (**SBA**) interface. T3, T2 and T1 stand for Type 3, 2 and 1 **SBA** requests, each of which is sent in two 8-bit parts labeled H and L. Note that there is an additional request type not shown in this example called Type 4 used for memory addresses beyond 36 bits. NOP reflects an idle bus where the **SBA#** signals are 1s or in low voltage level state. Description of these request types may be found in the AGP interface specification. Some points to be noted follow.

1. To initiate a new memory transaction, an AGP Master may need to generate anywhere from one to all four request Types. A complete request bundle including Types 4, 3, 2, and 1 now fits into one, common clock cycle in AGP8X. Such a request requires two common clock cycles in AGP4X and four common clock cycles in AGP2X.

2. While it is possible to initiate multiple transactions[2] in one common clock cycle, AGP enforces a rule of at most one transaction per common clock cycle. The first Type request of a transaction may start at any rising **SB_STBF** edge within a common clock cycle at the AGP3.0 Master. The Master must insert NOPs into any unused request slots.
3. The two strobes **SB_STBF** and **SB_STBS** alternately use the L->H edge to latch the **SBA** data. The illustration in Figure 1 shows the one strobe as the inverse of the other. However, the specification does not require this. The only requirement is for each strobe to provide an assertion edge centered in each alternate **SBA** transfer window.
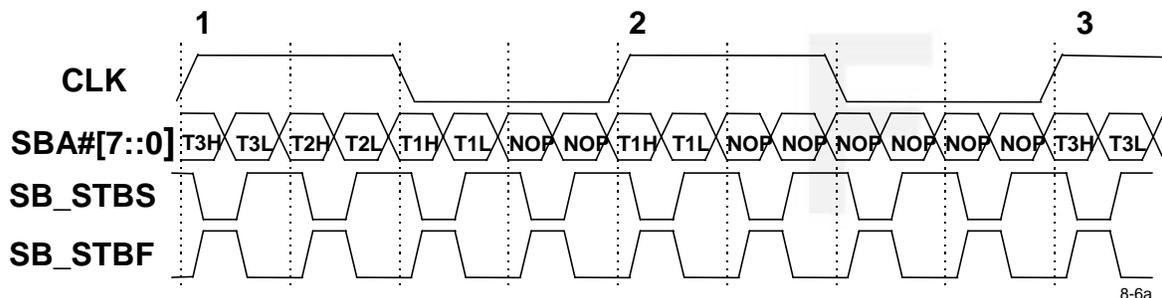
Figure 1: 8X SBA Addressing Showing Three Consecutive SBA Requests

To restart **SBA** strobes after they have been stopped, a synchronization sequence must be followed. This sequence is based on the AGP interface specification and is described in Sec 3.2.2. When stopped, the **SBA** Strobes are both in LOW state. In AGP2.0 the **SBA** Strobes are treated differentially. When restarting the strobes, **SB_STBF** starts first followed by **SB_STBS**. For either strobe, the rising edge is used for latching data.

---

[2] A transaction is initiated by a Type 1 request, which triggers the core-logic to begin execution. Types 2 through 4 requests do not initiate a transaction hence any number of these can be sent in a common clock cycle.

Figure 2 and Figure 3 illustrate the 8X data transfer timing diagram on the AD interface. The **AD_STBF** and **AD_STBS** are used to latch the data at the receiver. As in AGP, **CLK** based PCI signals **GNT**, **TRDY** and **IRDY** control the flow of data. The timing relationship between **IRDY/TRDY** and the strobes is designed specifically to transfer data from the strobe latches to the **CLK** based latches in the receiver.
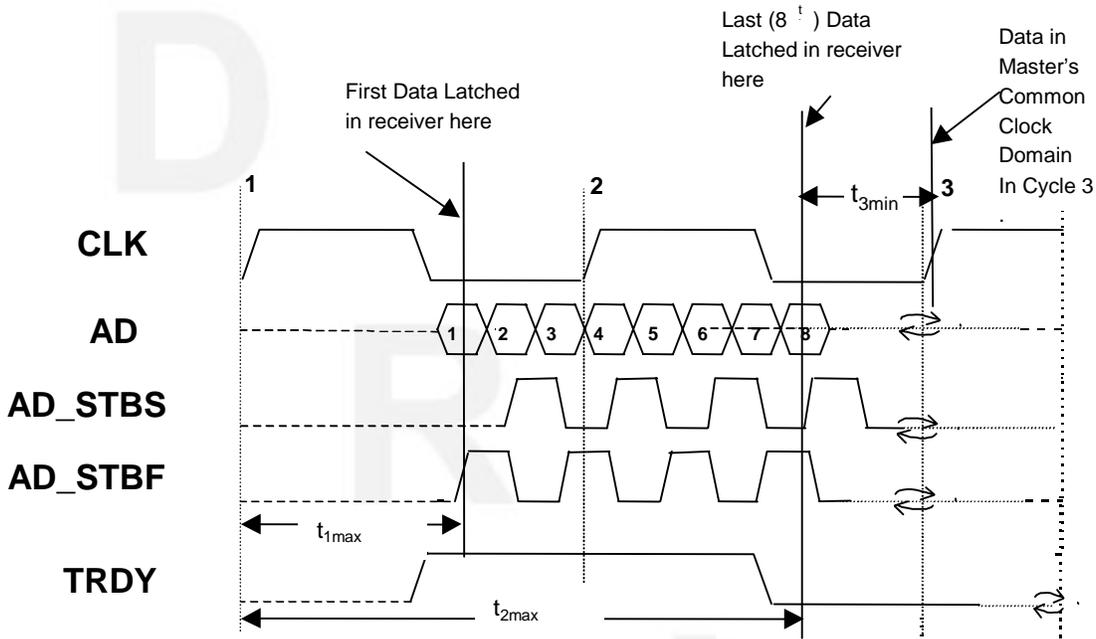
Figure 2: 8X Data Transfers on AD Interface

The major change from AGP2.0 is that the number of source synchronous AD transfers that happen within a common clock period has doubled. The timing parameters, t_1, t_2, and t_3, have values different from AGP2.0 to account for AGP3.0 signaling technology. The relationship between **TRDY** and the 8X data transfer remains the same as the relationship between **TRDY** and the 4X data transfer that is in AGP2.0. **TRDY** seen by the receiver after it is latched in cycle 2 means data will be available in cycle 3.
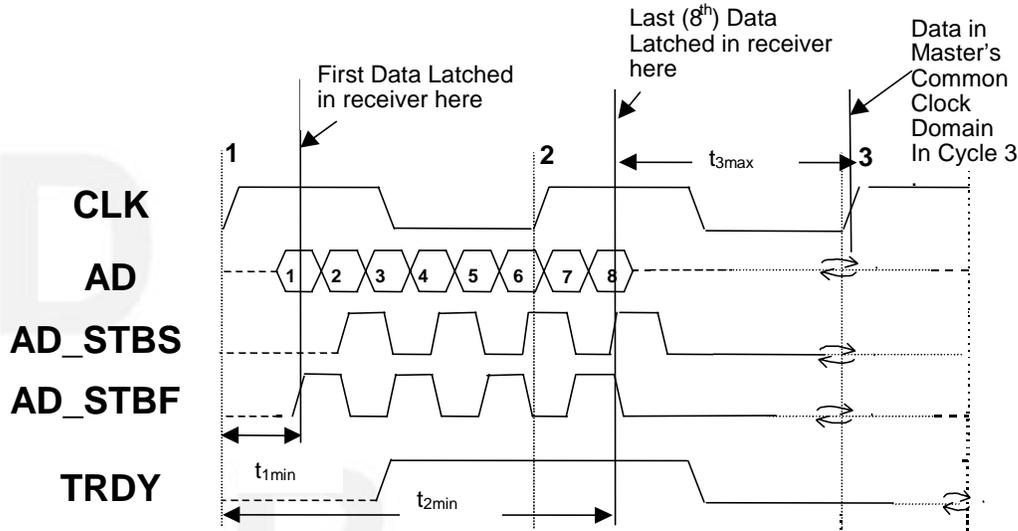
Figure 3: Minimum TRDY and AD_STB Timing Relationship

Figure 3 shows the minimum delay between common clock and **AD_STB** (and the corresponding data). The delay t_{1min} represents the earliest the first data is latched into the receiver using the strobe. The delay t_{2min} represents the earliest the last of the eight pieces of data per common clock is latched in the receiver.

The time period t_{3max} represents the setup time to the next common clock edge to transfer the eight pieces of data to the common clock domain. In this case, there is plenty of setup time, and the data appears to precede **TRDY** by a considerable amount.

Figure 2 illustrates a case where the delay between common clock and the **AD_STB** is the greatest. However, even with this delay, t_{3min} is still sufficient to meet the setup time of the common clock edge for cycle 3. Nothing has changed between the two cases in regards to the common clock's domain. However, data in this case appears to be lagging behind **TRDY** by a considerable margin, however, according to the timing diagrams.

2.1.3 AGP3.0 Signaling Scheme

The AGP3.0 interface is point-to-point, the same as AGP2.0. All signals that use the AGP3.0 signaling scheme are parallel terminated to **Vss** (signal reference) on both ends with a termination value of the interconnection trace impedance. The termination is provided on-chip with an implementation specific scheme to closely match the value to interconnect trace impedance.

Only the 8X signals need to adopt the new signaling to sustain the increased frequency, although the 1X signals will also use the same signaling scheme.

The 66 MHz **CLK** and **RST#** signals continue to be 3.3 V.

2.1.4 Calibration Cycle

The AGP3.0 signaling scheme requires tight control over variation in the key specification parameters: on-die termination impedance, signal swing, and slew rate. The key variables of process, voltage, and temperature cause the variations in these parameters. To maintain control over the variations, static or dynamic approaches could be utilized. In a static approach, the design accommodates, up front, the worst-case changes within the three variables.

A dynamic approach will adjust the parameters as changes in the variables occur. Process variations need only be adjusted once. Voltage and temperature, however, will change during operation and must be adjusted dynamically. In order for dynamic adjustments to occur, the interface must enter a quiescent state when there is no risk of corrupting an ongoing transaction. AGP3.0 provides a scheme called the *calibration cycle* that can be used for dynamic adjustment (or compensation) of the key specification parameters. The compensation scheme itself is implementation specific and is beyond the scope of this document.

Only the high-speed source synchronous AGP signals need dynamic calibration. Most other slower "common-clock" signals are calibrated once on power-up while a few don't need any adjustments. Table 7 describes this further.

Table 7: Signal Calibrations

Signals	Type	Core-Logic (Target)	Graphics Chip (Master)
SBA#, SBA Strobes	SBA Source Synchronous Uni-directional Type	Receiver Only Needs termination adjustment	Driver Only Needs signal swing and slew rate adjustment
AD, C#/BE, AD Strobes, DBI_HI, DBI_LO	AD GROUP Source Synchronous Bi-Directional	Needs termination, swing & slew rate adjustments	Needs termination, swing & slew rate adjustments
ST[2:0], IRDY, TRDY, FRAME, GNT, DEVSEL, STOP, REQ, etc.	Common Clock Group	Slow enough that one adjustment on power up is sufficient.	Same as Core-Logic
CLK, INT#, RST# etc.	Misc.	No adjustments	No adjustments

Since the **AD** signal group is bi-directional, an explicit bus event is needed to allow driver and termination update to occur. AGP3.0 defines a calibration cycle that facilitates this activity. The calibration cycle is initiated by the core-logic on a periodic basis. The period for this event is determined by the programmed value in field PCAL_CYCLE of the core logic's AGPCMD register (see section 2.7.5.). The core-logic sets PCAL_CYCLE with a conservative default value. This default period can be changed by configuration software based on the actual requirements advertised by core-logic and master (graphics chip) in the field CAL_CYCLE of the AGPSTAT register.

When the core-logic performs a calibration cycle, the following sequence of events occur:

1. The core-logic waits to regain control of the interface. The arbiter will not grant any pending request from the master until the calibration cycle is completed.
2. Core-Logic asserts **GNT** and **ST[2:0]**=110 to indicate start of Calibration Cycle.
3. Both the Graphics chip and the Core-Logic may start their **AD** group buffer updates starting in cycle T2.
4. Core-Logic and Graphics chip must assert **IRDY** and **TRDY** respectively to indicate completion of internal buffer updates. Neither side needs to wait for the other side to give its completion signal.
5. Earliest assertion of **IRDY** or **TRDY** is in T3. The latest assertion of **IRDY** and **TRDY** are implementation dependent. However, to minimize any performance impact on other AGP transactions, designs are strongly encouraged to not delay the assertion of **IRDY** or **TRDY** by more than 8 clock cycles, i.e. beyond T10.
6. Once the Core-logic asserts **IRDY** it must keep it asserted until it samples **TRDY** asserted. It will then de-assert **IRDY** to end calibration cycle.
7. Similarly, once the Graphics chip asserts **TRDY**, it must continue to assert it until it samples **IRDY** asserted upon which it will de-assert **TRDY** and end calibration cycle.
8. A new transaction may begin in the clock cycle immediately following the de-assertion of **IRDY** and **TRDY** on the AGP interface.

The timing diagram below illustrates this sequence.

Earliest that **IRDY** or
TRDY can be asserted

	T1	T2	T3	T4	T5	T1

AGP CLK

ST[2:0]   110   000   Next Cycle

GNT

IRDY

TRDY   **Delayed 1**

AD[31:0]

C#/BE[3:0]
AD_STBx
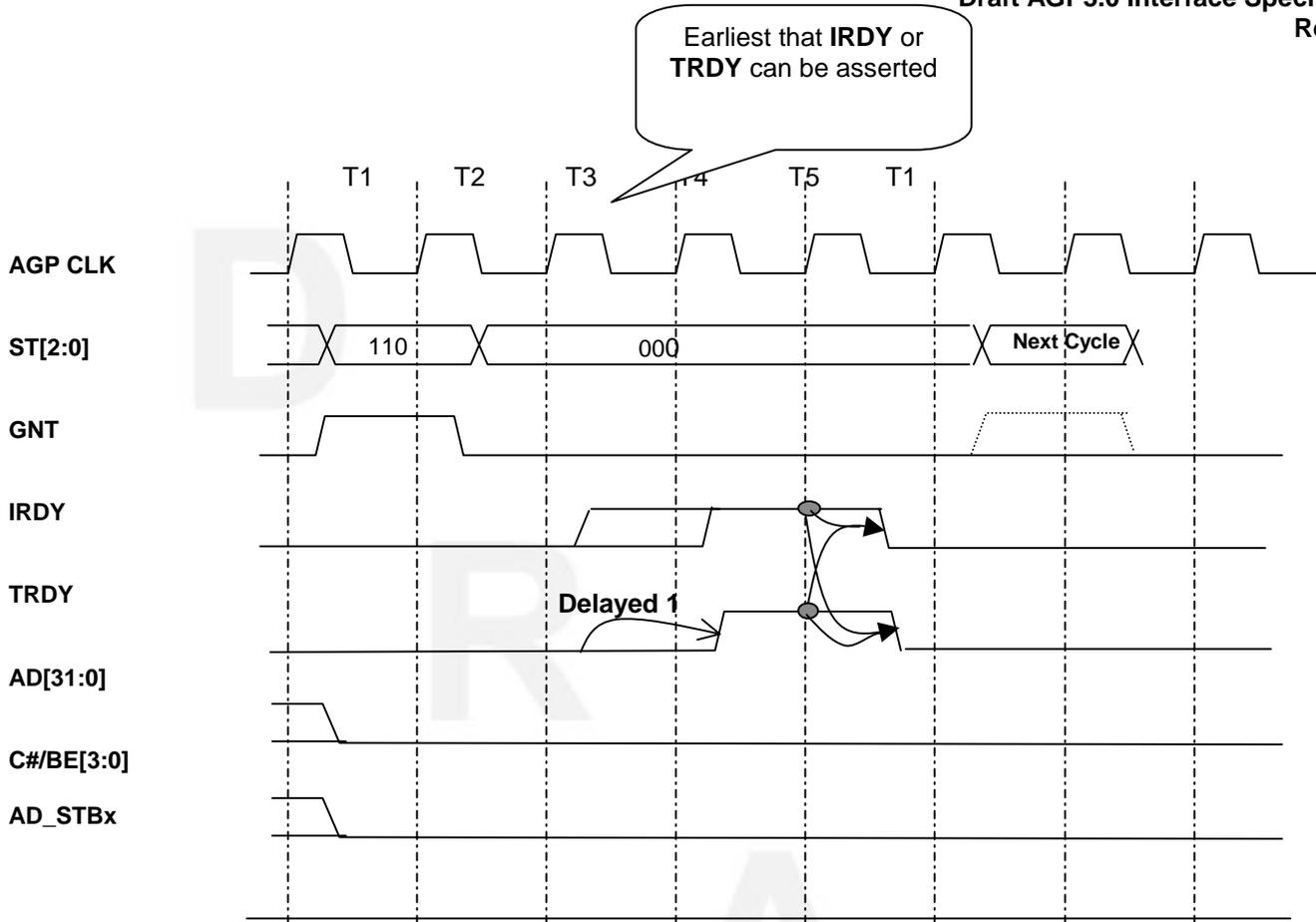
Figure 4: AGP3.0 Calibration Cycle

2.1.4.1 SBA Group Calibration

The **SBA** interface calibration is initiated by the AGP3.0 Master (Graphics Chip) and is triggered by the calibration cycle. Soon after the calibration cycle, the AGP3.0 Master initiates calibration of the **SBA** interface by stopping the **SBA** strobes and driving the entire interface low for a minimum of eight AGP common clock cycles (as required by the sideband synchronization specification) during which time it performs the needed adjustments to its **SBA** I/O buffers. The AGP3.0 Target detects the stopping of the strobes within a maximum of six common clock cycles and then performs the required termination calibration within two common clock cycles while the **SBA** activity is still suspended by the master. Following this, the AGP3.0 Master can restart the SBA strobes and resume normal activity. The stopping and starting of **SBA** strobes must follow the sideband synchronization scheme described in Section 3.2.2. The timing diagram describing this process is illustrated in Figure 5.

The Target is allowed a maximum of six cycles to detect that the strobes are stopped. During this period, the Master **must not** cause any glitches on the strobe lines due to any compensation updates. After this period is over the Master continues to drive NOPs on the **SBA** and keep the strobes low for at least two additional common clock cycles. Both the Master and Target may perform the buffer updates during this period. If the Master chooses, it may extend the update period by delaying the strobe restart sequence. However, the Target may not depend on this and must complete its buffer updates within two common clock cycles. The sequence of events is illustrated in the timing diagram shown in Figure 5.
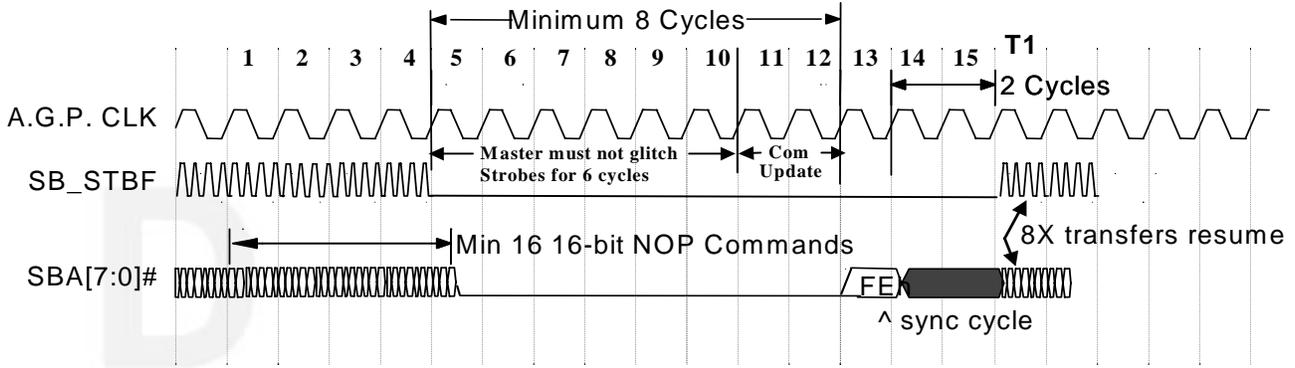
Figure 5: SB_STB Stopping and Starting Sequence

The exact timing between the **AD** group calibration cycle and the start of **SBA** calibration is not important as long as it is completed before the start of the next **AD** group calibration cycle. Since the smallest possible calibration period is 4ms (see section 2.7.4), the master must ensure that **SBA** calibration is completed within this time.

2.1.4.1.1   SBA CALIBRATION WITH SBA STROBES STOPPED

The above description of **SBA** calibration by the master and target assumed that the **SBA** strobes (**SB_STBF** and **SB_STBS**) were running when the **AD** calibration cycle occurred. Now we consider the case where the master, for any reason, stops the strobes prior to or during an **AD** calibration cycle. The following steps ensure proper calibration of the **SBA** buffers on both sides before the master restarts the **SBA** strobes. Figure 6 should be used as reference.

1. If the **SBA** strobes are stopped, the master has the opportunity to restart them to resume normal **SBA** activity, by asserting the Sync cycle on the **SBA** (as shown in Figure 5) as late as the 5th clock period of the **AD** calibration cycle (T5 in Figure 6), which is 4 clocks after **GNT** is sampled active by the master. This provides sufficient time for the master to detect the **AD** calibration cycle and prevent the resumption of **SBA** activity. Beyond T5, the master must no longer issue a Sync cycle to restart the strobes until 16 clocks have transpired which in this example is in T22.
2. Once the **AD** calibration cycle is complete (T5 in example shown in Figure 6), the target will have up to 4 additional clock cycles to test for **SBA** activity (presence of strobes or a Sync cycle on **SBA**). This period of testing completes by T9 per this example. If **SBA** activity is detected, the target waits for the strobe stopping sequence described in Section 2.1.4.1 in order to update the **SBA** buffers.
3. If no **SBA** activity is detected by the end of the testing period following **AD** calibration cycle, the master and target are given 12 clock cycles to calibrate the buffers in the **SBA** group. In this example, this period ends in T21. During this time, the target must ignore the **SB_STBx** signals since the master may glitch them during calibration.
4. Following this **SBA** calibration period, the master may assert Sync on **SBA** to restart the strobes in the immediately succeeding clock cycle – T22.
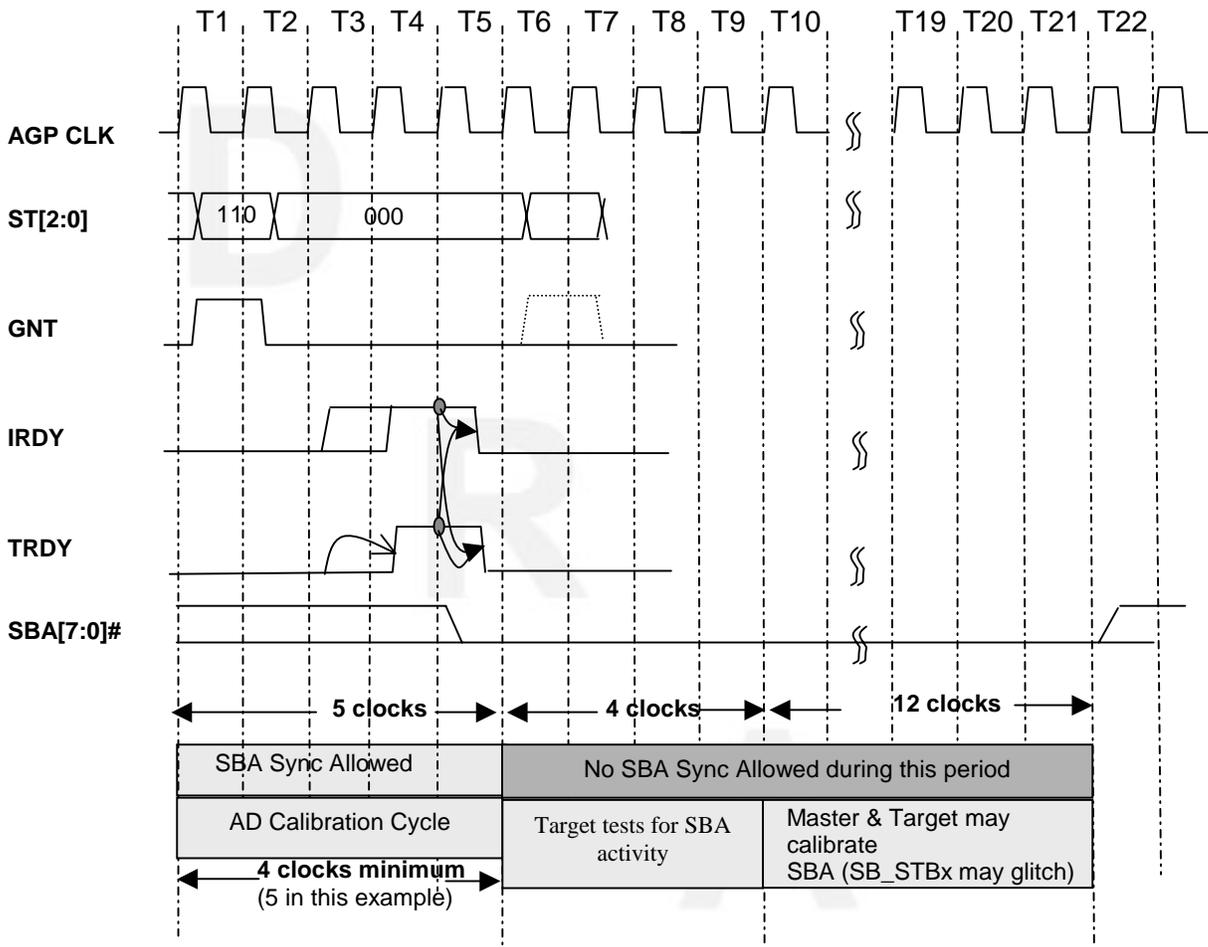
Figure 6: SBA Calibration When SBA Strobes are Stopped

2.1.4.2 Initial Calibration of Buffers

When the system comes out of a power-up reset, the AGP3.0 I/O buffers need to be calibrated prior to *any* operation on the AGP interface. A minimum of **100 microseconds** should be allowed following removal of AGP Reset condition during which the interface is locked and in unasserted state. The Master (graphics chip) and Target (core-logic) use this time to calibrate their buffers based on the selected mode of operation (AGP2.0V or AGP3.0). See section 2.4.2 for details on mode selection.

The reset condition sets the default value of PCAL_CYCLE to '0', which translates to a 4ms period. Once the initial calibration period is over, the normal operation is started by the core-logic. Calibration Cycles can only start after the initial calibration period is completed.

The core-logic enables calibration cycle only in the AGP3.0 mode of operation. The calibration cycle is disabled when operating in normal AGP2.0 mode (1.5V signaling).

2.1.5 Dynamic Bus Inversion

In order to mitigate the effects of simultaneous switching outputs, AGP3.0 adopts a scheme called Dynamic Bus Inversion (DBI) to limit the maximum number of simultaneous transitions on source synchronous data transfers. DBI impacts only **AD[31:0]** and is used during source synchronous and common clock transfers. Two new signals are defined to support DBI. **DBI_LO** and **DBI_HI** are used to implement DBI on **AD[15:0]** and **AD[31:16]** respectively. The scheme used to implement DBI on source synchronous transfers is as follows:

Whenever the number of bit transitions in **AD[15:0]** (or **AD[31:16]**) from one source synchronous period to the next exceeds eight, the entire field is inverted by the transmitter in order to limit the maximum transitions to eight. For example, if **AD[15:0]** changes from FF10 (hex) in source synchronous cycle A to 0000 (hex) in source synchronous cycle B, the DBI scheme is triggered in cycle B, thus inverting the **AD[15:0]** to produce FFFF (hex). In this example, the number of transitions without DBI is nine, while with DBI is seven. In order to signal to the receiver that the **AD[15:0]** are inverted in cycle B, **DBI_LO** will be asserted high. The same scheme is used on **AD[31:16]**. **DBI_HI** is used to signal the inversion. The receiver samples **DBI_HI** and **DBI_LO** to determine whether to invert **AD[31:0]** before using it.

A similar scheme applies to common clock and Frame based (PCI) address and data transfers. In these instances, DBI applies to transitions from one common clock period to the next.

Implementation of DBI is *required* in the transmitter and receiver for both the Master (graphics chip) and the Target (core logic) when operating in 8X speed and in AGP V3.0 signaling mode. When doing Frame based PCI transfers or 4X speed transfers in the same signaling mode, DBI is optional in the transmitter but still required in the receiver.

DBI is *not supported* when in AGP2.0 or AGP1.0 signaling modes. Table 8 illustrates the application of DBI in various modes of operation.

Table 8: DBI Implementation Requirements

Signaling Mode	Transfer Speed	Transmitter DBI	Receiver DBI
AGP V3.0	8X	Required	Required
AGP V3.0	4X	Optional	Required
AGPV3.0	Frame Based PCI	Optional	Required
AGP 1.0 or 2.0	Any	Not Supported	Not Supported

Note that the addition of the DBI signals extends the definition of the AD strobe groups as follows:

- **AD_STBF[0]** and **AD_STBS[0]** are associated with **AD[15:0]**, **C#/BE[1:0]**
- **AD_STBF[1]** and **AD_STBS[1]** are associated with **AD[31:16]**, **C#/BE[3:2]** and **DBI_HI** and **DBI_LO**

Due to the locations of the AGP connector pins assigned to the two DBI signals, both of them are placed in the strobe group associated with the upper sixteen AD bits.

Figure 7 illustrates the use of DBI on source synchronous transfers. In this example, only **DBI_LO** usage is shown.
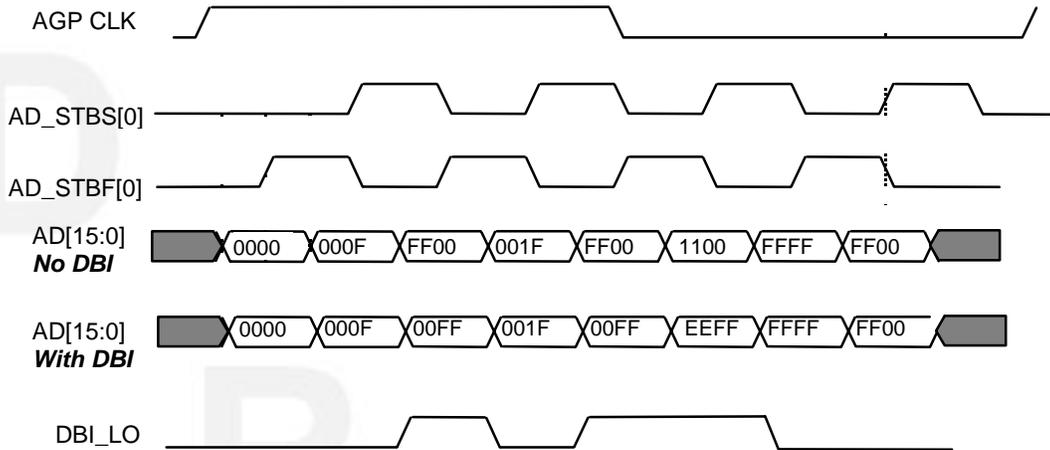
Figure 7: Usage of DBI on Source Synchronous Transfers

2.1.5.1 AGP Connector Pins for DBI

On the AGP connector, **DBI_LO** is assigned the pin B14 (reserved in AGP2.0 specification), and **DBI_HI** is assigned the pin A12 (used for **PIPE#** in AGP2.0 specification). The implication of using A12 for **DBI_HI** is that in designs that are "universal" and can operate in AGP2.0 or AGP3.0 modes, both the Master and Target must multiplex **PIPE#** and **DBI_HI** on the same wire. However, for a given mode of operation (AGP2.0 or AGP3.0), the meaning of this shared wire is fixed after power-up configuration. Such multiplexing is possible only because **PIPE#** is not used in AGP3.0, and DBI is not used in AGP2.0.

2.1.5.2 Parity Generation and DBI

During **FRAME** based PCI transfers, the signal **PAR** reflects even parity across **AD[31:0]** and **C#/BE[3:0]**. During these transfers, the transmitter generates parity *before* the application of DBI on **AD[31:00]**. The receiver checks parity only *after* performing any data inversion on **AD[31:00]** required by the assertion of **DBI_HI** and **DBI_LO**.

2.2 Summary of Interfaces

The signal pins defined by AGP3.0 may be used to perform PCI or AGP transactions. Each signal may be required, optional, or not applicable, depending on the type of device (graphics or core-logic), which interface (PCI or AGP) is supported, and the type of agent (Master or Target). Table 9 is a summary of the capability supported by function and agent. Table 10 also includes a summary of the PCI signals supported by function and agent.

Table 11 is a summary of the AGP3.0 signals supported by function and agent. For example, the second column represents a graphics agent that supports the PCI interface as a Target. The **Support**

column indicates whether the interface is required or optional. For example, a graphics agent requires the PCI target interface. The gray boxes indicate that the signal is not applicable to the function.

Table 9: Summary of Interfaces Based on Function and Agent

Device	Graphics				Core-logic			
Interface	PCI		AGP3.0		PCI		AGP3.0	
Agent	Target	Master	Master	FW$_{Target}$	Target	Master	Target	FW$_{Master}$
Support	R	O	O	O	R	R	R	O

Legend:

R = Required
O = Optional

Table 10: Summary of PCI Signals Based on Function and Agent

Device	Graphics				Core-logic			
Interface	PCI		AGP3.0		PCI		AGP3.0	
AGENT	Target	Master	Master	FW$_{Target}$	Target	Master	Target	FW$_{Master}$
FRAME	R	R		R	R	R		R
IRDY	R	R	R	R	R	R	R	R
TRDY	R	R	R	R	R	R	R	R
STOP	R	R		R	R	R		R
DEVSEL	R	R		R	R	R		R
IDSEL	R[1]				R[2]			
PERR	R	R			O	O		
SERR	O	O			O	O		
REQ		R				I	R	I
GNT		R	R			I	R	I
RST#	R	R	R	R	R	R	R	R
AD[31::00]	R	R	R	R	R	R	R	R
C#/BE[3::0]	R	R	R	R	R	R	R	R
DBI_HI, DBI_LO	R	R	R	R	R	R	R	R
PAR	R	R	NS	NS	R	R	NS	NS
LOCK	NS	NS	NS	NS	NS	NS	NS	NS
INTA#	O	O	O		R	O	R	
INTB#	O	O	O		R	O	R	
CLK	R	R	R	R	R	R	R	R
PME#	O				O			

Legend:

R   = Required
O   = Optional
I   = Internal signal
NS = Not supported
            A shaded cell indicates that the signal is not applicable to the function.

✎ NOTE

1. **IDSEL** is not a signal on the connector. See Section 2.1.4 for details of how **IDSEL** is implemented.
2. **IDSEL** is typically an internal signal for the core-logic.

Table 11: Summary of AGP3.0 Signals Based on Function and Agent

Device	Graphics		Core-logic	
Interface	PCI	AGP3.0	PCI	AGP3.0

Agent	Target	Master	Master	FW$_{Target}$	Target	Master	Target	FW$_{Master}$
ST[2::0]		O[3]	R			R	R	
PIPE			NS				NS	
SBA[7::0]#			R				R	
RBF			O[1]				R	
WBF				O[2]				R
AD_STBF0			R	R			R	R
AD_STBF1			R	R			R	R
AD_STBS0			R	R			R	R
AD_STBS1			R	R			R	R
SB_STBF			R				R	
SB_STBS			R				R	

Legend:

R = Required
O = Optional
I = Internal signal
NS = Not supported

              A shaded cell indicates that the signal is not applicable to the function.

✎ NOTE

1. *The AGP Master does not require **RBF** if it can always accept the return of Read data.*
2. *The AGP Master does not require **WBF** if it always can accept the first block of FW data.*
3. ***ST** signals are only optional to the PCI master if no AGP support is implemented.*

2.2.1 IDSEL Usage for Configuration

Initialization of an AGP device occurs via the configuration mechanism defined by the *PCI Local Bus Specification*. An AGP Master is composed of a PCI Target interface and an AGP Master interface. (Optionally, the AGP Master can also include a PCI Master interface). The PCI target interface follows the *PCI Local Bus Specification*. This requires the device to respond to a PCI configuration transaction when a configuration request (read or write) is decoded and **AD01** and **AD00** are both "0" and the device's **IDSEL** is asserted.

Since **IDSEL** is not a signal on the AGP connector, it must be connected to **AD16** on the graphics component. The designer of the AGP Master must be careful as to how this connection is made since the **AD** lines are running at very high speeds. This connection between **IDSEL** and **AD16** must not electrically load **AD16** in the physical interconnect between the core-logic and the graphics chip.

2.3 Transaction and Protocol Changes

Most changes in transactions and protocol result from the 8X data rate and removal of certain transaction types. These changes are described in this section. Additional enhancements in transaction types specific to workstation platforms are described in Appendix A: Workstation Enhancements.

AGP references two types of transactions: **AGP transactions** and **PCI transactions**. AGP transactions use AGP protocol semantics and include transfers initiated by the AGP Master using the **SBA** Interface as well as core-logic initiated Fast-Writes. **PCI transactions** refer to transactions initiated by the AGP Master or core-logic that use the PCI Protocol semantics.

2.3.1 AGP Transaction Requests

In the past, AGP transactions could be generated using two modes: PIPE and **SBA**. The core-logic had to support both modes, while the AGP Master could optionally use either. AGP3.0 does not support using PIPE mode to generate an AGP request. This leaves only the **SBA** scheme, which is no longer optional. When operating in AGP3.0 mode, the **PIPE#** signal pin on the AGP connector is given a new function, **DBI-HI**. A "universal" implementation must multiplex **PIPE#** and **DBI_HI** onto the same signal pin and select the right function based on the signaling mode of operation.

2.3.2 Removal of Transaction Types

AGP3.0 removes the use of certain AGP2.0 transactions, which are now classified as *reserved* events. The core-logic response to these reserved transactions is implementation specific and beyond the scope of this specification.

2.3.2.1 High Priority Transactions

AGP3.0 does not support the AGP High Priority Read and Write transactions.

Since only Low Priority transactions are supported, the priority classification no longer has any meaning under AGP3.0. Therefore Low Priority AGP Read or Write transactions will be referred simply as AGP Read or Write transactions.

2.3.2.2 Transactions of the "Long" Type

AGP does not support the "Long" version of the AGP Read transactions. The "Long" transactions were used to specify data payload sizes greater than 64 bytes. AGP3.0 transaction size is limited to a maximum of 64 bytes. These transactions are aligned to 8 byte address boundaries.

When the AGP master needs to read a large amount of data from system memory, it must split this up into multiple 64 byte transaction requests. For optimum performance, these requests should be naturally aligned to 64 byte boundaries.

In some cases, the Core-Logic may have been optimized for data transfers that are greater than 64 bytes. For instance, if the core-logic is designed for cache line accesses of 128 bytes, it may perform best with AGP request sizes that are some multiple of 128 bytes. A new 3-bit field called ARQSZ in the Core-Logic's AGPSTAT[15:13] register is used to provide the optimum transfer size information to software. The AGP Master should be programmed to group consecutively addressed 64 byte requests to effectively meet the Core-Logic's optimum transfer size expectations. Note that this is only a performance optimization and not a functional requirement. Refer to Sections 2.7.5 and 2.7.4 for more details.

2.3.2.3 Transaction Encoding Tables

The **SBA** Interface is used to send four types of requests similar to AGP's:

1. Type 1: address bits 14:3 and the Length Field [LLL]. The Length is in units of 8 bytes.
2. Type 2: 4 bit Request Codes (CCCC) and address bits 23-15.
3. Type 3: address bits 35:24.
4. Type 4: address bits 47:36. (This is optional and used only if the address is >64 GB.)

Types 2 through 4 are considered "sticky"; after they are sent to the Target, they need not be sent again unless any of the fields within them change. If such change occurs, only the affected Type request has to be re-sent to the target.

Whenever the core-logic executes a reset (as it would during a normal power-up event) the internally stored state of the "sticky" request fields, Type 2-4, must be initialized such that the address bits are cleared to 00h. Also, the internal state of the "sticky" requests in the core-logic survives the stopping and restarting of **SBA** strobes during normal operation. However, the AGP master should not depend on the core-logic to retain the state of the "sticky" requests through any power-down mode event such as those specified under ACPI.

The "CCCC" field contains the Bus operation or request as itemized in Table 12.

Table 12: AGP3.0/AGP2.0 Bus Requests

CCCC	AGP2.0	AGP3.0
0000	Read (low-priority)	Read (Asynchronous[5])
0001	Read (high-priority)	Reserved
0010	Reserved	Reserved
0011	Reserved	Reserved[3]
0100	Write (low-priority)	Write (Asynchronous[5])
0101	Write (high-priority)	Reserved
0110	Reserved	Reserved[3]
0111	Reserved	Reserved[3]
1000	Long Read (low-priority)	Reserved
1001	Long Read (high-priority)	Reserved
1010	Flush (low-priority)	Flush
1011	Reserved	Reserved
1100	Fence (low-priority)	Fence (for reads & writes)
1101[4]	Reserved)	Reserved
1110	Reserved	Reserved[3]
1111	Reserved	Reserved

Associated with the previous requests are the *status codes* from the core-logic to signal the completion response for the disconnected transaction. Certain codes are reserved in AGP3.0. The codes listed as ST[2:0] on the signal interface are common clock signals that are valid only when **GNT** is asserted. These codes are shown in Table 13.

Table 13: Status Codes

Status Code ST[2:0]	AGP2.0 Usage	AGP3.0 Usage
000	Low Priority Read	Read (Asynchronous[5])
001	High Priority Read	Reserved
010	Low Priority Write	Write (Asynchronous[5])
011	High Priority Write	Reserved
100	Reserved	Reserved[6]
101	Reserved	Reserved[6]
110	Reserved	Calibration Cycle
111	AGP Master can start PCI Transaction using FRAME# or PIPE#.	AGP3.0 Master can start PCI Transaction using FRAME.

As in AGP2.0, the AGP3.0 read access granularity is 8-bytes. The AGP3.0 address must be naturally aligned to 8-byte boundaries. Again, similar to AGP2.0, the read transfer time on AGP3.0 is in multiples

---

[3] See Table 38 in Appendix A for optional use of these reserved codes.

[4] This encoding is used on the `C#/BE[3:0]` for performing a "Dual Address Cycle" on a `FRAME` based PCI transaction. It is reserved for AGP requests.

[5] The term "asynchronous" differentiates this transaction type from the isochronous transactions described in Appendix A where latency and throughput are deterministic.

[6] See 4.1.6in Appendix A for optional use of these reserved codes.

of the common clock. The core-logic ensures that for read transfers that take less than a common clock cycle, the unused portion of the cycle is padded with meaningless data. The strobes continue to run for the entire cycle, and the byte enables (**C#/BE**) are driven only during the meaningful data portion. The critical data (the first data associated with the requested address) will always be driven first followed by any padding that is necessary. This is the same as AGP2.0 except that the amount of required padding could be 8, 16, or 24 bytes.

2.3.3 Flow Control

The maximum AGP3.0 data rate (AGP8X) is twice that of AGP2.0. However, the signals that are used for flow-control by the AGP3.0 Master and Target, as well as the timing relationships, remain the same as AGP2.0. The primary effect of this change in data rate is in the amount of data buffering required to support the AGP3.0 Interface.

Use of flow control during AGP transactions is discouraged and should be used only during unusual circumstances. This implies that a Master should never commit to a read of a given size if there isn't, under most circumstances, sufficient buffering available to hold the data, regardless of when it arrives. Similarly, on a write, the Target should ensure that it does not attempt to fetch the data unless it has the room to absorb the entire transaction request. A good balance must exist between the number of requests en-queued, and the buffering available to receive it.

AGP2.0 and AGP3.0 provide two methods for flow-control of the data. The first is used during a data transfer transaction, while the second is used to stall the start of a new data transaction. For this discussion these methods are called ***Block-Level flow-control*** and ***Buffer-Full flow-control,*** respectively. While these are new terms used for discussion purposes, they do not represent a change in the flow control mechanisms introduced in AGP.

In addition to the these methods, AGP3.0 continues to support the one cycle Wait State prior to the initial block of data that can be optionally inserted by the initiator of the data transfer. The timing diagrams illustrating this can be found in the AGP2.0 Interface Specification.

2.3.3.1 Block-Level Flow Control for AGP Reads and Writes

The PCI common clock signals **IRDY** and **TRDY** are used for this flow-control. These signals can be used during AGP read and write transactions initiated by the AGP Master. The AGP Target can also use this scheme during the data phases of a Fast Write transaction.

The basic scheme allows the receiver of the data to stall the next "block" of data transfer during a read (receiver is AGP Master) or a write (receiver is core-logic). A "block" consists of four common clock cycles of data, which is 128 bytes for AGP8X, or twice the size of AGP4X. To stall a block of data, the receiver must suppress the assertion of **TRDY** in a pre-defined cycle known as the "throttle point".

Note: *For a description of the throttle point in each transaction, refer to the AGP2.0 Interface Specification.*

Since "Long" transactions are not supported under AGP3.0, the size of a transaction initiated by an AGP3.0 Master can never exceed 64 bytes. Block-level flow control is not needed for AGP3.0 reads and writes since no throttle point exists during these transactions.

2.3.3.2 Block-Level Flow Control for Core-logic Fast Writes

The block-level flow-control is still applicable to AGP3.0 Fast Write transactions initiated by the core-logic. This is because there is no size limit for the data transfer during a Fast Write transaction. As with AGP2.0, the AGP Master must accept the first block of data transfer. Therefore the AGP Master must have at least 128 bytes of buffering. This is double that of AGP2.0 which requires a minimum of 64 bytes.

Subsequent transfers may be stalled on a block-by-block basis with the suppression of **TRDY** at the pre-defined throttle point.
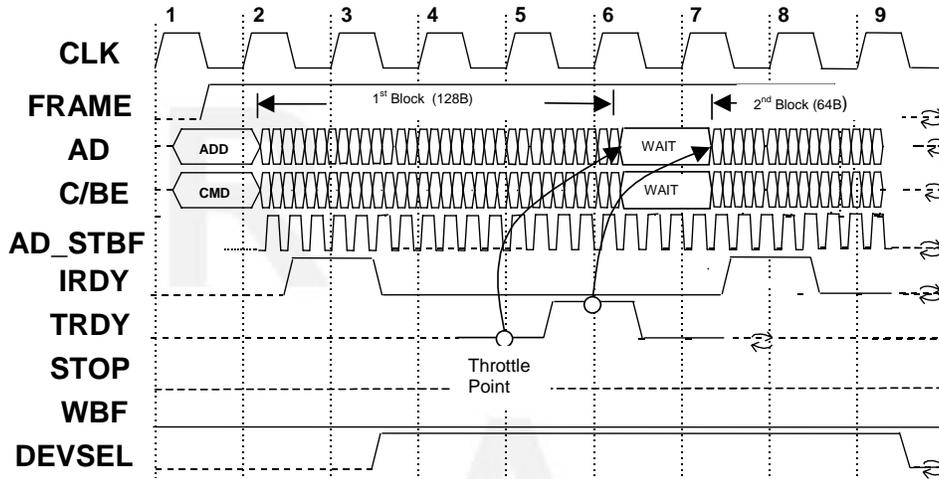
Figure 8: Fast-Write Showing Wait State Insertion

2.3.3.3 Buffer-Full Flow Control for AGP Reads

The Buffer-Full scheme for stalling data returns for AGP reads uses a signal called **RBF**. The assertion of **RBF** by the AGP Master is intended to prevent the start of a new data transfer transaction. However, since there is a delay from the time of **RBF** assertion to the time the core-logic can take action, **RBF** may not be able to stop the transactions (up to a maximum of two) that are already in the pipeline.

This happens if the first data transfer completes in one common clock cycle, and is followed by another data transfer transaction in the immediately following cycle. If **RBF** is asserted in the same cycle as the first transaction, it does not take effect until the third data transfer transaction. The amount of buffering the AGP Master needs to support is dictated by the minimum and maximum request sizes. Table 14 illustrates the difference between AGP3.0 and AGP2.0 buffer requirements.

Table 14: Buffer Size Requirements Based on RBF Flow Control

Interface Type	Min Transfer Size	Max Transfer Size	Min Buffering Needed
AGP2.0	=<16 bytes	=>64 bytes	80 bytes
AGP2.0	=<16 bytes	<64 bytes	Max transfer size+16 bytes
AGP2.0	>16 bytes	=>64 bytes	64 bytes
AGP2.0	>16 bytes	<64 bytes	Max transfer size
AGP3.0	=<32 bytes	64 bytes	96 bytes
AGP3.0	=<32 bytes	<64 bytes	Max transfer size + 32 bytes
AGP3.0	>32 bytes	64 bytes	64 bytes

AGP3.0	>32 bytes	<64 bytes	Max transfer size + 32 bytes

As with AGP, the use of **RBF** is discouraged since it negatively impacts the efficient use of the AGP3.0 interface. An AGP Master should, in general, not en-queue more read requests than the buffering available to receive the requested data.
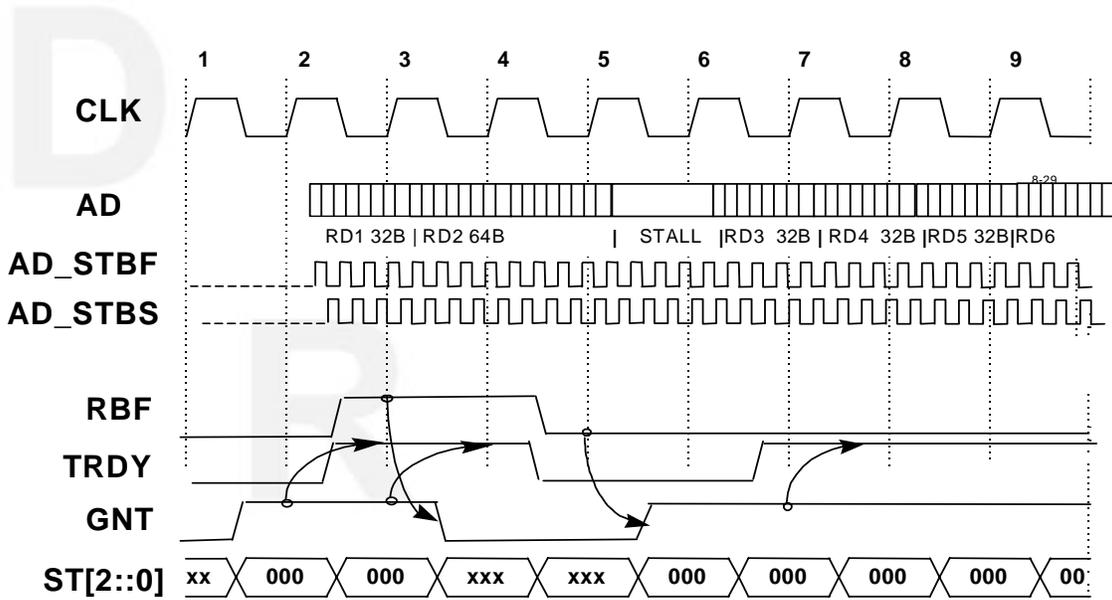
Figure 9: Use of RBF in Read Transaction Control

In Figure 9, **RBF** is not able to stop the two pipelined **GNT**s from completing their transactions. (These are labeled RD1 and RD2 in the diagram.)

RD1 performed a 32-byte transfer followed by an RD2 that performed a 64-byte transfer (too small to initiate **IRDY** flow control). **RBF** managed to block the third data transfer (labeled RD3) from starting by inserting two wait states.

2.3.3.4 Buffer-Full Flow Control on Core-logic Fast Writes

The target of the Fast Writes (AGP device) also has a signal called **WBF** that is used, if required, to prevent the start of a Fast write transaction. The arbiter in the core-logic checks the status of **WBF** before initiating a Fast Write transaction. If **WBF** is asserted at this time, the start of the write is suppressed.

The only change from AGP is that the target of the Fast Writes (AGP Device) must have twice the minimum buffering specified for AGP2.0 data rates before a fast write stall takes effect. This amounts to 160 bytes for AGP3.0 versus 80 bytes for AGP2.0.

2.3.4 Ordering Rule Changes

Ordering rules specify the relationship between concurrently active transactions as they progress to completion through the platform. AGP3.0 follows the same ordering rules as AGP2.0 except for AGP writes (formerly known in AGP as low priority writes).

An AGP write is considered to have *completed* if it has reached a point of global visibility where any future read access to the system memory location targeted by this write gets the modified data. Note that the definition for completion does not require that the actual physical memory location be updated – only that the new data is globally visible.

AGP requires that the completion of low priority writes occur in the same order as the issued requests from the Master. The AGP3.0 specification, however, does not require that the core-logic complete the AGP3.0 write transactions in the same order as the issued requests from the AGP3.0 Master. This implies that a sequence of writes, *AGP_write_memA/AGP_write_memB,* may complete in non-sequential order, such as **memB** being updated before **memA**.

An exception to this rule occurs when a sequence of writes target the same memory location. The Target must ensure that a sequence of writes to the same memory location *is* completed in the same order as issued by the master. For example, in the AGP sequence, *AGP_write_memA/AGP_write_memB/AGP_write_memA,* the Target needs to ensure that the first write to **memA** completes before the second write to **memA.** However, the write to **memB** completes in no particular order with respect to the other two writes.

In AGP3.0, the FENCE request is required for ordering writes. The core-logic must ensure that all AGP3.0 read **and** write requests prior to the issue of a FENCE will complete before any AGP3.0 read or write requests issued after the FENCE.

A summary of the AGP3.0 ordering rules, including those that remain unchanged from AGP2.0, is as follows:

1. AGP Reads complete in issue order on the AGP interface. However, the core-logic may fetch data from system memory in any order.
2. AGP3.0 Writes complete in issue order on the AGP3.0 interface. However, the core-logic may complete the writes to system memory in any order except when the writes are to the same memory location.[7]
3. The core-logic must ensure that an AGP Read following an AGP Write to the same memory address must not pass the AGP write to system memory. Similarly, an AGP Flush must not pass any AGP Write.
4. Unless there is an intervening FENCE, an AGP Write is allowed to pass any previously issued AGP Read, Flush or Write (not to the same address) on its way to completion[8].
5. The core-logic must ensure that no AGP Read, Write[9] or Flush request that follows a FENCE, will pass any AGP Read, Write[9] or Flush request that precedes it.
6. PCI transactions initiated by the AGP Master must follow PCI ordering rules for completion. There are no completion ordering requirements between PCI transactions and AGP transactions.
7. Finally, AGP Fast Writes must complete on the AGP interface in the same order they are issued by the processor.

---

[7] This is a change from AGP

[8] A request to system memory is considered "completed" when it reaches the point of "global visibility" where all requesting agents have access to the latest version of the data.

[9] This is a change from AGP.

2.4 Platform Architecture Differences

This section deals with changes to AGP that have platform level dependencies. These include support for hardware- enforced coherency, peer-to-peer access models, and system configuration issues.

2.4.1 Hardware Enforced Cache Coherency

The AGP Master performs AGP or PCI transactions that are directed at system memory, with a single address space being associated with all these transactions. In this address space, a contiguous region may be specified where an AGP address is re-mapped to a different physical memory address using a platform defined structure called the Graphics Address Re-map Table (or GART). The re-mapping region is called the AGP *aperture*.

The system memory has cacheable regions that can be kept coherent through hardware and software means. Hardware enforced cache coherency, such as snooping, is the responsibility of the core-logic and all caching agents. The performance of hardware-enforced coherency schemes varies between platforms. The appropriate use of the hardware-enforced coherency feature is the responsibility of the graphics card's device driver.

Hardware-enforced coherency for AGP Master accesses to system memory address space is as follows:

1. The core-logic must implement hardware-enforced coherency on all AGP Master transactions (AGP and PCI types) targeted outside the AGP aperture region.

2. For any AGP3.0 Master accesses inside the AGP3.0 aperture region, hardware-enforced coherency is optional*.* Appendix B: Workstation Programming Model describes the scheme used to implement this optional feature on workstation platforms.

2.4.2 AGP3.0/AGP2.0 Compatibility

AGP3.0 and AGP2.0 use the same connector and signal interface with a few additions. This means that in addition to the configurations allowed in the AGP specification, several new ones are created. The following tables define the various motherboard and card types that will be considered in this section.

Table 15: Motherboard Options

Motherboard Types	Connector Type	Description
AGP 3.3 V Motherboard	3.3 V keyed	Supports only AGP 3.3 V signaling. Available speeds 1x, 2x.
AGP 1.5 V Motherboard	1.5 V keyed	Supports only AGP 1.5 V signaling. Available speeds 1x, 2x, 4x.
Universal AGP Motherboard (UAGP)	Universal (UAGP)	Supports both AGP 1.5 V and 3.3 V signaling. Available speeds 1x, 2x, 4x.
AGP3.0 Motherboard	1.5 V keyed	Supports only AGP3.0 signaling. Additional electrical ID to prevent AGP 1.5 V operation. Available speeds 8x, 4x.
Universal AGP3.0 Motherboard (Universal AGP3.0)	1.5 V keyed	Supports AGP 1.5 V and AGP3.0 signaling. Available speeds 1x, 2x, 4x, in AGP2.0 mode and 8x, 4x in AGP3.0 mode.

Since the AGP3.0 and Universal AGP3.0 motherboards use the 1.5 V keyed version of the AGP connector, 3.3 V AGP Cards cannot be used. This means that 3.3 V signaling is not supported even in a Universal AGP3.0 environment.

Table 16: Graphics Card Options

Graphics Card Types	Connector Type	Description
AGP 3.3 V Card	3.3 V & UAGP	Supports only AGP 3.3 V signaling. Available speeds 1x, 2x.
AGP 1.5 V Card	1.5 V & UAGP	Supports only AGP 1.5 V signaling. Available speeds 1x, 2x, 4x.
UAGP Card	All connectors	Supports AGP 3.3 V and 1.5 V signaling. Available speeds 1x, 2x, 4x.
AGP3.0 Card	1.5 V & Universal AGP3.0	Supports only AGP3.0 signaling. Available speeds 8x, 4x.
Universal AGP3.0 Card	1.5 V & Universal AGP3.0	Supports AGP 1.5 V and AGP3.0 signaling. Available speeds 1x, 2x, 4x, in AGP2.0 mode and 8x, 4x in AGP3.0. mode

The following matrix describes the configuration and operating restrictions. The **X** symbol in a box indicates an illegal configuration and must be excluded either by mechanical or electrical means. A box with a ✔ symbol means the configuration is legal and the selected mode of operation and signaling is indicated in parenthesis.

Table 17: Connector Keying Configuration and Operating Restrictions

MB \ CARD	AGP 3.3	AGP 1.5	UAGP	AGP3.0	Universal AGP3.0
AGP 3.3	✔ (AGP 3.3 V)	X	✔ (AGP3.3 V)	X	X
AGP 1.5	X	✔ (AGP 1.5 V)	✔ (AGP1.5 V)	X Note 2	✔ (AGP 1.5 V)
UAGP	✔ (AGP 3.3 V)	✔ (AGP 1.5 V)	✔ (AGP1.5 V)	X Note 2	✔ (AGP 1.5 V)
AGP3.0	X	X Note 1	X Note 1	✔ (AGP3.0-0.7 V)	✔ (AGP3.0-0.7 V)
Universal AGP3.0	X	✔ (AGP 1.5 V)	✔ (AGP1.5 V)	✔ (AGP3.0-0.7 V)	✔ (AGP3.0-0.7 V)

✍ NOTES:

1. *In these configurations, the mechanical keying on the AGP3.0 motherboard connector does not prevent the AGP2.0 graphics card from plugging into the motherboard. However, the AGP3.0 motherboard will be able to detect the presence of a non-AGP3.0 Card using a newly added signal pin* **GC_DET#** *and use this information to prevent the configuration from functioning improperly. More details on how this works can be found in Chapter 3.*
2. *In these configurations, the mechanical keying on the AGP motherboard connector does not prevent the AGP3.0 graphics card from physically plugging into the motherboard. Since the AGP2.0 motherboard will not look at* **GC_DET#** *to identify an AGP3.0 card, it is the responsibility of the AGP3.0 Card to use the newly defined* **MB_DET#** *pin to detect the non-AGP3.0 motherboard and prevent improper functioning or damage. The Card must be capable of tolerating 1.5V signals from the AGP motherboard. More details on how this works can be found in Chapter 3.*

2.4.2.1 AGP3.0 Mode of Operation

The mode of operation for configurations with AGP3.0 or Universal AGP3.0 motherboards and AGP3.0 or Universal AGP3.0 Cards will be AGP3.0. This mode of operation will be selected automatically by hardware during power-up reset and cannot be changed by software. AGP3.0 Mode implies the following:

1. AGP3.0 electrical signaling scheme is used.
2. The speed of the interface is 8x or 4x.
3. All changes to AGP introduced in AGP3.0 will be enforced.

If selected by the hardware, the AGP3.0 Mode will be in effect even in a Legacy OS environment that supports AGP. However, optional new features introduced in AGP3.0 may only be enabled with appropriate changes to the OS and other software. The details of these changes are found in a separate document. The registers that need to be programmed to use these features are described in Appendix B.

The motherboard and Card use the **MB_DET#**, **GC_DET#**, and **TYPEDET#** pins to determine mode of operation according to Table 18:

Table 18: Selecting Platform Mode of Operation

Motherboard	Graphics Card	MB_DET pin on motherboard	GC_DET pin on Graphics Card	TYPEDET pin on Graphics Card	Mode	Speed of Operation
N/A	AGP3.3V	Don't Care	Open	Open	X	N/A
AGP1.5V/UAGP	AGP1.5V/UAGP	Open	Open	Ground	AGP1.5V	1x,2x,4x
AGP1.5V/UAGP	AGP8X Only	Open	Ground	Ground	Illegal	Not supported
AGP1.5V/UAGP	UAGP8X	Open	Ground	Ground	AGP1.5V	1x,2x,4x
AGP8X Only	AGP1.5V/UAGP	Ground	Open	Ground	Illegal	Not supported
UAGP8X	AGP1.5V/UAGP	Ground	Open	Ground	AGP1.5V	1x,2x,4x
AGP8X/UAGP8X	AGP8X Only	Ground	Ground	Ground	AGP8X	8x
AGP8X/UAGP8X	UAGP8X	Ground	Ground	Ground	AGP8X	8x

The detection of card type and mode of operation is resolved during power-up reset of the system. Once the mode of operation is determined, the signaling scheme cannot be changed. When the motherboard and card mutually decide to operate in AGP3.0 Mode, both set the AGP3.0_MODE in the configuration register AGPSTAT [3]. This bit is '0' for AGP mode of operation.

2.4.2.2 Speed Determination

AGP3.0 specification allows both 8x and 4x speeds of operation while using the AGP3.0 signaling scheme. Note that the 4x mode using AGP3.0 signaling is not the same as the 4x mode when using AGP 1.5V signaling. The latter mode obeys the AGP2.0 specification for signaling and protocol while the former complies with the AGP3.0 specifications. Some key differences include:
- Strobes in 8x signaling are not differential and use only the rising edge as the active edge.
- Assertion levels for various signals have changed due to the termination scheme.
- Protocol changes such as removal of PIPE, High Priority reads, Read Long etc. also impact 4x speed operation while in 8x signaling mode.

The speed selection is based on the signaling mode of operation. As described in the previous section, the status of AGPSTAT[3] register is set by hardware to indicate the signaling mode. When this bit is

cleared, AGPSTAT[2:0] are used to specify the speeds supported as described in the AGP2.0 specification. When AGPSTAT[3] is set to pick the 8x signaling mode, AGPSTAT[2:0] takes on a different meaning, described in Table 19.

Table 19: Supported Speed Description

AGPSTAT[3]	AGPSTAT[2:0]	Signaling Mode	Speed Supported
0	xxx	AGP2.0	See AGP2.0 Spec
1	001	AGP3.0	4x only
1	010	AGP3.0	8x only
1	011	AGP3.0	4x and 8x
1	1xx	AGP3.0	Reserved

The core-logic or graphics card that supports "universal" AGP3.0 must dynamically change the meaning of AGPSTAT [2:0] based on the setting of AGPSTAT [3]. The selection of the speed of operation is done by software, which looks at the supported speeds in the core-logic and graphics card to pick one that is common to both. This speed is programmed into AGPCMD [2:0] as shown in Table 20:

Table 20: Setting Speed of Operation

AGPSTAT[3]	AGPCMD[2:0]	Speed Selected
0	xxx	See AGP2.0 Spec
1	001	4x speed
1	010	8x speed
1	All other codes	reserved

2.4.3 Peer-to-Peer Access

AGP3.0 does not, in general, require the support of PCI Peer-to-Peer accesses. Devices needing to share data do so through shared buffers in system memory. Note that the AGP3.0 specification explicitly relaxes some peer-to-peer requirements of the *PCI-to-PCI Bridge Specification.* Table 21 describes the requirements. In this table, "AGP3.0$_a$" represents one AGP3.0 port in the platform.   A "non-AGP3.0$_a$" port could be another PCI or some other port. The "type" field represents the transaction on the initiator's port.

Table 21: PCI Peer-to-Peer Access

Initiated From	Targeted To	Type	Support
Device(Port == AGP3.0$_a$)	Device(Port != AGP3.0$_a$)	AGP3.0 / PCI Read and Write	**Not Required**
Device(Port == AGP3.0$_a$)	Device(Port == AGP3.0$_a$)	AGP3.0 / PCI Read and Write	**Not Required**
Device(Port != AGP3.0$_a$)	Device(Port == AGP3.0$_a$)	AGP3.0 or PCI Read	**Not Required**
Device(Port != AGP3.0$_a$)	Device(Port == AGP3.0$_a$)	AGP3.0 Fast Write or PCI Write	**Required**

2.5 AGP3.0 Programming

The configuration of registers in the AGP port, for the core-logic and graphics card is described below. This has not changed from AGP2.0. However, there are changes to some of the fields in the NCAPID, AGPSTAT and AGPCMD registers. These are described in this section. Additional changes to support certain workstation specific optional features are described in Appendix B.

AGP requires all core-logic AGP configuration registers to be entirely located in a Host-to-PCI Bridge. AGP3.0 eases this restriction by allowing all of the core-logic AGP3.0 configuration registers to be entirely located in a Host-to-PCI and/or a PCI-to-PCI Bridge function. Splitting the register set across the two bridge types is not allowed.
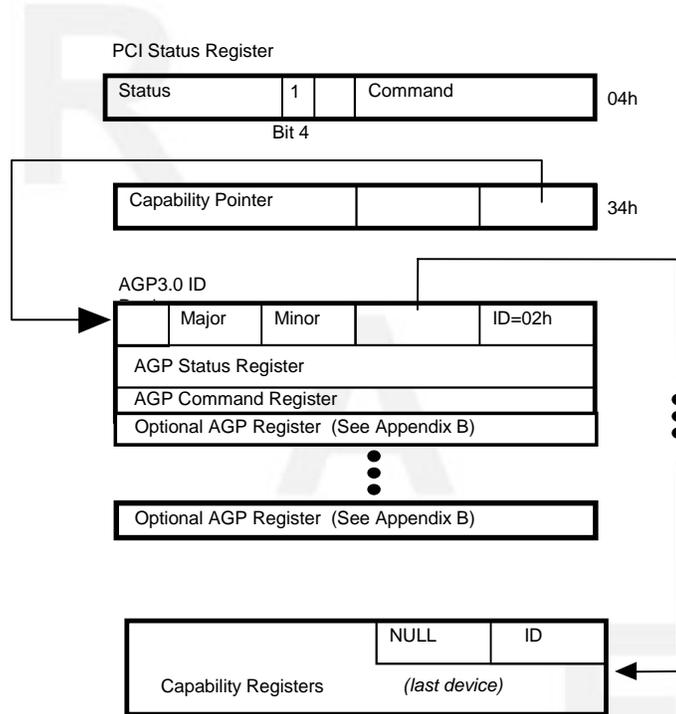
Figure 10: AGP3.0 Configuration Register Space

2.6 Register Table Format

Each Register is specified using the following format:

Table 22: Register Description

Bits	Access	Field	Description
HighBit : LowBit (Zero-origin)	<ACCESS>	Name	Description
Bit (Zero-origin)	<ACCESS>	Name	Description

The **ACCESS** column is of the form:

[READ] – [WRITE] – [DEFAULT] Table 23 defines the value of each sub-field:

Table 23: Sub-field Values

Field	Value	Meaning
[READ]	**R**	Read Allowed
	RZ	Always Read-as-Zero
	R1	Always Read-as-One
	--	Read not allowed
[WRITE]	**W**	Write Allowed
	W1C	Write of any value clears to zero
	IW	Writes Ignored
	MW	Must Write back what is Read (Write-Under-Mask)
	--	Write not allowed
[DEFAULT]	**D'0**	Power-on-Default is Zero
	D'1	Power-on-Default is One
	Dx	Power-on-Default is not specified by AGP3.0
	D'??	Power-on-Default needs to be specified.
	D'xxxx	Power-on-Default is xxxx
	--	Hardwired-Value doesn't need Power-on-default

2.7 Required Master and Target Registers

2.7.1 PCISTS: PCI STATUS REGISTER

Offset: **06h**

Size: 2 bytes

Table 24: PCI Register

Bits	Access	Field	Description
15:5			See the *PCI Local Bus Specification*.
4	R-IW	**CAP_LIST**	If the **CAP_LIST** bit is set, the device's configuration space implements a list of capabilities. The capability pointer is located at 34h.
3:0			See the *PCI Local Bus Specification*.

2.7.2 CAPPTR: CAPABILITIES POINTER

Offset: **34h**

Size: **1 byte**

Table 25: CAPPTR Capabilities

Bits	Access	Field	Description
7:0	R-IW	**CAP_PTR**	This field contains a byte offset into the device's configuration space containing the first item in the *Capabilities List* and is a *Read Only* register.

2.7.3 NCAPID: AGP IDENTIFIER REGISTER

Offset: CAPPTR

Size: 4 bytes

MAJOR AND MINOR REVISION ID NUMBER

Table 26: Major/Minor Revisions

Bits	Access	Field	Description
31:24	RZ-IW	**Reserved**	Always returns 0 on read; write operations have no effect.
23:20	R-IW-D'0011	**MAJOR**	Major revision number of AGP3.0 interface specification this device conforms to. A Value of 3h or greater indicates AGP3.0.
19:16	R-IW-D'0000	**MINOR**	Minor revision number of AGP3.0 interface specification this device conforms to. Range for minor revision for Core Specification registers is 0h-4h. First release will use 0h.
15:8	R-IW-Dx	**NEXT_PTR**	Pointer to next item in *Capabilities List*. Must be NULL for final item in list.
7:0	R-IW-D'CAPID Value	**CAP_ID**	The value of this field is determined as follows: <table><tr><td>Location of Registers</td><td>CAPID Value</td></tr><tr><td>AGP Target Host to PCI Bridge</td><td>00000010</td></tr><tr><td>AGP Target PCI-to-PCI Bridge</td><td>New ID</td></tr><tr><td>AGP Master</td><td>00000010</td></tr></table>

The Major/Minor Revision fields are used to determine register set support by a given target or master. The Minor Revision field is split between Core Specification base features support and Appendix features support. When the AGP configuration registers are located in the host bridge, the CAP_ID remains the same as AGP to maintain backward compatibility with legacy software. A new CAPID is assigned for registers located in a P2P bridge.

The Major and Minor Revision IDs in the NCAPID register are used to inform software of the register set being implemented. Table 27 shows this relationship:

Table 27: Major and Minor Revision ID Selection

Major Revision	Minor Revision	Register Set Support
2h	Don't Care	AGP2.0 Register Support
3h or greater	0h-4h	AGP3.0 Core Specification Register Support
3h or greater	5h-9h	AGP3.0 Appendix Specification Register Support
3h or greater	Ah-Fh	Reserved for future use

The **NEXT_PTR** field contains a pointer to the next item in the list. The **NEXT_PTR** field in a final list item must contain a NULL pointer.

2.7.4 AGPSTAT: AGP STATUS REGISTER

Offset: **CAPPTR + 04h**

Size: 4 bytes

Table 28: AGP Register

Bits	Access	Field	Description
31:24	MST: R1-IW TGT: R-IW-Dx	**RQ**	**TARGET ONLY:** The RQ field contains the maximum number of AGP command requests that can be enqueued to the target. "0" means a depth of 1 entry, while 0xFF means a depth of 256 entries.
23:16	RZ-IW	**Reserved**	Always returns 0 when read; write operations have no effect.
15:13	MST: RZ-IW TGT: R-IW-D'x	**ARQSZ**	**TARGET ONLY**: LOG2 of the optimum asynchronous request size in bytes minus 4 to be used with the target. The MASTER should attempt to issue a group of sequential back-to-back asynchronous requests that total to this size and for which the group is naturally aligned. **Optimum_request_size = 2 ^ (ARQSZ+4)** If ARQSZ is zero, then the target has no recommendation.
12:10	R-IW-Dx	**CAL_Cycle**	**MASTER & TARGET:** Specifies required period for core-logic initiated bus cycle for calibrating I/O buffers.<table><tr><td>CAL_CYCLE</td><td>Period</td></tr><tr><td>000</td><td>4 ms</td></tr><tr><td>001</td><td>16 ms</td></tr><tr><td>010</td><td>64 ms</td></tr><tr><td>011</td><td>256 ms</td></tr><tr><td>100-110</td><td>Reserved for future use</td></tr><tr><td>111</td><td>Calibration Cycle Not Needed</td></tr></table>
9	R1-IW	**Reserved (SBA)**	Always returns 1 when read; write operations have no effect. AGP3.0 devices are required to support side band addressing, even when operating in AGP 2.0 compatibility mode.
8	RZ-IW	**Reserved**	Always returns 0 when read; write operations have no effect.
7	RZ-IW	**Reserved**	Always returns 0 when read; write operations have no effect.
6	MST: RZ-IW TGT: R-IW-Dx	**htrans#**	**TARGET ONLY:** When 0, core-logic will translate host processor accesses through the AGP3.0 aperture using the GART. When 1, core-logic will not send host processor accesses through the aperture. Host translation may be required for certain legacy software support.
5	R-IW-Dx	**OVER4G**[10]	**MASTER & TARGET**: If set, this Master or Target supports addresses greater than 4 GB.
4	R-IW	**FW**	**MASTER & TARGET:** If set to a 1, this Master or Target supports Fast Writes. Fast-Write support is optional for both Master and Target.
3	R-IW	**AGP3.0_MODE**	1 means AGP3.0_Mode. 0 means AGP2.0 Mode. Set on power-up reset. See section 2.4.2 for details. Note that when AGP3.0_MODE = 0, the RATE field (AGPSTAT[2:0]) and DRATE field (AGPCMD[2:0]) in both the master and target must function as defined by the AGP Interface Specification V2.0 for compatibility with existing software.

---

[10] If Target (Core-Logic) sets OVER4G it must support >32bit address PCI cycles generated by the Master using DAC. See PCI Local Bus Specification V2.2 for details. However, support of outbound DAC transactions from the core-logic to the Graphics Card is optional.

Bits	Access	Field	Description		
2:0	R-IW	**RATE**	Data Rate Support (RATE) -		

AGPSTAT[3]	Code	Speed Supported
0	xxx	See AGP2.0 Specs
1	001	4x
1	010	8x
1	011	4, and 8x
1	All other codes	Reserved

2.7.5 AGP_CMD: AGP COMMAND REGISTER

Offset: CAPPTR + 08h

Size: 4 bytes

Table 29: Command Register

Bits	Access	Field	Description
31:24	MST: R-W-D'0 TGT: RZ-IW	**PRQ**	**Master:** The **PRQ** field must be programmed with the maximum number of AGP3.0 command requests the master is allowed to enqueue in the target. "0" means a depth of one entry, while FFh means a depth of 256 entries. **Target:** IGNORED.
23:17	RZ-MW	**Reserved**	Always returns 0 when read; write operations have no effect.
16	RZ-MW	**Reserved**	Always returns 0 when read; write operations have no effect.
15:13	MST: R-W-D'0 TGT: RZ-IW-D'x	**PARQSZ**	**MASTER ONLY**: Programmed based on ARQSZ in TARGET's AGPSTAT[15:13]. LOG2 of the optimum asynchronous request size in bytes minus 4 to be used with the target. The MASTER should attempt to issue a group of sequential back-to-back asynchronous requests that total to this size and for which the group is naturally aligned. **Optimum_request_size = 2 ^ (ARQSZ+4)** If ARQSZ is zero, then the target has no recommendation.
12:10	MST: RZ-IW TGT: R-W-D'000	**PCAL_Cycle**	**TARGET ONLY:** Programmed with period for core-logic initiated bus cycle for calibrating I/O buffers for both master and target. This value is updated with the smaller of the value in CAL_CYCLE from Master's/Target's AGPSTAT. The translations of the encoding are the same as in CAL_CYCLE. PCAL_CYCLE is set to 111 (Calibration Cycle Disabled) by s/w only if both Target and Master have AGPSTAT.CAL_CYCLE = 111.
9	R-W-D'0	**Reserved (SBA_ENABLE)**	Reserved. This must be a read/write bit for software compatibility with AGP interface specification V2.0. AGP3.0 devices are required to support side band addressing. This bit must be set to 0'b when AGPSTAT[3] = 1.
8	R-W-D'0	**AGP_ENABLE**	**Master:** Setting the **AGP_ENABLE** bit allows the master to initiate AGP3.0 operations. When cleared, the master

Bits	Access	Field	Description
			cannot initiate AGP3.0 operations.
			Target: Setting the **AGP_ENABLE** bit allows the target to accept AGP3.0 operations. When cleared, the target ignores incoming AGP3.0 operations. Notes: 1. The target must be completely configured and enabled before the master is programmed. 2. A device can make no assumptions as to the sequence of command field programming, except that **AGP_ENABLE** is the last bit set. Concurrently setting all command fields and the **AGP_ENABLE** bit using a single 32-bit write is also permitted. 3. Any AGP3.0 operations received while this bit is set to 1 will be serviced even if this bit is reset to 0. If this bit transitions from a 1 to a 0 on a clock edge in the middle of an SBA command being delivered in AGP3.0 mode, the command will be issued. The **AGP_ENABLE** bit is cleared by **power-on-reset**.
7	RZ-MW	**Reserved**	Always returns 0 when read; write operations have no effect.
6	RZ-MW	**Reserved**	Always returns 0 when read; write operations have no effect.
5	R-W-D'0	**OVER4G**	**Master:** Setting the **OVER4G** bit allows the master to initiate AGP3.0 Requests to addresses above the 4 GB address boundary. When cleared, the master is only allowed to access addresses in the low 4 GB of the address space. **Target:** Setting the **OVER4G** bit enables the target to accept a Type 4 command and to utilize **A[35::32]** for a Type 3 command.
4	R-W-D'0	**FW_ENABLE**	**MASTER & TARGET**: When set to 1, FW is enabled in Master or Target.
3	RZ-MW	**Reserved**	Always returns '0' on reads. Writes are ignored.
2:0	R-W-D'000	**DRATE**	Data Rate Enable (DRATE) - The setting of these bits determines the data transfer rate. One (and only one) bit in this field must be set to indicate the desired data transfer rate. The same bit must be set on both master and target. The encoding assumes AGP3.0_Mode is set in AGPSTAT. **Encoding Meaning** 001 4X Data Transfer Mode 010 8X Data Transfer Mode 100 Reserved

43

2.8 AGP3.0 Connector Pin-outs

Both Universal AGP3.0 and AGP3.0 motherboards use the same AGP1.5V connector. A few additional signals have been defined to take the place of previously reserved pins. Furthermore, the polarity of certain signals is different from AGP. Table 30 contains the updated pin assignments.

Table 30: AGP3.0 Motherboard Connector Pinout

Pin#	B	A	Pin#	B	A	Pin#	B	A
1	OVRCNT#	12V	23	GND	GND	45	KEY	KEY
2	5.0V	TYPEDET#	24	3.3V AUX	Reserved	46	DEVSEL	TRDY
3	5.0V	GC_DET#	25	VCC3.3	VCC3.3	47	Vddq1.5	STOP
4	USB+	USB-	26	AD31	AD30	48	PERR	PME#
5	GND	GND	27	AD29	AD28	49	GND	GND
6	INTB#	INTA#	28	VCC3.3	VCC3.3	50	SERR	PAR
7	CLK	RST#	29	AD27	AD26	51	C#/BE1	AD15
8	REQ	GNT	30	AD25	AD24	52	Vddq1.5	Vddq1.5
9	VCC3.3	VCC3.3	31	GND	GND	53	AD14	AD13
10	ST0	ST1	32	AD_STBF1	AD_STBS1	54	AD12	AD11
11	ST2	MB _DET#	33	AD23	C#/BE3	55	GND	GND
12	RBF	DBI_HI	34	Vddq1.5	Vddq1.5	56	AD10	AD9
13	GND	GND	35	AD21	AD22	57	AD8	C#/BE0
14	DBI_LO	WBF	36	AD19	AD20	58	Vddq1.5	Vddq1.5
15	SBA0#	SBA1#	37	GND	GND	59	AD_STBF0	AD_STBS0
16	VCC3.3	VCC3.3	38	AD17	AD18	60	AD7	AD6
17	SBA2#	SBA3#	39	C#/BE2	AD16	61	GND	GND
18	SB_STBF	SB_STBS	40	Vddq1.5	Vddq1.5	62	AD5	AD4
19	GND	GND	41	IRDY	FRAME	63	AD3	AD2
20	SBA4#	SBA5#	42	KEY	KEY	64	Vddq1.5	Vddq1.5
21	SBA6#	SBA7#	43	KEY	KEY	65	AD1	AD0
22	Reserved	Reserved	44	KEY	KEY	66	AGPVrefcg	AGPVrefgc

✎ NOTE

1. *Reserved pins are only for future use by the AGP3.0 interface specification.*
2. ***IDSEL#** is not a pin on the AGP3.0 connector. AGP3.0 graphics components should connect the **AD16** signal to the **IDSEL#** function internal to the component.*
3. ***TYPEDET#** & **GC_DET#** should be both grounded by AGP3.0 and Universal AGP3.0 cards. These will be pulled up to the appropriate voltage by the motherboard.*

3 AGP3.0 Physical Layer Specification

The *AGP3.0 Physical Layer Specification* defines a set of signaling levels, timing relationships and topologies that support the enhanced performance capabilities of AGP3.0. The physical layer specification provides for full backward compatibility with 1.5 V AGP2.0.

The purpose of this chapter is to set requirements for and make recommendations on DC and AC specifications, maximum and minimum bus lengths, buffer characteristics and location of terminations. The details are provided in the following subsections.

3.1 Overview

3.1.1 Introduction

The AGP3.0 interface is designed to support several platform generations based upon 0.25μ (and smaller) component silicon technology, spanning several technology generations. As with AGP2.0, the physical interface is designed to operate at a common clock frequency of 66 MHz. Its source synchronous data strobe operation, however, is octal-clocked and transfers eight double words (Dwords) of data within the span of time consumed by a single common clock cycle. The AGP3.0 data bus provides a peak theoretical bandwidth of 2.1 GB/s (32 bits per transfer at 533 MT/s). Both the common clock and source synchronous data strobe operation and protocols are similar to those employed by AGP2.0.[11]

To accommodate the downward trend of V_{CC}[12] voltages brought about by continued process evolution and the higher data transfer rates, AGP3.0 specifies a parallel-terminated bus with a fixed nominal voltage swing of 800 mV peak-to-peak. To maintain compatibility with AGP2.0 topologies, the V_{DDQ} connector pin remains as 1.5 V nominal.

AGP3.0 requires impedance compensation for signal integrity purposes. When the component's I/O interface is in "receive mode," the integrated termination device[13] must have already been adjusted to the appropriate impedance. The impedance of the AGP3.0 pull-up driver depends only on the driving component's V_{DDQ} voltage and the impedance value of the termination; the pull-up driver impedance should be set so that the proper output swing voltage is observed.

3.1.2 AGP3.0 Signal Definitions

AGP3.0 is a point-to-point interconnect that contains three types of signals. The two primary sets of signals are the source synchronous signals for data transfer and common clock signals for arbitration and control. The third type of signals (referred to as "asynchronous" in subsequent text) is not bound to the AGP3.0 clocking schemes in any way whatsoever and pertains solely to out-of-band communications. No other topology, beyond a point-to-point interface, is supported.

AGP3.0 specifies only four additional signals to the interface specification of AGP2.0: **GC_DET#**, **MB_DET#**, **DBI_HI**, and **DBI_LO**.

---

[11] See the AGP2.0 Interface Specification for detail.

[12] The V_{CC} voltage refers to the voltage that a given component uses to power its core logic.

[13] The "integrated termination device" may very well include the AGP3.0 pull-down transistor, which also is used for driving the bus when the component is in its drive mode of operation.

Both **GC_DET#**, and **MB_DET#** are of the asynchronous signal type. The **GC_DET#** signal is a static signal supplied from the graphics card to indicate to the motherboard and/or core-logic that the AGP3.0 signaling and protocol are supported. Likewise, the **MB_DET#** signal is a static signal supplied from the motherboard or core-logic to indicate to the graphics card and/or AGP3.0 Master that the AGP3.0 signaling and protocol are supported.

The **DBI_HI** and **DBI_LO** signals are source synchronous signals.
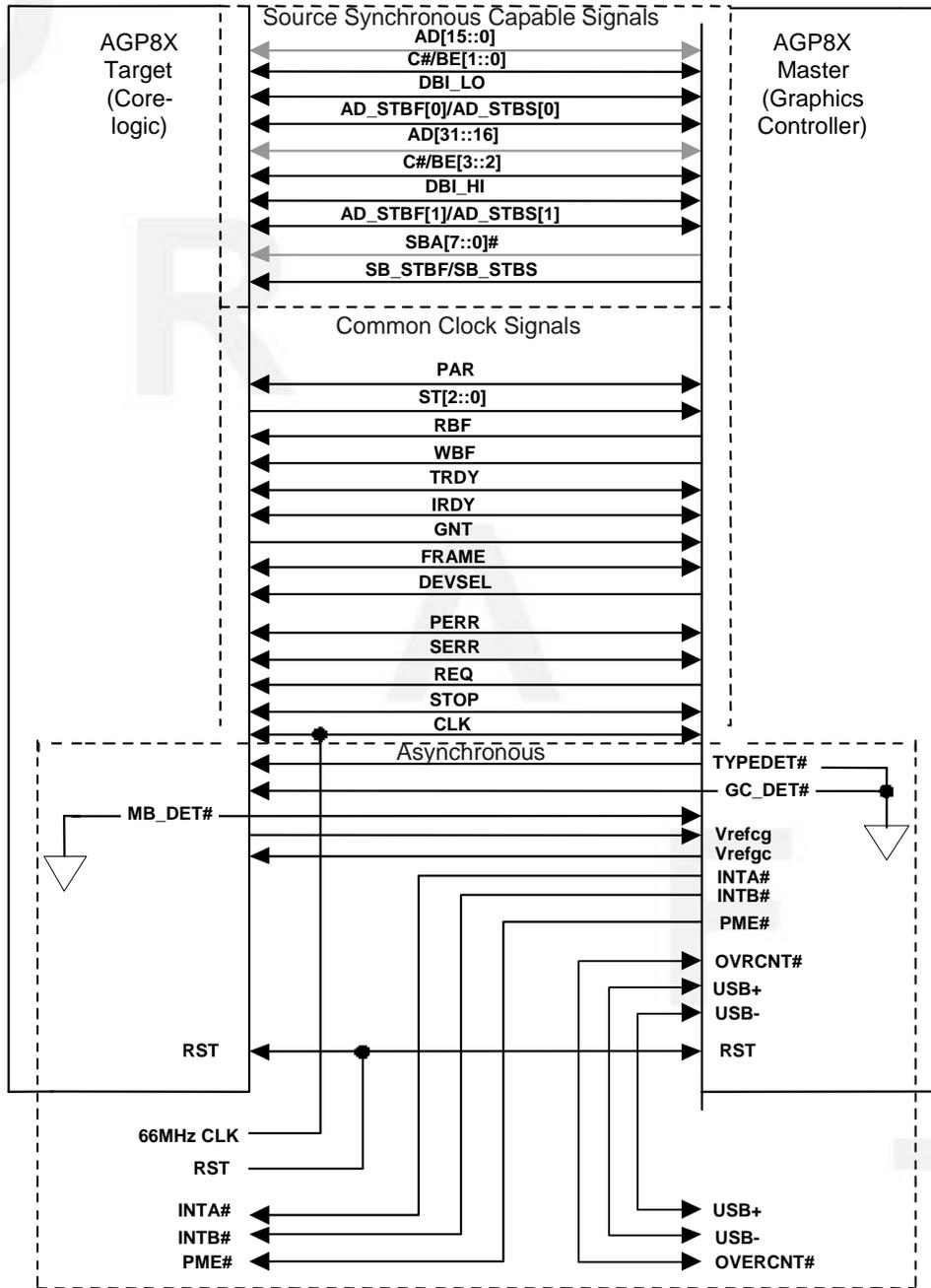
Figure 11: AGP3.0 Logical Diagram

Table 31 lists the various signals of the AGP3.0 interface and identifies the clock domain to which each signal belongs. Signals identified with a double asterisk (**) specification have inverted their signaling

relationship from that of AGP2.0 (e.g. the **SBA** bits are now **SBA**#, indicating that a logic 1 is signaled as a low voltage on the interconnect).

Table 31: AGP3.0 Signals and Associated Clock Domains

Signal Name	Clock Domain			
	Source Synchronous	**Common Clock**	**Asynchronous**	**USB**
**SBA#[7::0]	✓			
**RBF		✓		
**WBF		✓		
ST[2::0]		✓		
AD_STBF[1::0]	✓			
AD_STBS[1::0]	✓			
SB_STBF	✓			
SB_STBS	✓			
CLK		✓		
USB+				✓
USB-				✓
OVRCNT#				✓
PME#			✓	
TYPEDET#			✓	
GC_DET#			✓	
MB_DET#			✓	
**FRAME		✓		
**IRDY		✓		
**TRDY		✓		
**STOP		✓		
**DEVSEL		✓		
**PERR		✓		
**SERR		✓		
**REQ		✓		
**GNT		✓		
RST#			✓	
AD[31::0]	✓	✓		
**C#/BE[3::0]	✓	✓		
PAR		✓		
INTA#, INTB#			✓	
Vrefcg, Vrefgc			✓	
DBI_HI, DBI_LO	✓	✓		

Signals marked as either Common Clock or Source Synchronous in Table 31 are capable of supporting 533 MT/s interface using a 800 mV swing referenced to ground and a 350 mV input

reference voltage.[14] Collectively these signals are referred to as AGP3.0 signals because they share this same signaling scheme. The remainder of this chapter is concerned primarily with this set of signals.

✎ **NOTE**

> *In many cases, AGP3.0 signals have been defined as logically inverted from those of the AGP2.0 Interface specification. This has been done to minimize power consumption, especially in the quiescent state of the interconnect. This is more fully described in Section 2.1.*
>
> *Further, note that the weak pull-up or pull-down resistance that is required for AGP2.0 signals must not be present on the AGP3.0 channel. Therefore, these resistances must be integrated into the motherboard and electrically removed during AGP3.0 operation.*

Electrically, **GC_DET#** is defined identically to the AGP **TYPEDET#** signal, a static signal either left floating or pulled to ground by the AGP3.0 capable add-in card. The **MB_DET#** signal also is electrically similar to **TYPEDET#.** however, it is either left floating or pulled to ground by the AGP3.0-capable motherboard. **GC_DET#** and **MB_ DET#** are used to establish whether or not the channel is to be operated in AGP3.0 or AGP2.0, 1.5V configuration. If **TYPEDET#** is open, the graphics card is indicating that its capability is limited to AGP3.3V.

The **Vrefcg** connector pin is supplied by the motherboard (or core-logic) to provide AGP2.0 Vref or AGP3.0 Vref to the graphics card based on the configuration detected. Likewise, the **Vrefgc** pin is supplied by the Graphics Card to provide AGP2.0 or AGP3.0 Vref to the motherboard based on the configuration detected. The connector signals **TYPEDET#**, **GC_DET#**, and **MB_DET#** are used to establish the channel's configuration. Refer to section "3.4.1.2) Vref Generation" for more detail.

INTA#, **INTB#**, and **PME#** are all 3.3 V "open drain" signals driven by the AGP Master, referenced to the VCC3.3 power supply. These signals' output buffers either need to be 3.3 V tolerant, or a solution external to the component needs to be provided so that acceptable levels are maintained at the interface.

CLK and **RST#** are also 3.3 V signals that may require that special controller input circuitry or dividers be provided to prevent over-voltage or signal distortion at the pin.

USB signals are provided to support a Universal Serial Bus; the reader is directed to the USB specification for further details.

3.2 Transfer Mode Operations

A description of the principal in-band operating modes for AGP3.0 (the so-called "common clock" and "source synchronous" transfer modes) is provided in the following section. Functional characteristics of the AGP3.0 Transfer Modes are quite similar to those of AGP2.0 with all modifications guided by the need to support the 8x data transfer rate of AGP3.0's source synchronous mode.

3.2.1 AGP3.0 Common Clock Transfer Mode Operation

All timings for common clock signals are referenced to a single 66 MHz clock. This clock is required to be common between the two agents at each end of the AGP3.0 channel. This clock may or may not

---

[14] CLK is the single exception, as it is only a 66MHz signal clock. Its electrical characteristics remain unchanged from the AGP interface specification.

be synchronous to some other clock in the system (e.g. the FSB clock). If there is more than one AGP3.0 segment within the system, it must be assumed that different base clocks have been used by each of the segments.

Figure 12 highlights timing relationships for the common clock mode. These relationships are unchanged from that of the AGP interface specification.
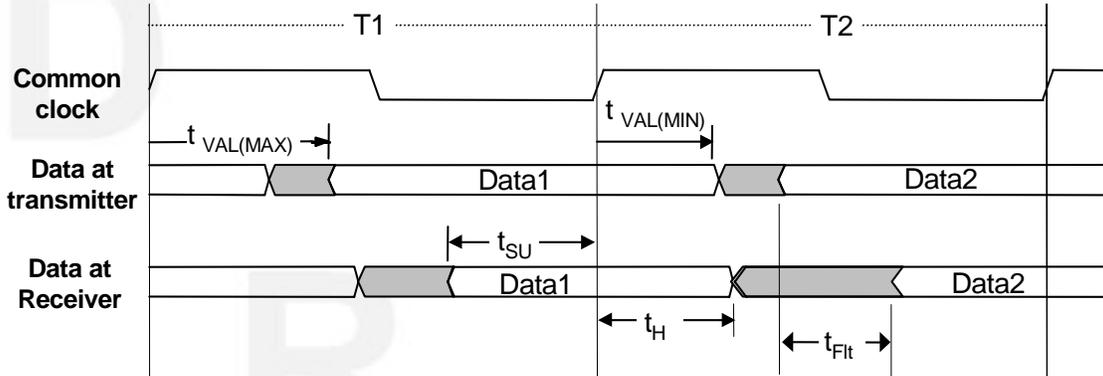
Figure 12: Common Clock Transfer Timings

3.2.2 Source Synchronous Transfer Mode Operation

The source synchronous transfer mode operation provides a mechanism for multiplying the data transfer rate of the **AD[31:0]** data bus signals relative to the common (base) clock frequency. In this mode, 32 bytes of data are transferred during a time equal to a single common clock cycle.

As in AGP2.0, the source synchronous clock mode is implemented as a timing layer *below* the baseline protocol's flow control mechanisms. This timing layer, referred to below as the *inner loop*, specifies timing relationships for the reliable transfer of data from the output latches at the transmitting device to the input latches at the receiving device.

The logical protocol mechanisms operate above this layer, in the so-called *outer loop*, to control the actual transfer of data between the data queues.

A simple model showing these various time domains is shown in Figure 13:
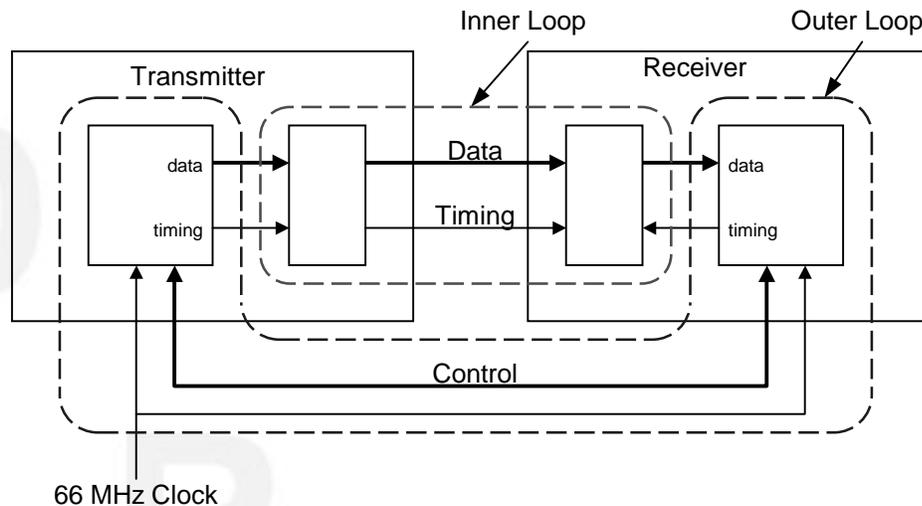
Figure 13: Source Synchronous Mode Time Domain

As previously indicated, the outer loops of both devices operate from a common clock, with all outer loop controls specified relative to this clock. The inner loop timings use additional source synchronous strobe timing signals employed to realize the high data transfer rates.

Source timed strobes -- where the device supplying the data also sources timing signals for use by the receiver -- are used to null out data transport delays at the receiver. The AGP3.0 specification requires that a pair of **AD_STBF/AD_STBS** strobes be supplied for each group of 16 data bits. These source synchronous strobes are to be centered within the output data's valid window, and used by the receiver to directly capture data at the interface. With source synchronous signaling, the limiting parameter is no longer the *absolute* time delay between transmitter and receiver, but is rather the delay *matching* between the strobes and the data bits. Note that strobe usage is different than AGP2.0 (strobe usage is covered in the section on Transmit to Receive Inner Loop).

The timing dependencies between the inner and outer loops are defined by a precise relationship between the strobes and the common clock. This relationship allows for a *deterministic* transfer of data between the inner and outer loops, where these timing dependencies are specified in such a way as to allow implementation flexibility at the receiver. Tradeoffs may be made between the latency through the inner loop, implementation technology, and/or design complexity.

This timing model contains four different time domains, to be detailed in the following sections:

- Transmit/Receive Outer Loop
- Transmit to Receive Inner loop
- Receive Inner to Outer Loop
- Transmit Outer to Inner Loop

3.2.2.1 Transmit/Receive Outer Loop

The outer loop between the AGP3.0 devices uses the common clock as its fundamental timing source. These timings allow for bi-directional control of information transfer between the transmitter and receiver.

The signals in this timing group are the same as those for AGP2.0 and have the timing relationships defined for common clock signals as listed in Section 3.3.1. The signal electricals, however, are as

defined in this specification -- logically inverted from that of AGP2.0 with different V_{OH} and V_{OL} characteristics, etc.

3.2.2.2 Transmit to Receive Inner Loop

Transfer of source synchronous data between transmit and receive inner loop circuits is accomplished using a strobe pair (**AD_STBF/AD_STBS**) that is sent from the transmitter to the receiver.

For AGP3.0, the rising edges of both **AD_STBF** and **AD_STBS** are used to transfer data, with the first data transfer corresponding to the first rising edge of **AD_STBF,** and the second data transfer corresponding to the first rising edge of **AD_STBS**.

Note that in many diagrams that follow, the falling edge of one strobe will be shown as coincident with the rising edge of the other. This relationship is only coincidental and not to be counted on in design implementations; strobes should not be used in a differential manner (i.e., do not base data transfer on the crossover point of **AD_STBF** and **AD_STBS**).

As with AGP2.0, transmit strobe edges are to be positioned near the center of the minimum data valid window, to provide the receiver the greatest possible input data sampling window for the widest range of system timing skew cases. The AGP3.0 interface specification provides both a minimum data valid time *before* the strobe edge (T_{DVb}), as well as a minimum data valid time *after* the strobe edge (T_{DVa}). These transmit strobe/data timings are shown in Figure 14.
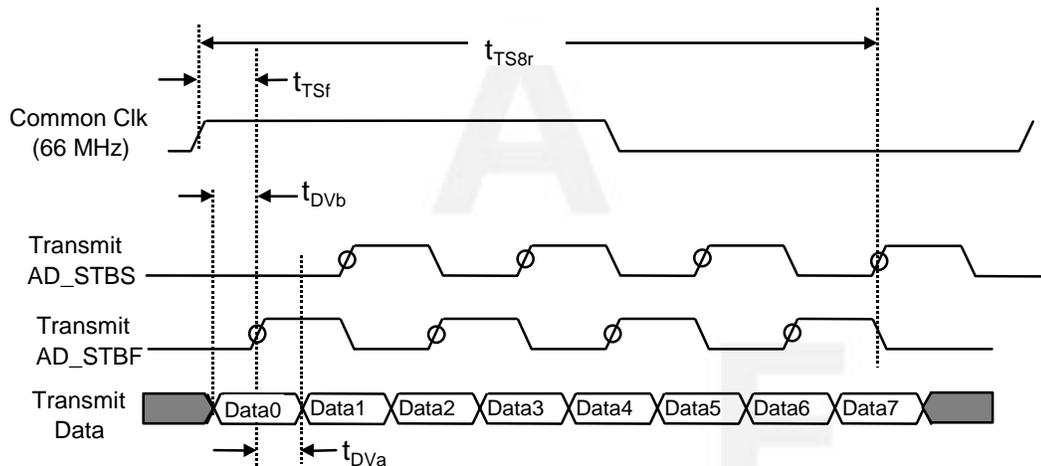
Figure 14: Transmit Strobe/Data Timing for 8X Source Synchronous Timing

The receive **AD_STBF/AD_STBS** strobe inputs are directly used to latch data into the receiving device based on their rising edges. Therefore, a minimum setup (t_{RX_SU}) and hold time (t_{RX_H}), relative to the strobe, is required at the receiver, as shown in the following diagram:
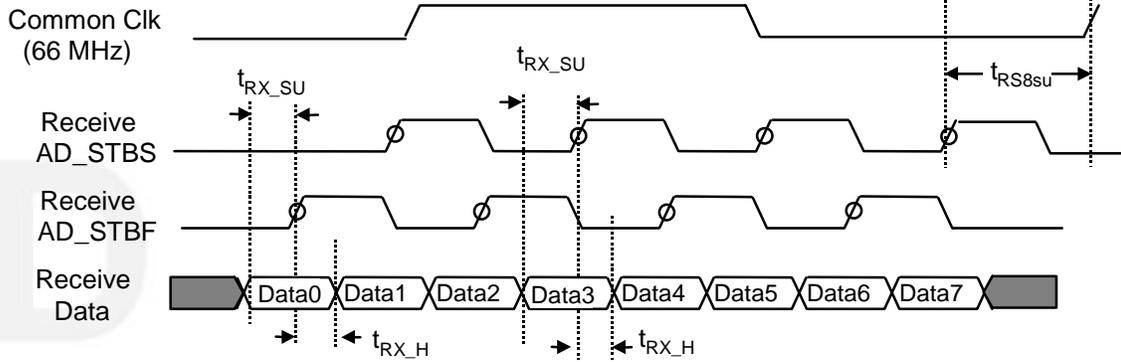
Figure 15: Receive Strobe/Data Timings for 8X Source Synchronous Timing

3.2.2.3 Receive Inner to Outer Loop:

The most complex set of timings are those that address the receiver inner to outer loop relationships. To better understand these timings, a model of the inner to outer loop transfer interface is required. For the following discussion, refer to the receiver transfer diagram shown Figure 16.[15] In the case of the **AD** interface (after the fourth <u>rising</u> edge of the receiver **AD_STBS** strobe), eight DWords of valid data are available to the target core. Data is then transferred from the inner loop to the outer loop based on the common clock. In many respects, the challenge is to define a circuit to reliably affect this transfer for all system conditions.

Figure 15 depicts the possible minimum and maximum **AD_STBF/AD_STBS** relationships at the receiver in the 8X mode. When two consecutive data transfers occur, the first of these is described as occurring in T1. In the minimum [**AD_STBF/AD_STBS**] strobe delay, all eight rising strobe edges are guaranteed to occur during T1. For the maximum strobe delay, however, there can be a variable number (less than four) of rising edges for the **AD_STBF** and **AD_STBS** strobes that cross the common clock boundary.
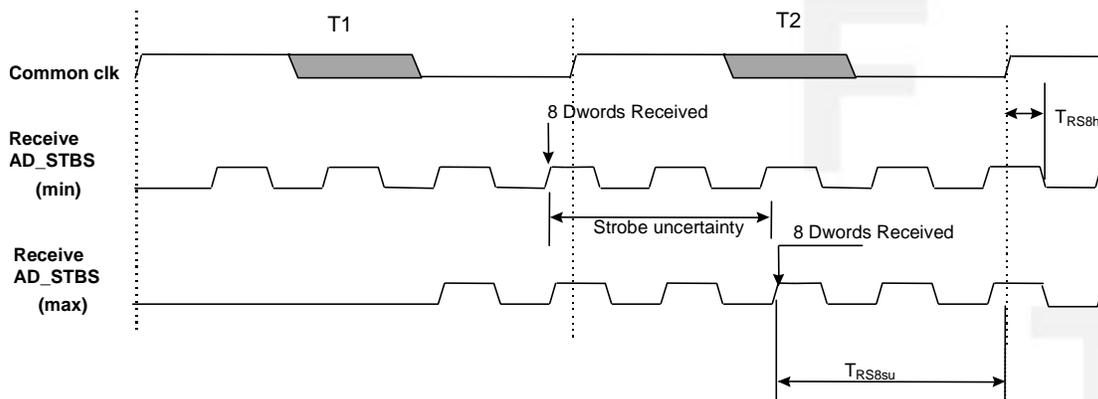
Figure 16: Minimum and Maximum Inner to Outer Loop Receive Timings

---

[15] Note that the diagram shows only the AD_STBS waveform. References that appear to be to the <u>falling</u> edge of AD_STBS are actually to the rising edge of AD_STBF. As previously noted, the strobes are not to be used differentially.

Due to this uncertainty window as to when the first edge of **AD_STBF** will be sent, the earliest *safe* receiver transfer point from the inner to the outer loops occurs at the end of T2. The minimum specification scenario implies that a second set of **AD_STBF/AD_STBS** strobes occurs in T2, when a second set of eight DWords of data is being transferred.

Therefore, to prevent data from being overwritten before the safe transfer point, at least eight stages of latches, for each strobe, must exist in the receiver inner loop input circuitry (i.e., at least two common clock periods of data must be buffered).

The latched data will be transferred to the outer loop into an edge-triggered latch driven by the common clock. The inner loop latched data must be guaranteed to remain stable at the point of transfer, (i.e., on the rising edge of the common clock).

The minimum setup specification on the receive **AD_STBS** to clock (T_{RS8su}) exists to ensure that data from the output of an inner loop latch has a defined setup time to the input of the outer loop's latch. Figure 16 highlighted T_{RS8su} (Figure 14highlights the transmit side equivalent, T_{TS8r}). Likewise, the minimum hold spec on the **AD_STBF/AD_STBS** (T_{RS8h}) exists to ensure that data from the output of an inner loop latch has a defined hold time relative to the input of the outer loop's latch.

As with AGP2.0, the receive **AD_STBF/AD_STBS** strobe specification values were chosen to allow most implementations to transfer data at the *earliest* safe point, or at the end of T2. However, the *actual* transfer point is not specified, only the earliest viable point. An implementation may elect to increase the effective setup time through additional buffering of the inner loop.

3.2.2.4 Transmit Outer to Inner Loop

This section addresses the timing relationship between the outer loop (common clock) and inner loop (8X source synchronous) at the transmitter.

These timings are required to create a deterministic relationship between the inner loop data transfer and the associated outer loop flow control events (i.e., **FRAME, TRDY**). These AGP3.0 timing relationships are very similar to those of AGP2.0.[16]

As with AGP2.0, to guarantee a deterministic relationship between the inner loop data transfer and the corresponding outer loop flow control within the 8X mode, the first **AD_STBF** rising edge is required to occur within the T1 clock period, as seen at the receiver. Proper timings are managed by a minimum specification from the common clock to first **AD_STBF** rising edge (T_{TSf}) and a maximum specification for the last **AD_STBS** rising edge (T_{TS8r}).

---

[16] AGP2.0 defines a parameter termed t_{TS4R} that logically equates to t_{TS8R} for AGP3.0.
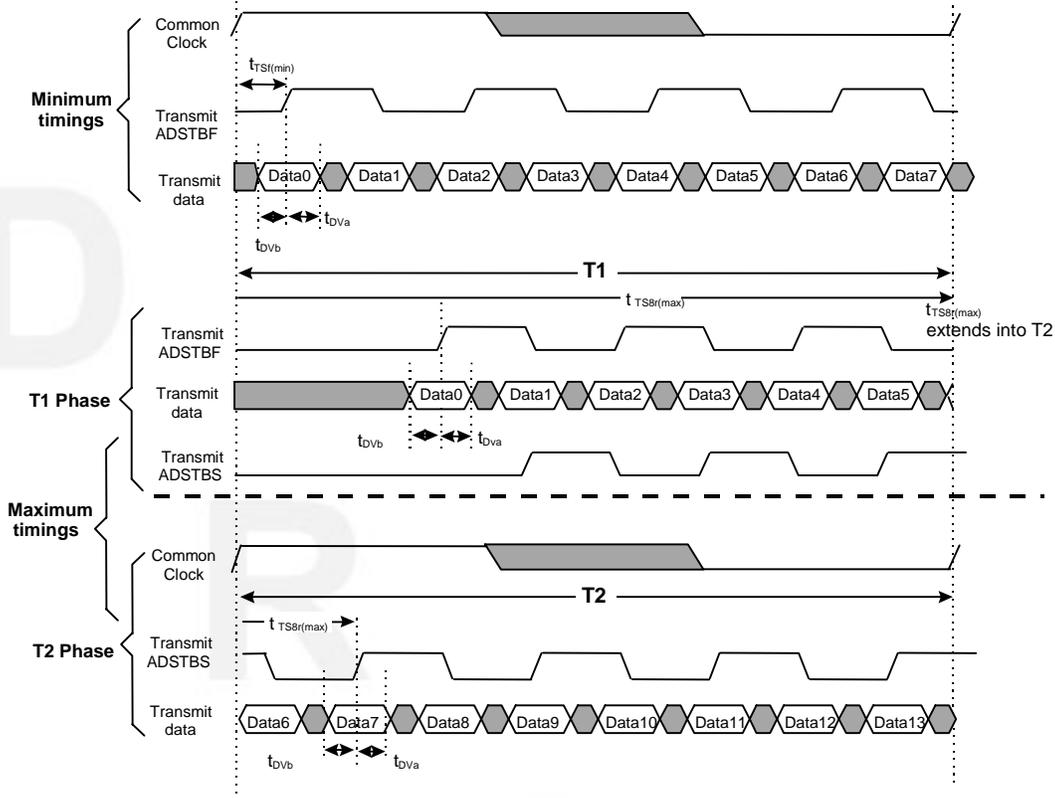
Figure 17: Composite Receive Timing for 8X

3.2.3 Sideband Strobe Synchronization

Data transfer on the **S**ideband **A**ddress **P**ort (**SBA**) is similar to that for the **AD** port. However, there are differences that must be managed (such as proper synchronization of the port). Protocol rules for managing the **SBA** port are unchanged from those of AGP2.0; logical extension for such things as the number of AGP3.0 NOP commands prior to stopping the sideband strobes, restart synchronization, etc., is to be expected.

Also noted earlier is that the **SBA** port is logically inverted for AGP3.0 with regard to AGP2.0.

3.3 Timing Definitions

3.3.1 Common Clock Operations

During common clock operations, the signal timing specifications are based purely on their relationship to the base clock operating at a frequency of 66 MHz. In other words, the signal at the receiver must satisfy the receiver latches' setup and hold times with regard to the common clock.

In most instances, the interconnect distance between two bus agents will be relatively long, and in such cases, the critical speed path is setup time limited and is defined by the following expression:

$$T_{CYC} > T_{VAL(max)} + T_{FLT(max)} + T_{SU} + T_{SKEW} \qquad \textbf{Equation 1}$$

where T_{CYC} is a single common clock period and T_{VAL}, T_{FLT}, T_{SU}, and T_{SKEW} are as defined in Table 32 below. If a very short bus connects the driver and receiver, it is then necessary to make sure that there is not a hold time violation where the timing relationship, which must be met, is the following:

$$T_{VAL(min)} + T_{FLT(min)} - T_{SKEW} > T_H$$ **Equation 2**

where T_{VAL} is the sum of the skews between an agent's input clock, the clock on the driver's output latch, and finally through the output driver. T_{VAL} is composed of the following components:

- Input delay from the system clock through the package ball/pin/pad onto the die
- Clock skew in the internal clock distribution tree
- Clock phase error (for PLL operation only)
- Clock jitter (for PLL operation only)
- Data rise/fall time delay

T_{SKEW} is the sum of all sources of skew between the two agents and includes the following components:

- Clock jitter difference between the separate outputs on the clock generator
- Clock skew between separate outputs on the clock generator
- Trace skew for the two clock traces to the agents due to length or impedance mismatches and crosstalk
- Skew from input loading differences between the two agents

For common clock operations, the sum of the factors affecting clock skew can easily be kept to less than 10% of the total timing budget of 15.0 ns. Table 32 lists the common clock timing budget.

Table 32: Common Clock Timing Budget

Symbol	Parameter	Min	Max	Units
T_{CYC}	CLK Cycle time	15	30	ns
T_{Skew}	CLK Skew between AGP3.0 devices	-	1.0	ns
T_{Val}	CLK to command valid	1.0	5.5	ns
T_{Flt}	Flight time to load	0	2.5	ns
T_{SU}	Setup to CLK	6.0		ns
T_H	Hold from CLK	0		ns

3.3.2 Source Synchronous Operation

In source synchronous operation, the signal timing relationships are defined in two parts: first and foremost is the "inner loop" timing requirements that must be met. Within this loop, common clock parameters such as buffer T_{VAL} and flight time are mostly cancelled out by the source synchronous methodology. The timing specifications of primary concern are those affecting the matching of data to strobes. These include the following effects:

- ISI[17] due to imperfect impedance matching resulting from tolerances on package, board trace, connector, driver and receiver impedances
- Rising/falling edge delay matching
- Vref uncertainty
- Core circuitry noise effects (i.e., clock jitter)
- Package pad-to-pin length variation
- Skew between the data and strobe signals on the motherboard and graphics card (due to trace length mismatch and coupling effects, as well as RPD[18] at the connector)
- Skew between the data and strobe signals on the sending chip (including package SSO[19] and RPD effects)
- Skew between the data and strobe signals on the receiving chip (input capacitance mismatches and timing path mismatches)
- Guard band and other allowances for testing

Source synchronous operation implies the placement of the strobe in "quadrature" with respect to the data. Therefore, the sum of silicon and interconnect skews plus the time required to meet the setup or hold times at the receiver latch must be less than ½ bit time or 937.5 pS. For AGP3.0, the quantitative effects of the interconnect have been evaluated for worst-case parametric variations and are incorporated in the timing skew budget.

The skew of the data relative to the associated strobe is only visible as the specs T_{DVb} and T_{Dva}.

The following timing relationships must be met for source synchronous timing:

$$T_{INTC\_SUS} + T_{RX\_SU} <= T_{DVb} \qquad \textbf{Equation 3}$$

$$T_{INTC\_HS} + T_{RX\_H} <= T_{DVb} \qquad \textbf{Equation 4}$$

Definitions for these timings are provided in Table 33. Note that the interconnect skew targets listed are "pad-to-pad," covering the AGP3.0 channel from the transmitter pad to the receiver pad.

Table 33: Source Synchronous Skew Timing Budget

Symbol	Setup	Hold	Description
Budget	937.5	937.5	Source synchronous setup/hold skew budget
T_{DVb}	527.5		Total setup data valid at the driver/transmitter pins in ps
T_{DVa}		477.5	Total hold data valid at the driver/transmitter pins in ps
T_{RX_SU}	85		Total setup at the receiver in ps
T_{RX_H}		210	Total hold at the receiver in ps
T_{INTC_SUS}	442.5		Total interconnect setup skew from pad to pad in ps
T_{INTC_HS}		267.5	Total interconnect hold skew from pad to pad in ps

---

[17] Inter-Symbol Interference: This includes the effects of impedance mismatches and coupling on a sequence of edge transitions.

[18] Return Path Discontinuity

[19] Simultaneous Switching Outputs

The **AGP3.0 Design Guide** provides an example distribution of first and second order timing budget effects impacting signal skew.

The previous discussion is directed at inner loop timings. Timing specifications between the inner and outer loops must also be closely managed. Specifically, the system must be designed to ensure that data sent in conjunction with T1 is available for capture, in the common (base) clock domain, at the end of T2. This imposes constraints on such design parameters as flight time, and buffer output delays, as well as clock skew. All of the clock skew parameters of both common clock mode and source synchronous mode must be analyzed and controlled to ensure safe data transfer between the clock domains.

3.4 Interface Signaling

AGP3.0 specifies two basic signal types, generally referred to as AGP3.0 signals; these are the source synchronous signals for data transfer and common clock signals for arbitration and control. There is a third type of signal (termed "asynchronous") that is bound to no particular (AGP3.0) clock domain whatsoever and deals solely with out-of-band communications.

3.4.1 AGP3.0 Signaling Details

3.4.1.1 Signaling Levels

AGP3.0 specifies a 0.8 V voltage swing, end terminated, and referenced to V_{SS}, as opposed to AGP2.0, which specified a rail-to-rail 1.5 V series terminated voltage swing.

This change permits a higher data rate and a common signaling voltage, which can be realized for multiple-generations of silicon technology. Figure 18 shows the relationship between the V_{DDQ} and V_{SS} rails and the corresponding output voltage swing.
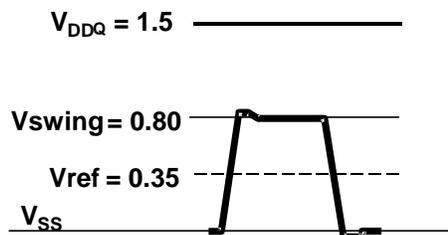
Figure 18: AGP3.0 Voltage Swing

Simulations determined that the best termination value R_{TT} is 50 Ω. If the driver pull-down also serves as the termination, the N-channel device of the driver R_{ON_N} should be set to this same value. The series combination of the receiver's termination R_{TT} and the driver's p-channel device R_{ON_P} determines the voltage swing for a positive transition V_{OH}, the relationship that is shown in Figure 19. For a nominal V_{DDQ} of 1.5 volts and a target signal swing (V_{SWING}) of 0.8 volts, this puts R_{ON_P} at 43.75 Ω nominal.

The target V_{SWING} level is 0.8 volts for a nominal V_{DDQ} of 1.5 volts. The actual V_{SWING} target is proportionally depend on the actual V_{DDQ}. This level is obtained by the proper sizing of the driver pull-

up device against a standard load device of 50 Ω.   The method of setting the driver pull-up to the proper resistance value is implementation specific and is not covered in this specification.
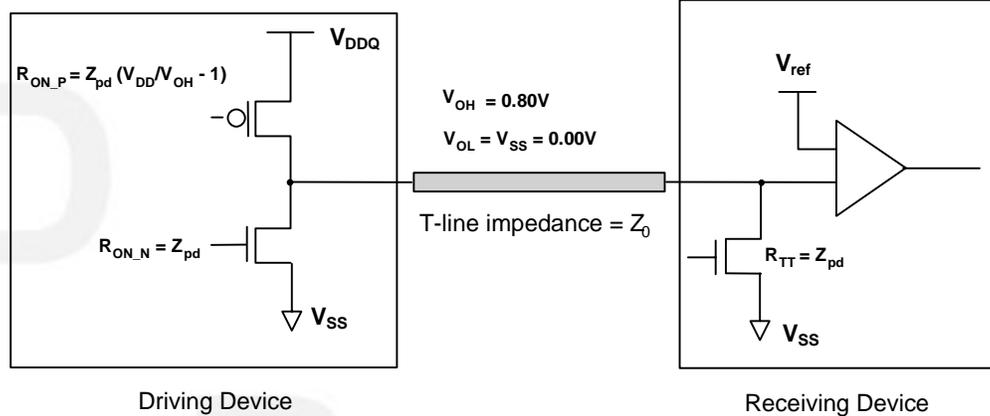
Figure 19: Determination of Device RON Values

3.4.1.2 Vref Generation

Similar to that of AGP2.0, AGP3.0 specifies that a Vref voltage reference signal be <u>source</u> generated for single-ended sensing on the differential inputs of the AGP3.0 receivers. The Vref signal should be implemented with 1% tolerance resistance values from V_{DDQ}, and must be maintained within tolerance while sinking as much as 50 µA of leakage current. Vref is 350 mV for a nominal V_{DDQ} and varies proportionally to the actual value of V_{DDQ}. The generated Vref needs to be properly decoupled to ground at the receiver to manage switching currents; decoupling elements must be as close to the receiver's interface as possible. There must be less than ±15 mV of coupling to the Vref signals (at the receiver pin) from adjacent signal lines. Specific decoupling methods are platform dependent and, therefore, not specified.

AGP3.0 does not define a new set of connector pins for passing along the source generated AGP3.0 Vref data. Instead, the **Vrefcg** connector pin is supplied by the motherboard (or core-logic) to provide AGP2.0 Vref or AGP3.0 Vref to the graphics card based on the configuration detected. Likewise, the **Vrefgc** pin is supplied by the Graphics Card to provide AGP2.0 or AGP3.0 Vref to the motherboard based on the configuration detected. The connector signals **TYPEDET#**, **GC_DET#**, and **MB_DET#** are used to establish the channel's configuration. The rules governing configuration determination are described in Table 18.

3.4.1.3 Driving the Interface to a High Level

When the AGP3.0 interconnect is being driven to a high value the pull-up of the driver is enabled while its pull-down is disabled (see Figure 20).
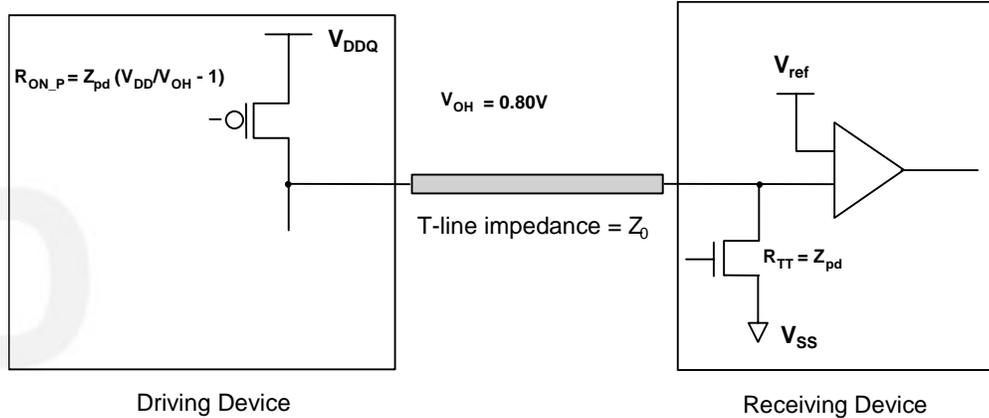
Figure 20: Driving the Interface High

3.4.1.4 Driving the Interface to a Low Level

The AGP3.0 interface has the same make-up in its default (or quiescent) state as when the driver is driving the interface low. In this case the pull-down of the driver is enabled while its pull-up is disabled (see Figure 21).
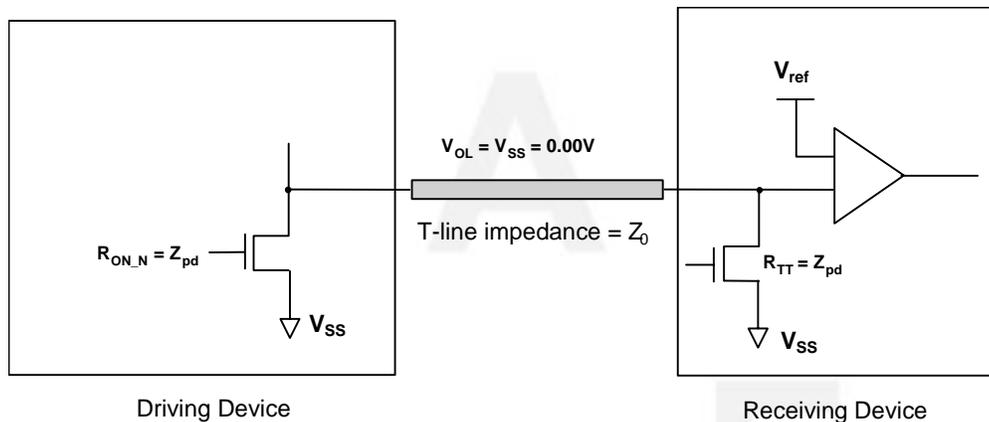
Figure 21: A Quiescent Interface or Being Driven Low

3.4.2 Signaling Details for Non-AGP3.0 Signals

The "asynchronous" signals are signals that are not bound to the AGP3.0 clocking scheme and provide some level of out-of-band communication. They are used to support the Universal Serial Bus, interrupts, power management, and such basics as reset. Several signals within this grouping present special concerns for the component and system designer. Although briefly described in this section, requirements for these signals have not changed from AGP2.0.

The (common) clock signal is also included here, since its requirements also have not changed from that of AGP2.0.

USB signals are provided to support a Universal Serial Bus. Design considerations must be made according to the requirements set forth by the USB specification. Interposing traces for USB need to be designed to 45 Ω $\pm$15% to match the impedance of the USB drivers and cable to preserve signal integrity.

Care must be taken to avoid coupling to any high frequency signals that might cause EMI radiation problems when a cable is attached. Associated power lines should be properly bypassed to decouple noise.

INTA#, **INTB#**, and **PME#** are all 3.3 V open drain signals driven by the AGP Master, referenced to the VCC3.3 power supply. These interrupt and power management signals subsequently may be interfaced to +5 V PCI devices on the motherboard. It is a requirement of the motherboard designer to properly interface the AGP interrupts to the PCI bus. This can be done in several ways. One option is to pull up the PCI interrupts to 3.3 V only, allowing the AGP interrupts to connect directly to the PCI interrupts. Alternatively, the AGP interrupts can be buffered to the PCI bus, thus isolating the 5V environment from the AGP interconnect.

CLK and **RST#** are also 3.3 V signals which may require that special controller input circuitry be supplied, or dividers provided in the receiver or on the platform, to prevent over-voltage or signal distortion at the pin.

3.5 System Topologies and Specifications

As with AGP2.0, the AGP3.0 interface is a logical point-to-point[20] network. The AC timings and electrical loading on the AGP3.0 interface are optimized for one active host component on the motherboard and one active AGP3.0 agent. More than two physical connections to the interconnect are not recommended since there is little timing margin for the added load, stubs and other signaling discontinuities. If the interconnect is comprised of more than two loads and/or branching in the topology, it is the system designer's responsibility to ensure compliance to this interface specification.

Interconnect and package requirements are explicitly called out in subsequent sections of this specification for the topologies listed later. This specification takes into consideration many concerns including different design cost constraints. For example, lower cost solutions can choose to utilize microstrip interconnect technology over symmetric stripline.

3.5.1 Universal AGP3.0 Topologies

The "Universal" topology (Universal AGP3.0) anticipates a host component that resides directly on the motherboard or that is possibly connected through an AGP connector to an AGP2.0, AGP3.0 or Universal AGP3.0 device.

Associated transmission line routing must meet both AGP2.0 and Universal AGP3.0 routing rules. Routing for AGP3.0 also supports AGP2.0, except that AGP3.0 can support longer stripline trace lengths. AGP3.0 interconnects must be referenced to ground to preserve noise margin. Universal topologies are limited to source synchronous interconnects with a flight time of 2.1 ns. It is expected that most early AGP3.0 designs will target the Universal AGP3.0 topology requirements.

Section 3.5.6 outlines the motherboard and add-in card interoperability options for Universal AGP3.0, AGP3.0, and AGP2.0. For all of the topologies the V_{DDQ} signal is required to be 1.5 V.

---

[20] This means that active communication can only occur between two AGP3.0 agents that reside on the interface, where one agent is referred to as the AGP3.0 target and the other the AGP3.0 Master. The simplest implementation is to have only two devices attached to the bus. The specification does not preclude attaching more than two devices to the interface as long as there is only one active master and one active target. Any other device must not respond to or interfere with the interface operation. More than two devices are not recommended. When more than two devices are attached to the interface, the system designer is responsible to ensure that all requirements of this interface specification are met; since the component and/or add-in card designer has no control on how the devices are used.

3.5.2 Other Topologies

It is possible to design a system with master and target devices both "down" on the motherboard, without an intervening connector. Also, custom systems with a connector other than the standard AGP connector can be built. Both of these cases can benefit from these specifications, however this specification is not meant to cover or constrain either of these configurations.

3.5.3 System Level Definitions

3.5.3.1 Interconnect Layout Definitions

Interconnect layout strongly influences its effective impedance; "effective impedance" incorporates all coupling effects including crosstalk. Furthermore, the sensitivity of effective impedance to simultaneous switching (its magnitude and direction) of nearby AGP3.0 traces is greatly influenced by the relationship of these traces to their coupled power and ground/reference planes. AGP3.0 requires that the traces be strongly coupled to ground. Stripline[21] and microstrip styled board implementations are shown in the figures that follow. Space and height definitions for Table 34 and Table 35 are shown in Figure 22.
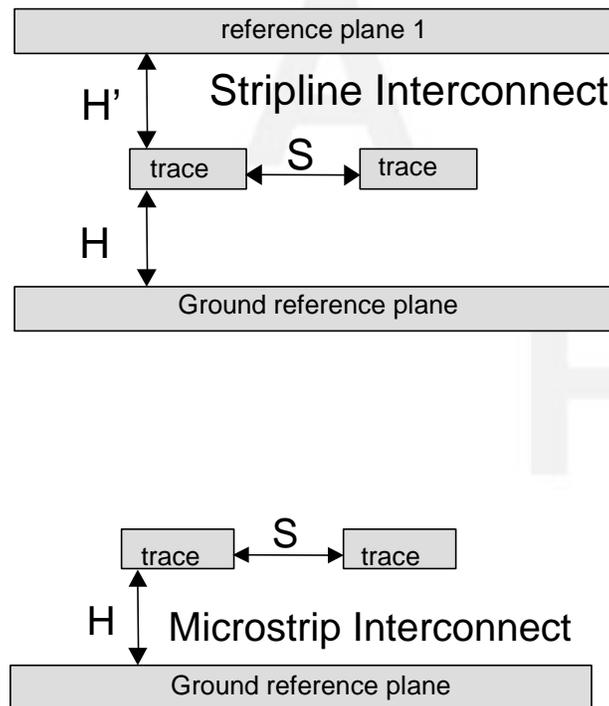
Figure 22: Spacing to Height Definitions for Stripline and Microstrip Implementations

---

[21] Symmetric stripline is defined as H = H'.

3.5.3.2 System (Common) Clock Skew

The maximum allowable, <u>total</u> common clock skew is 1.0 ns. This value includes skew and jitter, which originates on the motherboard and clock synthesizer, and represents the amount of skew observed between the clock inputs to the transmitter's output latches and those of the AGP3.0 Master and target receiver's input latches. Clock skew must be evaluated not only at a single threshold voltage, but at all points on the clock edge that fall within the switching range of the rising edge of the common clock as shown in Figure 23[22].

The total skew is allocated such that 0.1 ns originates from the add-in card routing, and 0.9 ns originates from the motherboard routing and clock synthesizer. The motherboard designer shall determine how the 0.9 ns value is allocated between the board and the synthesizer. To correctly evaluate clock skew, the system designer must take into account clock distribution on the add-in board as specified in the following section.

Common clock skew is measured between the pins of the two AGP3.0 devices at the rising clock edge only; duty cycle is independent of the AGP3.0 interface design, and no specification exists regarding the falling edge of clock.
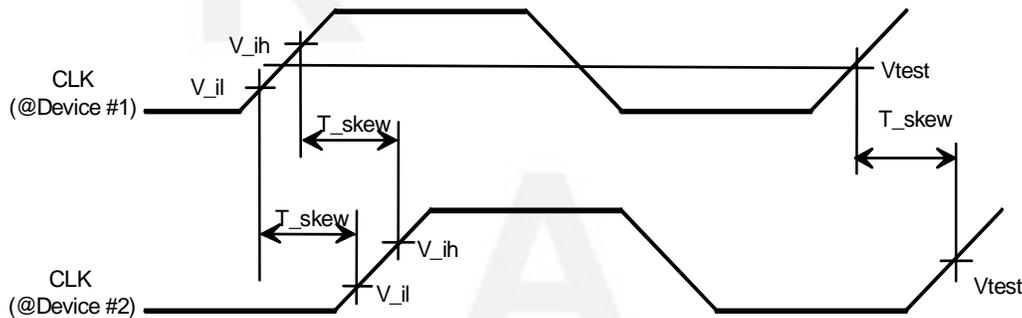
Figure 23: Clock Skew Diagram

NOTE
For systems that employ SSC (spread spectrum clocking), the additional clock skew from the different tracking speeds of the component's PLL must be accounted for in skew calculations.

3.5.3.3 External Pull-ups/Pull-downs

AGP3.0 signals do not require external resistor pull-ups or pull-downs on the motherboard, since the internal terminations serve this purpose during normal operation. The AGP3.0 devices must ensure that the electrical interface is in its de-asserted state during reset (see section 3.5.7). The asynchronous signals (see section 3.4.2) may require pull-up resistors as dictated by a particular platform design.

Universal platforms that support AGP2.0 and earlier modes need to provide pull-ups and pull-downs as required by the AGP2.0 spec. These sustainers must be integrated into the AGP target device to avoid the stubs and vias associated with external resistor sustainers.

---

[22] The system designer may need to address an additional source of clock skew. This clock skew component occurs between two devices that have clock input trip points at opposite ends of the V_{il} - V_{ih} range. In certain circumstances, this can add to the clock skew measurement as described. Nevertheless, in all cases, actual total clock skew must be limited to the specified number.

3.5.4 Motherboard Level Specification

3.5.4.1 Topology

The AC timings and electrical loading specified for the AGP3.0 interface are optimized for one host component on the motherboard and one I/O component on an add-in adapter card. In the material that follows, the AGP3.0-WS (workstation) topologies listed here consider symmetric stripline interconnect only. AGP3.0-DT (desktop) topologies listed here consider microstrip interconnect only. The WS and DT designations indicate the likely usage of these topologies, but are not meant to constrain them to these system types. Further variations can be conceived based on anticipated cost considerations associated with different platform implementations.

3.5.4.2 System Timing Budget

Timing budget issues for the entire channel have already been addressed in the previous sections. Further details on requirements for the motherboard portion of the channel are addressed in what follows:

Table 34 Motherboard Interconnect Requirements[1]

Parameter	AGP3.0-DT		AGP3.0-WS		Units	Notes
	Min	Max	Min	Max		
Source Synchronous Signals						
Interconnect Length - Stripline			2.0	7.0	inches	2
Interconnect Length - Microstrip	2.0	6.00			inches	2
Interconnect Mismatch – strobe-to-strobe		5		5	mils	3
Interconnect Mismatch – strobe-to-data		25		25	mils	3
Trace Characteristic Impedance (Stripline)			50	62	Ω	
Trace Characteristic Impedance (Microstrip)	54	66			Ω	
Trace Effective Impedance	48	70	48	70	Ω	7
Data-to-Data Spacing	4/1		4/1		S/H	4
Strobe-to-Strobe Spacing	5/1		4/1		S/H	4
Strobe-to-Data Spacing	5/1		4/1		S/H	4
Interconnect Setup Skew		432		432	ps	5
Interconnect Hold Skew		257		257	ps	5
Trace-to-Connector fan-in/fan-out		200		200	mils	10
Pin-to-Trace Breakout		550		550	mils	
Common Clock Signals						
Signal Propagation Delay		1.65		1.65	ns	6
Common Clock Skew		900		900	ps	8
Package Types						
Package Types	FCBGA or OLGA		FCBGA or OLGA			9
Package Trace Characteristic Impedance	48	64	48	64	Ω	8

✎ **NOTES**

1. *All interconnects must be ground referenced.*
2. *Worst-case interconnect skews listed in this table are based on simulations that take into account likely layout topologies and a wide range of interconnect. Listed trace lengths include pin-to-trace breakout and trace-to-connector fan-in/-out.*
3. *This mismatch budget applies to the combined trace lengths of the package <u>and</u> board signals. Further, note that the set of 16 data signals and the corresponding strobes, etc. must be routed on the same PCB layer(s).*
4. *Parameter refers to the Space-to-Height ratio of spacing between signal traces and the height above the associated ground plane. Definitions for microstrip and stripline traces are described in subsequent text.*
5. *Sum of all interconnect skew contributors, including crosstalk, interconnect mismatch, etc., across the entire channel (including package and connector).*
6. *Propagation delay must account for all contributors up to the connector pins.*
7. *Effective impedance incorporates all coupling effects including crosstalk.*
8. *This represents the contribution to skew on the motherboard and includes <u>all</u> causes, not just interconnect.*
9. *Package information is added here for simple reference. Flip-chip packaging is highly recommended. For further details, consult section 3.7 on package types in this chapter.*
10. *This is the fan-in and fan-out length that does not follow the normal trace separation requirements in the connector area on the motherboard and the edge fingers of the graphics card.*

3.5.5 Add-in Card Specifications

Table 35 outlines requirements specific to the add-in card interconnect.

Table 35: Add-in Card Interconnect Requirements[1]

Parameter	AGP3.0-DT		AGP3.0-WS		Units	Notes
	Min	Max	Min	Max		
Source Synchronous Signals						
Interconnect Length - Stripline/Microstrip	1.5	1.5	1.5	1.5	inches	1, 2, 3
Interconnect Mismatch – strobe-to-strobe		5		5	mils	4
Interconnect Mismatch – strobe-to-data		25		25	mils	4
Trace Characteristic Impedance (Stripline)	50	62	50	62	Ω	
Trace Characteristic Impedance (Microstrip)	54	66	54	66	Ω	
Trace Effective Impedance	48	70	48	70	Ω	7
Data-to-Data Spacing	4/1		4/1		S/H	4
Strobe-to-Strobe Spacing	5/1		4/1		S/H	4
Strobe-to-Data Spacing	5/1		4/1		S/H	4
Pin-to-Trace Breakout		500	500		mils	
Common Clock Signals						
Signal Propagation Delay		0.7		0.7	ns	6
Common Clock Skew		100		100	ps	8
Package Types						
Package Types	FCBGA, OLGA or BGA		FCBGA or OLGA			9
Package Trace Characteristic Impedance	48	64	48	64	Ω	

☞ NOTES

1. *All interconnects must be ground referenced.*
2. *Worst-case interconnect skews listed in this table are based on simulations that take into account realizable layout topologies and a wide range of interconnect. Trace lengths include pin-to-trace breakout and trace-to-connector fan-in/-out.*
3. *Add-in-Card may use stripline or microstrip routing. The routing must be strongly ground referenced.*
4. *This mismatch budget applies to the combined trace lengths of the package and board signals. Further, note that the set of 16 data signals and the corresponding strobes, etc. must be routed on the same PCB layer(s).*
5. *Parameter refers to the Space-to-Height ratio of spacing of trace to dielectric thickness. Definitions for microstrip and stripline traces are provided in the previous section.*
6. *Propagation delay must account for all such delays up to edge fingers.*
7. *Effective impedance incorporates all coupling effects including crosstalk.*
8. *This represents the contribution to skew on the add-in card and includes all causes, not just interconnect.*
9. *Package information is added here for simple reference. Flip-chip packaging is highly recommended. For further details, consult section 3.7 on package types in this chapter.*

3.5.6 Motherboard / Add-in Card Interoperability

A significant amount of interoperability is required by AGP3.0 and Universal AGP3.0 configurations. Table 36 shows the matrix of compatible motherboard and add-in cards.

Table 36: Motherboard / Add-in Card Interoperability

Motherboard	AGP2.0 Cards			AGP3.0 Cards	
	3.3	1.5	Universal	Universal AGP3.0	AGP3.0
AGP 3.3	✓		✓		
AGP 1.5		✓	✓	✓	(1)
AGP Universal	✓	✓	✓	✓	(1)
Universal AGP3.0		✓	✓	✓	✓
AGP3.0		(2)	(2)	✓	✓

▮   Indicates that this combination is precluded by position of key in AGP connector

☞ **NOTES**

1   AGP3.0 cards can physically plug into an AGP 1.5 volt or universal connector but will not function. AGP3.0 cards need to protect their inputs if there is a possibility of any damage due to a 1.5 V signal level from the motherboard. AGP3.0 cards can use the MB_DET# signal level to determine if it needs to take necessary precautions. Since the AGP3.0 card does not support the signaling mode of the motherboard component, it must not attempt to respond to any cycle (e.g. PCI configuration cycle). AGP3.0 cards will also need to strap TYPEDET# to indicate 1.5 voltage level, otherwise an AGP Universal motherboard may drive 3.3V to the card.

2   AGP cards can plug into an AGP3.0-only motherboard but will not be powered and will not function. The motherboard must detect that GC_DET# is not asserted and remove power from V_{DDQ} pins.

This level of interoperability requires that all AGP3.0 devices and boards have a superset of the electrical characteristics of all the signaling levels and transfer rates they supports. Note that V_{DDQ} is fixed at 1.5 V for AGP2.0, Universal AGP3.0, and AGP3.0-Only topologies. This includes I/O timings, buffer drive characteristics, input signal clamping, and board layout requirements.

In general, the higher transfer rate electrical interface and board requirements are backward compatible to the lower rates for a particular signaling level. The AGP3.0 interface for a core-logic component on a universal motherboard must meet all electrical requirements up to its maximum capable transfer rate.

3.5.7 Reset Requirements for AGP3.0 Universal systems

In general, reset requirements have changed very little from those of the AGP interface specification. Specifically, the only changes have been to require proper management for motherboard/add-in card interoperability and (for calibration purposes) to formalize the minimum amount of time that must pass after reset is de-asserted (100 microseconds) before any device can initiate any type of transaction.

AGP3.0-specific system/interface reset requirements are as follows:

- Establish if the interface signaling is to be AGP2.0 or AGP3.0 in universal platforms.
- The signals **GC_DET#** and **MB_DET#** should be used to make the necessary determination. **TYPEDET#** must also be consistent with these connector signal settings. Decode details are described in Table 18.
- Both the motherboard and add-in card sides need to establish the interface's mode of operation.
- If the interface is to be AGP2.0:
  - The motherboard (or core-logic component) must force **Vrefcg** to 750 mV.
  - The add-in card (or graphics component) must force **Vrefgc** to 750 mV.
  - The core-logic and graphics components must establish proper drive strengths on their interface upon exiting reset. Interface ownership is as defined for AGP.
- If the interface is to be AGP3.0:
  - The motherboard (or core-logic component) must force **Vrefcg** to 350 mV.
  - The add-in card (or graphics component) must force **Vrefgc** to 350 mV.
  - The core-logic and graphics components must establish proper drive and terminator strengths on their interface upon exiting reset. Interface ownership is as defined for AGP.

To manage interoperability concerns, the following special cases/steps must be handled.

- When an AGP3.0-only card is plugged into an AGP2.0 slot, it must take the necessary precautions to protect its electrical interface from damage. In this configuration, the AGP3.0 card must not attempt to respond to any cycle initiated by the motherboard component.
- When an AGP2.0 card is plugged into an AGP3.0-only motherboard, the motherboard must remove power from V_{DDQ} pins. Further activity on the interface should be suspended.

The drive levels for AGP2.0 and AGP3.0 around **RESET** are given in Table 37 and Table 38. Note that the correct level must be stable well before the end of **RESET**. A Universal AGP3.0 design needs to use the **TYPEDET#**, **GC_DET#** and **MB_DET#** to make the necessary determination of the selected signaling. The controllers may not detect these signals properly during the power up sequence, so they must be able to change the signaling on the interface as necessary during **RESET.**

Table 37: AGP Target Signal State During and After RESET

Signal Name	AGP3.0 Signaling		Signal Name	AGP 2.0 Signaling	
AGP3.0	During Reset	After Reset[1]	AGP2.0	During Reset	After Reset[1]
REQ	PD or Term	Term	REQ#	Pull-up	Pull-up
GNT	PD or Term	Drive Low	GNT#	Pull-up	Drive High
ST[2:0]	PD or Term	Drive Low	ST[2:0]	Pull-up	Drive to determinate state
DEVSEL	PD or Term	Term	DEVSEL#	Pull-up	Pull-up
AD[31:0] C#/BE[3:0] PAR	PD or Term	Drive Low or Term	AD[31:0] C/BE[3:0]# PAR	Pull-up	Driven (as parked master)
DBI_LO	PD or Term	Drive Low or Term	(Not used)	Pull-up (not used)	Pull-up (not used)
ADSTBF[1:0] ADSTBS[1:0]	PD or Term	Term	ADSTB[1:0] ADSTB[1:0]#	Pull-up Pull-down	Float or Driven (as parked master)
DBI_HI	PD or Term	Drive Low or Term	PIPE#	Pull-up	Pull-up
SBA[7:0]	PD or Term	Term	SBA[7:0]#	Pull-up	Pull-up or input disable (Note 2)
SB_STBF SB_STBS	PD or Term	Term	SB_STB SB_STB#	Pull-up Pull-down	Pull-up idle or input disable (Note 2)
FRAME IRDY TRDY STOP	PD or Term	Term	FRAME# IRDY# TRDY# STOP#	Pull-up	Pull-up
RBF WBF	PD or Term	Term	RBF# WBF#	Pull-up	Pull-up
PERR SERR	PD or Term	Term	PERR# SERR#	Pull-up	Pull-up

☞ NOTES

1   *The values under "After Reset" apply to the idle cycles that follow RESET.*

2   *An AGP2.0 graphics device might not use these pins. They should be pulled up by the core logic, or the inputs should be disabled until it is known that the graphics chip is driving them.*

3   *PD, Pull-down = normal AGP2.0 sustainer pull-down (8KΩ nominal)*

4   *Pull-up = normal AGP2.0 sustainer pull-up (8KΩ nominal)*

5   *Term = Termination impedance to ground. This is the standard AGP3.0 termination. A default value must be used prior to the impedance of the terminator being calibrated to guarantee a low logic level.*

Table 38: AGP Master Signal State During and After RESET

Signal Name	AGP3.0 Signaling		Signal Name	AGP 2.0 Signaling	
AGP3.0	During Reset	After Reset[1]	AGP2.0	During Reset	After Reset[1]
REQ	Float	Drive Low	REQ#	Float	Drive High
GNT	Float	Term	GNT#	Float (input)	Float (input)
ST[2:0]	Float	Term	ST[2:0]	Float (input)	Float (input)
DEVSEL	Float	Term	DEVSEL#	Float	Float
AD[31:0] C#/BE[3:0] PAR	Float	Term	AD[31:0] C/BE[3:0]# PAR	Float	Float
DBI_LO	Float	Term	(Not used)	Float (not used)	Float (not used)
ADSTBF[1:0] ADSTBS[1:0]	Float	Term	ADSTB[1:0] ADSTB[1:0]#	Float	Float
DBI_HI	Float	Term	PIPE#	Float	Float or Drive High
SBA[7:0]	Float	Drive Low	SBA[7:0]#	Float	Float or Drive High (Note 2)
SB_STBF SB_STBS	Float	Drive Low	SB_STB SB_STB#	Float	Float or Drive High Float or Drive Low (Note 2)
FRAME IRDY TRDY STOP	Float	Term	FRAME# IRDY# TRDY# STOP#	Float	Float
RBF WBF	Float	Drive Low (Note 2)	RBF# WBF#	Float	Float or Drive High (Note 2)
PERR SERR	Float	Term	PERR# SERR#	Float	Float

✑ NOTE

1. The values under "After Reset" apply to the idle cycles that follow RESET.

2. The sideband signals are optional for AGP2.0 master devices and **RBF** and **WBF** are optional for all AGP masters.

3. Drive Low = Driven to AGP3.0 low value, which may be the same as the Termination.

4. Drive High = Driven to AGP2.0 high value.

5. Term = Termination impedance to ground. This is the standard AGP3.0 termination. A default value must be used prior to the impedance of the terminator being calibrated to guarantee a low logic level.

3.6 Component Level Electrical Specifications

This section provides details of the electrical characteristics for the AGP3.0 interface. I/O buffer design technology to meet these requirements will not be addressed as it is beyond the scope of this specification.

3.6.1 DC Specs

The parameters below list the DC requirements for the common clock and source synchronous modes of AGP3.0.

Table 39: DC Specifications for AGP3.0 Source Synchronous Signaling

Symbol	Parameter	Condition	Min	Max	Units	Notes
V_{DDQ}	I/O Supply Voltage		1.425	1.575	V	
V_{REF}	Input reference voltage	measured at receiver pad	0.34	0.36	V	1
I_{REF}	Vref pin input current	$0 < V_{in} < V_{DDQ}$		±50	µA	1
V_{IH}	Input High Voltage		$V_{REF} + 0.1$	$V_{DDQ} + 0.3$	V	1
V_{IL}	Input Low Voltage		-0.3	$V_{REF} - 0.1$	V	1
V_{OH}	Output High Voltage	Standard 50Ω load to ground. Measured at V_{DDQ} = 1.5 V	0.750	0.850	V	2
V_{OL}	Output Low Voltage	I_{out} = 1500 µA		0.05	V	
C_{DIE}	Input Die Pad Capacitance		1.0	2.5	pF	3
Z_{TERM}	Terminator Equivalent Impedance	V_{OH} = 0.8V Z_{TARG} = 50 Ω	45	55	Ω	4
Z_{PU}	Pull-up Equivalent Impedance	V_{OH} = 0.8V Z_{TARG} = 50 Ω	39.3	48.2	Ω	5

✎ NOTE

1. *AGP3.0 requires differential input receivers to achieve the tight timing tolerances needed for 533MT/s. The nominal value of V_{REF} is 0.350 V at V_{DDQ} = 1.5 V and tracks V_{DDQ} proportionally. The given range can be designed using 1% resistors. The value of V_{REF} is specified to be the center point of the V_{IL}/V_{IH} range.*

2. *V_{oh} is expected to be within 50 mV of Vswing At V_{DDQ} =1.5 volts and will track V_{DDQ} proportionally.*

3. *Signal quality is significantly impacted by large die capacitance at the receiver pad. Furthermore, to meet signal integrity requirements for 533 MT/s signaling, AGP3.0 devices are required to be in high performance packages – modeled with appropriately detailed distributed elements – with low overall inductance and capacitance (i.e., flip-chip BGA with 4 or more routing layers).*

4. *The receiver terminator can also be used as the driver pull-down. To improve signal quality and timing margin, the pull-down/terminator impedance should be as close as possible to its*

specified nominal value across the entire range between Vss and V_{OH} (Refer to Table 46 for complete details).

5. The driver pull-up impedance must be adjusted to obtain an output swing within the V_{OH} specification. More information on driver characteristics can be found in section 3.6.6.1.

3.6.2 AC Measurement and Test Conditions

Because of the high-speed nature of the AGP3.0 interface, test measurements must be performed on a tester with highly accurate edge placements (±100ps or better is recommended). Unless otherwise specified, the reference point for all AC timing measurements is half the voltage swing (V_{OH}) measured with the standard test load. The model for the test structure is shown in Figure 24. Buffer slew rates can be measured using the same setup used for buffer delay determination.
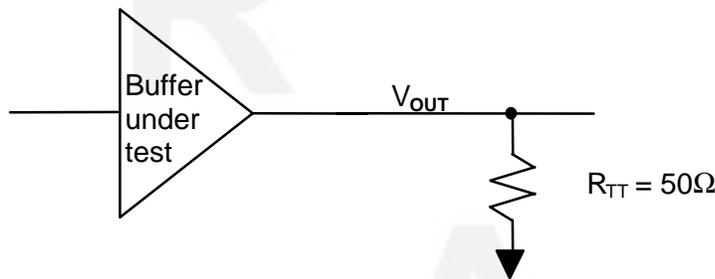
Figure 24: AGP3.0 Standard Buffer Test Load

R_{TT} is set equal to the target buffer impedance (50 Ω as shown in the previous diagram).
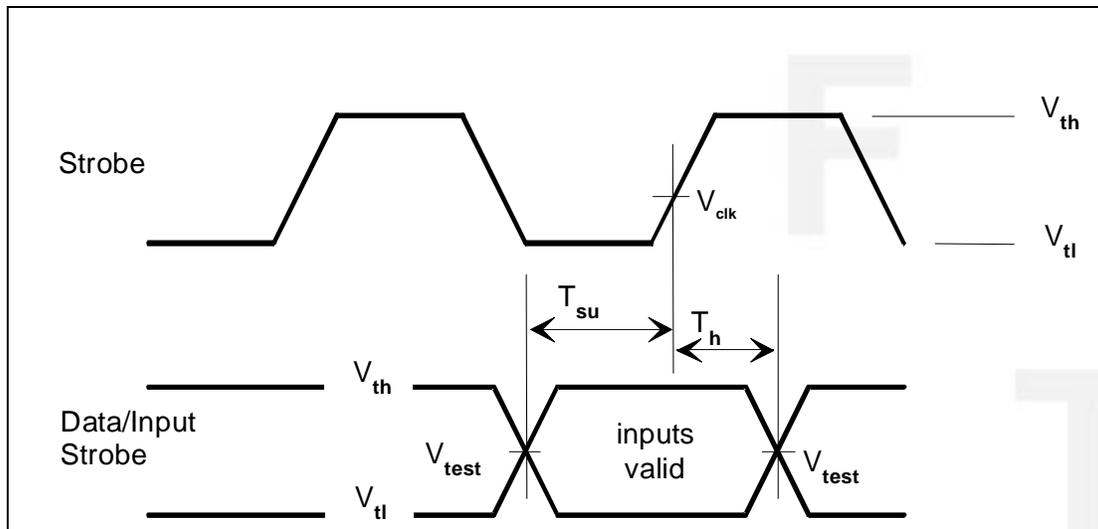
Figure 25: . Input Timing Measurement

Table 40: Measurement and Test Condition Parameters

Symbol	Measure Point	Units	Notes
V_{clk}	V_{REF}	V	1
V_{test}	V_{REF}	V	
V_{th}	$V_{REF} + 0.200$	V	2
V_{tl}	$V_{REF} - 0.200$	V	2
slew rate max	3.5	V/ns	3
slew rate min	2.0	V/ns	3

✎ NOTE

1. V_{clk} is a strobe signal in this example.

2. The input timing measurement test is done with at least 0.1 Volts of overdrive greater than V_{il} and V_{ih}.

3. Outputs will be measured at the die pad and characterized and simulated at the package pins with the loads shown above. Signal slew rate will be measured between V_{tl} and V_{th}.

3.6.3 AC Timings

AGP3.0 timings are specified through two sets of parameters, one set that defines the AGP3.0 common clock operation (for the outer loop control signals), and the second set that defines source synchronous operation. Table 41 and Table 42 provide a summary of the interface timings for 66 MHz and 533 MT/s AGP3.0 operation. The timings are divided between "common clock" for the arbitration signals and "source synchronous" for data transmission and reception.

Table 41: AGP3.0 AC Timing Parameters, 66 MHz Common Clock

Symbol	Parameter	Min	Max	Units	Notes / Comments
T_{Skew}	CLK Skew between AGP3.0 devices	-	1.0	ns	
T_{CYC}	CLK Cycle time	15	30	ns	
T_{HIGH}	CLK high time	6.0		ns	
T_{LOW}	CLK low time	6.0		ns	
T_{Val}	CLK to command valid	1.0	5.5	ns	Into std 50 Ω load to Vss
T_{Flt}	Flight time to load	-	2.5	ns	Includes PCB traces and connector(s)
T_{SU}	setup to CLK	6.0		ns	
T_H	hold from CLK	0		ns	
T_R, T_F	Rise, Fall time	2.0	3.5	V/ns	Measured into std 50 Ω load to Vss

Table 42: AGP3.0 AC Source-Synchronous Timing Parameters

Symbol	Parameter	Min	Max	Units	Notes / Comments
Source Synchronous Transmitter[1]:					
$T_{BIT\_TIME/2}$	One half bit time	937.5		ps	
T_{Dvb}	Data valid before strobe	527.5		ps	At the transmitter pads
T_{Dva}	Data valid after strobe	477.5		ps	At the transmitter pads
T_{INTC_SUS}	Strobe-to-data skew on setup time caused by interconnect effects	0	442.5	ps	Includes (receiver) package effects
T_{INTC_HS}	Strobe-to-data skew on hold time caused by interconnect effects	0	267.5	ps	Includes (receiver) package effects
T_{TSf}	CLK to first AD_STBF rising edge	1.5		ns	
T_{TS8r}	CLK to 4th AD_STBS rising edge		19.5	ns	
T_R, T_F	Rise, Fall slew rate	2.0	3.5	V/ns	See Table 44
T_{R-F}	Rise, Fall slew rate matching deviation		25	%	
Source Synchronous Receiver:					
T_{RX_SU}	Receiver setup margin	0	85	ps	Includes receiver induced strobe-to-data skew on setup margin
T_{RX_H}	Receiver hold margin	0	210	ps	Includes receiver induced strobe-to-data skew on hold margin
T_{RS8su}	8th AD_STBS rising edge before CLK	6		ns	
T_{RS8h}	Receive strobe hold time after CLK	0.5		ns	

NOTE
1. *Measured into standard 50 Ω load to Vss.*

3.6.4 Signal Integrity Requirement for AGP3.0

Table 43 lists the basic signal integrity requirements of the AGP3.0 channel.

Table 43: Signal Integrity Requirements

Symbol	Parameter	Min	Max	Units	Notes
-	Output Slew Rate	2.0	3.5	V/ns	1, 2, 3
V_{OSH_P}	Output positive overshoot above $V_{OH(max)}$		200	mV	1
V_{OSH_N}	Output negative overshoot below ground		-200	mV	1
V_{RB}	Output ringback to Vref		150	mV	1

✎ **NOTE**

1. *Output buffer measurements are made with a standard 50 Ω load to ground (see* Table 40*.)*

2. *Slew rate as measured at the driver's output pad.*

3. *For system designs requiring a minimum of EMI, it is recommended that the output slew rate be held as close to the minimum as possible.*
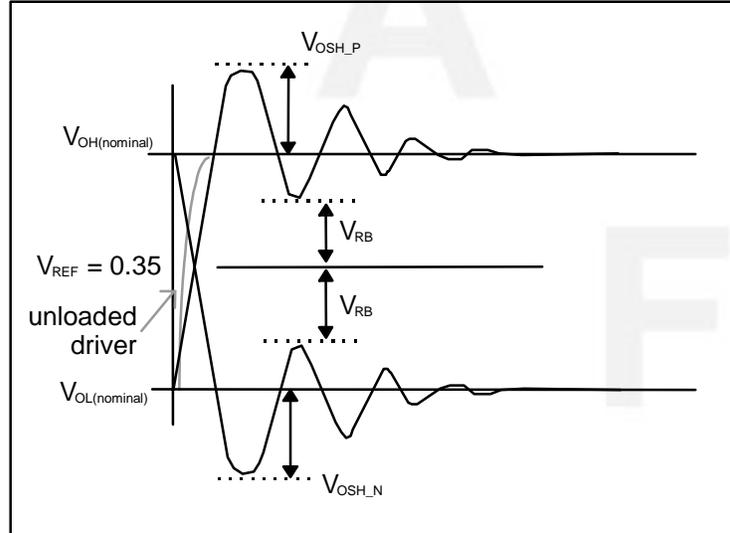
Figure 26: Signal Integrity Requirements

3.6.5 Maximum AC Ratings and Device Protection

All AGP3.0 input, bi-directional, and output buffers must be capable of withstanding continuous exposure to a waveform such as that shown in Figure 26. It is recommended that these waveforms be used as qualification criteria against which the long-term reliability of each device is evaluated. This level of robustness must be guaranteed by design; however, it is not intended that this waveform be used as a production test.

These waveforms are applied with the equivalent of a zero impedance voltage source, driving through a series resistor directly into each AGP3.0 input or tri-stated output pin. The open-circuit voltage (of the voltage source is shown in Table 44) is based on the expected worst-case overshoot and undershoot that is expected in actual AGP3.0 interconnects.

NOTE

The voltage waveform is supplied at the resistor shown in the evaluation setup, not the package pin.
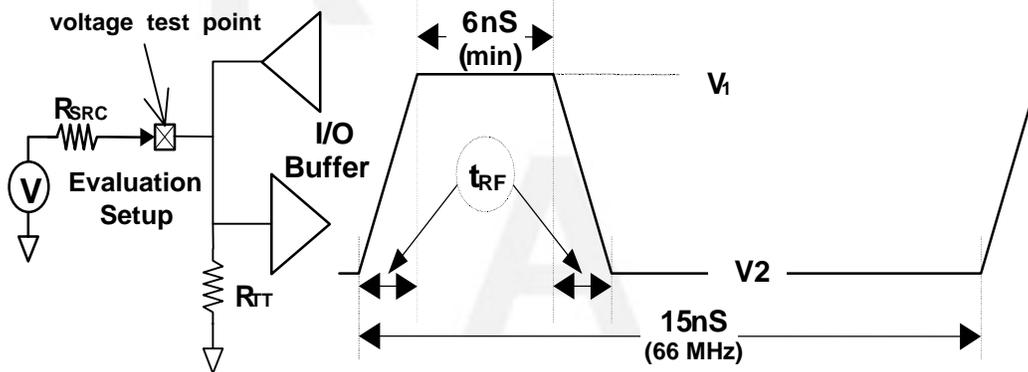
Figure 27: Open-circuit Voltage

Table 44: Parameters for Maximum AC AGP3.0 Signaling Waveforms

Symbol	Parameter	V_{IH} Signaling		Units
		Min	Max	
R_{SRC}	tester source resistance	45		Ω
V_1	Positive-Going Overshoot		1.2	V
V_2	Negative-Going Overshoot	-0.3		V
t_{RF}	Rise fall time	0.72	0.87	ns

3.6.6 Driver and Receiver Characteristics

3.6.6.1 AGP3.0 Driver Characteristics

AGP3.0 is designed for point-to-point communication. The output driver must be able to deliver a voltage swing of 0.8 V (nominal) to the terminated receiver (through the interconnect), with known characteristic impedance Z_0. To reduce signal skew, output buffers are designed to fit in the range specified by the following V/I curves and have characteristics which are shown in Table 45 and Figure 29.

The minimum and maximum drive characteristics of the AGP3.0 output buffers are defined by V/I curves. As with AGP2.0, these curves are interpreted as traditional "DC" transistor curves, except that the curves are more linear than standard CMOS curves with a less pronounced saturation region "knee."

The terminator/pull-down device must be sufficiently linear to suppress reflections that can cause timing skew. This linear region has to extend across the entire range between ground and 0.9 volts. The linearity in the region beyond this point is not as critical, and the V-I curves are relaxed in this region. The shape of the pull-up V-I characteristic also is not as critical. The driver pull-up must ensure, however, that the signal swing is as close to the target **Vswing** of 0.8 volts (nominal) as possible into its load. Making the pull-up more linear does cause the driver to present consistently stable impedance to reflections on the bus, which decreases system noise and increases timing margin.

It is recommended that impedance compensated buffers with slew rate control mechanisms be used in the I/O buffer design to maintain acceptable signal quality. The technological details required, describing the implementation of output compensation or slew rate controls, are beyond the scope of this specification. More information on compensated buffers can be found in the **AGP3.0 Design Guide**. In addition to controlled impedance and slew, the data output buffers may need to be designed with significant "rise and fall" delay matching across process, temperature, and voltage conditions.

Adherence to the V/I curves must be evaluated at worst-case system conditions, including the tolerances on any components used to target the buffer impedance compensation. The minimum curves must also be evaluated at minimum V_{DDQ} and high temperature at the weakest process corner. The maximum curve test points then should be evaluated at maximum V_{DDQ} and low temperature at the strongest process corner.

Table 45: Equations for Current Limits on Pull-down Driver and Terminator

Equation	Description	Condition	Formula[1]	Notes
A	Max Limit	$0 \leq V_{out} \leq V_{DDQ}$	$I_{out} < V_{out} / (0.9*Z_{targ})$	
B	Min Limit	$V_{OL(max)} < V_{out} \leq V_{swing}$	$I_{out} > V_{out} / (1.1*Z_{targ})$	
C	Min Limit	$V_{swing} < V_{out} < 1.1*V_{swing}$	$I_{out} > V_{swing} / (1.1*Z_{targ}) +$ $(V_{out} - V_{swing}) / (1.11* Z_{targ})$	1
D	Min Limit	$1.1* V_{swing} < V_{out} \leq V_{DDQ}$	$I_{out} > V_{swing} / (1.1*Z_{targ}) +$ $(0.1)* V_{swing} / (1.11*Z_{targ})$	1
E	Min Limit	$0 < V_{out} \leq V_{OL(max)}$	$I_{out} \geq 0$	
	Impedance Variation	$0 < V_{out} \leq V_{swing}$	10%	2

 NOTE

1. *The min current limit is relaxed above Vswing to accommodate a standard CMOS transistor saturation characteristic. Termination impedance tolerance above Vswing becomes less important for signal integrity purposes.*
2. *Equations A and B reflect a ± 10% tolerance over <u>the entire set</u> of pull-down and termination impedance, across the range from 0 Volts to Vswing. Any single buffer is required to limit its impedance variation to within a 10% <u>total variation</u>, although it may reside anywhere within the overall 20% of tolerance described by the ± 10% region.*
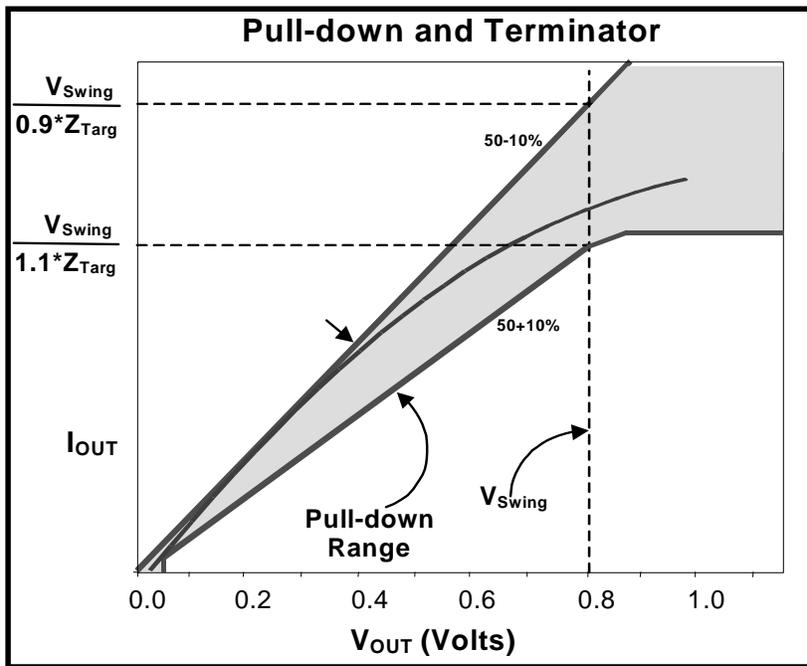
Figure 28: V/I Curve for AGP3.0 Receiver Termination Device
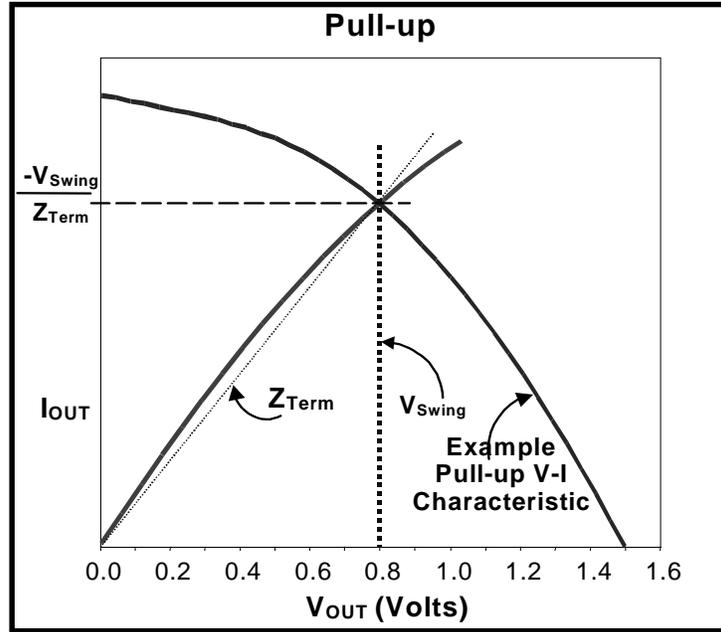
Pull-up

Figure 29: V/I Curve for AGP3.0 Transmitter Pull-up

Table 46: Specifications for AGP3.0 Driver and Terminator

Symbol	Parameter	Condition	Min	Max	Units	Notes
ZTarg	Target Impedance for Pull-down and terminator		45	55	Ω	
Iol	Switching Current Low	0 <Vout≤1.0 V	Eqt'ns B,C,D	Eqt'n A	A	1
Ioh	Switching Current High	Vswing	14.54	17.78	mA	2
Icl	Low Clamp Current	-3<Vin ≤ -1	-25 + (Vin+1)/0.015		mA	
Ich	High Clamp Current	V_{DDQ}+4 > Vin ≥ V_{DDQ}+1	25 + (Vin-V_{DDQ}-1)/0.015		mA	
slewr	Output Rise Slew Rate	Vref ± 200 mV	2.0	3.5	V/ns	4
slewr-f	Output Slew Rate Matching Deviation	Vref ± 200 mV		25	%	5

✏ **NOTE**

1. *Refer to the V/I curves and equations in Table 45.*
2. *There is no specified curve shape or range for the driver pull-up. It should be sized to drive to Vswing with a 50 Ω load to ground. It is recommended that the pull-up be as linear as practical, as this will improve signal timing voltage margins.*
3. *This parameter is to be interpreted as the cumulative edge rate across the specified range, rather than the instantaneous rate at any point within the transition range. Adherence to being within the minimum and maximum parameter is required. Designers are responsible for simulating the complete network to ensure that there are no adverse signal integrity issues related to board or package.*
4. *Mismatched rise and fall slew rates can adversely impact overall t_{co}. For this reason, it is necessary to specify that these slew rates be relatively well matched.*
5. *Slew rate matching calculated as absolute value of 1 - (slew-r + slew-f)/(2*slew-r).*

3.6.6.2 Signal swing (V_{OH}) Accuracy

Signal timing characteristics and quality are highly dependent on the signal swing values observed at the receiver. In general, V_{OH} represents the voltage attained at the receiver input as the signal settles. Designs are expected to maintain V_{OH} at least above 750 mV at V_{DDQ} = 1.5 volts, and preferably near the **Vswing** target value of 800 mV but below 850mV. Signal swing values that are too large may result in unacceptable levels of overshoot, ringback, or settling time.

3.6.6.3 Receiver Characteristics

A differential input receiver is essential for AGP3.0 operations. The voltage reference is specified to be 0.350 V at a nominal V_{DDQ} value. This reference voltage may be generated locally from V_{DDQ}. To reduce the current consumption of the V_{REF} supply, the differential input buffer must be designed with low input leakage current. The combined load on either **Vrefcg or Vrefgc** of all inputs must be less than 50 μA.

The receiver should be designed to handle input swings as little as 250mV to 450mV.

Furthermore, buffer V/I characteristics outlined above are primarily to ensure a proper receiver termination, although they also apply to the driver pull-down.

3.6.6.4 Input Switching

The differential input buffer must be designed to have sufficient gain to convert a small differential input voltage to a full internal CMOS voltage swing without introducing additional skews. Timing analysis indicates that the variance in switch point voltage must be relatively tight for a given signal group.

Switch point voltage is the voltage at which the receiver determines that the input is switching from a one to a zero or vice-versa. Part-to-part and inter-group variations are more relaxed than intra-group variations and are listed in Table 47.

Table 47: Part-to-part and Intra-group Variations

Grouping	Margin Window	Units	Notes
Intra-group	± 30	mV	1
Inter-group	± 100	mV	2, 3
Part-to-part	± 100	mV	2.3

✎ **NOTE**

1. *Within a signal group, all data receivers must switch within ± 30 mV of their associated strobe being sensed on its rising edge.*
2. *Inter-group and part-to-part requirements are such that all receivers must switch within the range of Vref ± 100mV.*
3. *Switch point variations must take into account variations in the external Vref reference signal due to all effects.*

3.6.6.5 Calibration of Driver Pull-up and Pull-down and Receiver Load

To maintain proper receiver load impedance and the proper signal swing (V_{OH}), the driver and receiver will need to be occasionally adjusted to account for changes in operating point. To support this activity, a calibration mechanism has been developed for the AGP3.0 interface. The framework to support this calibration mechanism (boot time requirements, calibration cycles, etc.) is described in section 2.1.3.

3.6.6.6 Component Die Considerations

3.6.6.6.1 PHYSICAL LAYOUT RECOMMENDATIONS

Due to the high-speed signal transmission and fast turn-around time required for the AGP3.0 interface, it is essential that custom layouts be employed throughout the portions of the component encompassing the inner and outer (communication) loops and appropriately managed to optimize routing distances and cell placements. In effect, all logic and circuit elements comprising the AGP3.0 interface may need to be laid out as a single embedded block that bonds directly to the package lead tips.

3.6.6.6.2 DIE PAD DISTRIBUTION

In principle, the AGP3.0 channel is designed for AGP3.0 signals to be referenced to ground throughout the entire channel. Therefore, to achieve the necessary signal return paths, every buffer/driver should have adequate bypass capacitance from V_{DDQ} to Vss as close to the signal pad as possible. This bypass capacitance should meet or exceed 100pF per high-speed output buffer to limit voltage collapse resulting from simultaneous switching outputs.

Furthermore, there must be an adequate number of Vss pads distributed amongst the data and strobe pads. The ratio for the number of die pads is listed in the following section (on packages) since there is a strong relationship between the package type, this ratio, and the resulting signal characteristics. An inadequate distribution of Vss pads can result in detrimental levels of ground bounce and SSO push out.

In addition to providing the necessary levels of bypass capacitance and proper number and placement of Vss pads, the design should be such that strobe signals are placed in the center of their 16-bit data group. This will help to reduce skew due to distribution variance of the input strobe signals along the data buffers.

3.7 Package Considerations

There are two basic package types that have been evaluated and are provided here as a guide. These package types are the flip-chip OLGA or FC-BGA and wirebond BGA packages. This section addresses some of the issues and expectations associated with these packages and with package design concerns in general.

Wirebond packages are not recommended for AGP3.0 signaling because of the large bond wire inductance and crosstalk. Bond wires are usually closely spaced and have no signal return reference plane nearby. The bond wires couple to their neighbors very efficiently causing crosstalk and timing skew. The crosstalk can be reduced only at the expense of adding shielding ground wires around all source synchronous signals.

Simulations have shown that a wirebound suffers from potential power delivery resonance effects. The cause for this is the LC circuit formed by the V_{DDQ}/V_{SS} bond wires and the on-die capacitance of the driver. A certain amount of on-die capacitance is necessary since it decreases signal timing push-out caused by simultaneous switching outputs (SSO). However, the C_{ON_DIE} must be selected such that it does not cause a resonance peak at a frequency where the data signals have significant spectral energy.

Power delivery resonance must be carefully managed regardless of which package type is used. It is sometimes helpful to insert a small resistance in series with the on-die decoupling capacitance to lower the "Q" of the resonant circuit. The resistor value is a balance between sufficiently lowering the resonance and maintaining the bypassing quality of the capacitance.

For both (all) package types it is important to:

- limit reflections due to impedance mismatches
- limit loop inductance and signal return path concerns
- limit trace-to-trace coupling

Package trace characteristic impedance should be matched to that of the terminator target value and board interconnect to as great a degree as possible. Furthermore, package trace lengths should be made as short as is reasonable. Matching trace lengths in the package removes the requirement for compensating for package length mismatch on the board. However, be careful not to increase the inter-signal coupling or loop inductance if matching is attempted. Note that these characteristics directly impact T_{Dvb} and T_{Dva}.

Package traces may be microstrip, embedded stripline, or symmetric stripline as long as the package impedance range is not exceeded. As with board routing, the strobes and corresponding signals within a signal group are to be routed on the same layer(s), and are to be Vss referenced (this is true for the entire channel and not just throughout the package).

The package designer is strongly encouraged to avoid routing signals such that they cross planes or travel over voids for long distances. Bypass capacitance should be provided where possible to account for coupling to signals and planes other than Vss.

Note that regardless of package type, signal current return path continuity must be carefully validated through simulation for the **complete channel:** package, sockets, motherboard, cards, and connectors.

Such simulations should use models that are multi-line and fully coupled since pad-to-pad timings (outlined in section 3.3.2) can be severely impacted by poor design choices with regard to the above items.

3.7.1 Bump-out/Leadframe Requirements

As previously mentioned, the AGP3.0 channel is designed so that AGP3.0 signals are fully and completely referenced to Vss throughout the entire channel. Therefore, to provide the necessary signal return paths, an adequate number of Vss pads must be distributed amongst the data and strobe pads. It is recommended that at least one die ground be supplied for every two AGP3.0 signals. Wirebond will likely require a bond wire per AGP signal for the AC return path and additional wires for proper, low inductance power delivery.

Example bondouts and pinouts are given in the **AGP3.0 Design Guide**.

3.7.2 Pin-out/Ball-out Requirements

All AGP3.0 signal pins must be located to facilitate meeting the card connector with a minimum of package/board vias or trace crossovers to minimize overall trace lengths, crosstalk, mismatched number of layer changes, and trace length mismatches within a given signal group.

In addition, Strobe(s) and Vref signals must be properly grouped within their associated data groups:

AD_STBF[0], AD_STBS[0] with **AD[15::00], C#/BE[1::0]**

AD_STBF[1], AD_STBS[1] with **AD[31::16], C#/BE[3::2]** and **DBI_HI** and **DBI_LO**

SB_STBF and **SB_STBS** with **SBA#[7::0]**

To maintain a strong signal return path along Vss throughout the entire channel it is necessary to provide adequate numbers of Vss pins distributed amongst the data and strobe pins. The ratio (or better) for the number of AGP3.0 signals to Vss balls is recommended.

Ideally, VDDQ pins will be moved away from the region where signals and Vss traces and pins reside. If not, then bypass capacitance should be added very near the package pins between the VDDQ and Vss pins.

3.8 Power Delivery and Distribution

3.8.1 Power Supply Delivery

For AGP, the power supplied to the add-in card for its core supply (V_{CC}) and I/O supply voltage (V_{DDQ}) must be separated on the die, package, and add-in card. Specifically, the I/O buffers must be powered using V_{DDQ}. For AGP3.0, the I/O buffers can be powered by a supply other than $V_{DDQ.}$ However, component designers must be certain to manage noise coupling (such as) between the core and I/O ring and to prevent excessive power supply droop.

As with AGP, the VCC3.3 and V_{DDQ} power supplies must be sequenced such that the V_{DDQ} voltage level is never more than 0.5 V above the level of VCC3.3. Furthermore, V_{DDQ} (supplied to the add-in card through the connector) can never exceed 1.5 V.

The motherboard must connect all power supply pins on the connector as shown in the *AGP Interface Specification* to guarantee proper current delivery and to provide proper AC signal return paths. Likewise, the add-in card should also attach all connector power pins to appropriate power planes on the card for good power delivery and signal returns. In addition, connector power pins should be terminated to the Vss plane with high frequency, low inductance, decoupling capacitors.

Add-in cards must use all ground pins, and any power pins not used must be bypassed to ground on the card with a good quality, low inductance 0.01 µF or larger capacitor. Table 48 lists the voltage ranges for the supplies to the add-in card and the maximum currents that can be supplied via the connector.

Table 48: Add-in Card Power Supply Limits

Symbol	Parameter	Condition	Min	Max	Units	Notes
$V_{DDQ}1.5$	I/O Supply Voltage	I_{MAX} = 2.0 A	1.425	1.575	V	1
VCC3.3	V Power Supply	I_{MAX} = 6.0 A	3.15	3.45	V	
3.3 VAUX	V Auxiliary Supply	I_{MAX} = 0.75 A	3.15	3.45	V	
VCC5	5 V Power Supply	I_{MAX} = 2.0 A	4.75	5.25	V	
VCC12	12 V Power Supply	I_{MAX} = 1.0 A	11.4	12.6	V	

NOTE

1. *AGP3.0 requires no more than 1.0 amp average V_{DDQ} current through the connector. Sufficient bulk capacitance should be provided on the add-in card to provide for higher instantaneous current requirements. This allows each add-in card vendor the ability to determine the correct bulk capacitance. The add-in card should consume no more than 2 amps at any time (from the motherboard).*

   In addition, it is good design practice to limit the AC current through the AGP connector to improve signal integrity. It is necessary to provide additional bulk capacitance on the motherboard near the AGP connector. The V_{DDQ} delivery recommendations can be found in the AGP3.0 Design Guide.

The add-in card is limited to a current increase or decrease on its 3.3 V rail of 2.5 amps in any 500 μs window of time (exclusive of initial power-up). The add-in card current can have an arbitrary waveform within this time as long as the peak-to-peak current change is less than 2.5 amps, and the aggregate slew rate at all pins of the connector is less than 25 A/us. Slew rates faster than this must be handled by add-in card decoupling. Current pulses lasting less than a microsecond cannot be resolved by the motherboard, and consequently, the add-in card vendor must identify if specific cases will be a problem for their design and deal with them accordingly.

The motherboard must respond to any 2.5 amp current step as described above with less than 100 mV of voltage change when measured at the AGP connector pins on the motherboard (averaged at connector pins A9, A16 and A28). The add-in card could increase the absolute magnitude of its current step by over 2.5 amps if it does so in stages that are at least 500 μs apart, giving the motherboard/power supply time to respond.

4 Appendix A: Workstation Enhancements

The following workstation specific changes and enhancements are described in this Appendix. The remaining workstation specific changes are described in Appendix B.

1. Isochronous Operation
2. Fast Write Flow Control Change
3. Synchronization Schemes
4. Fan-Out Bridge Requirements

4.1 Isochronous Mode Operation

Traditional AGP devices can demand up to the maximum bandwidth available over the AGP ports. However, the AGP system does not guarantee to deliver the requested bandwidth, nor does it guarantee transfers will take place within some clearly specified request/transfer latency time. This "best efforts" arrangement works reasonably well for applications requiring low average latency and high average throughput, as long as the device can tolerate an arbitrarily long delay now and then without losing data.

In contrast, streaming applications require the ability to transfer data on a periodic schedule and do so continuously. Further, low cost designs must minimize the amount of request and data buffering. Isochronous data transfer services provide a contract, established during application startup that indicates precisely when data transfers may be requested and sets bounds on when the corresponding data transfers will occur. This greatly alleviates the problem of arbitrarily long delays and places a strict bound on the amount of buffering needed to avoid data loss.

This is done by the system guaranteeing to process a specified number (N) of read or write transactions of a specified size (Y) during each isochronous time period (T). An AGP3.0 device can divide this bandwidth between read and write traffic as appropriate. Further, the system transfers isochronous data over the AGP3.0 Port within a specified latency (L).

Given these parameters, the system guarantees to deliver isochronous bandwidth equal to:

$$BW = \frac{N\,Y}{T}$$

To simplify synchronization with the processor, isochronous write transactions are guaranteed to become visible to the rest of the system without the need for an explicit "flush" operation. Both consumer desktop and professional workstation systems benefit from isochronous support; however, specific platform types may differ as to the number of transactions per period, payload and transaction size and transaction latency.

4.1.1 4x Speed and Isochronous Support

Isochronous operation is not supported when the 4x speed of operation is selected even when operating in 8x signaling mode. The core-logic hardware must ensure that the isochronous operation is disabled when this speed is selected and the Isoch_Support bit in AGPSTAT [17] is cleared.

4.1.2 Contract Parameters

Isochronous parameters are defined in this part of the AGP3.0 specification. The values for other parameters are contained in an AGP3.0 core-logic's PCI configuration registers.

4.1.2.1 Isochronous Time Period

The isochronous time period, T, on all AGP3.0 systems is 1.0 microsecond *(66.667 common clock periods).* All isochronous components *(i.e., master, bridge, core-logic…)* in the system use this same "isochronous period."

4.1.2.2 Isochronous Data Payload Size

AGP3.0 core-logics may support up to four isochronous data payload sizes: 32, 64, 128 or 256 bytes. By default the core-logic specifies a default size based on the internal design of the core-logic. The default size is indicated in an AGP3.0 core-logic PCI configuration register NISTAT.ISOCH_Y. The core-logic must also support all larger sizes. Core-logics may be able to support higher isochronous bandwidth when large payload sizes are used.

AGP3.0 Master devices may use the default size or they may request a larger size. Once selected, all isochronous devices use the same payload size.

4.1.2.2.1 PAYLOAD SIZE VS TRANSACTION SIZE

Isochronous data *payload size* specifies the minimum number of bytes of isochronous data that will be transferred on the AGP Interface without other intervening transactions. The data within the payload has consecutively increasing addresses (naturally aligned to 8-byte boundaries) in AGP memory.

Isochronous *transaction size* specifies the number of data bytes transferred in one isochronous read or write transaction. Isochronous reads, transaction size, and payload size are one and the same. This is because the transaction size supported for isochronous reads are: 32, 64, 128, or 256 bytes.

However, for isochronous writes, only two transaction sizes are allowed: 32 or 64 bytes. Therefore, for payload sizes of 128Bytes and 256Bytes, the Master must generate, without interruption, multiple consecutive isochronous write requests. The rate at which these consecutive requests are generated is one per common clock cycle, which is the fastest allowed by the AGP interface.

Table 49: Transaction Size Vs Payload Size

Type	Payload Size	Transaction Size	# of Consecutive Requests/Payload
Read	32Bytes	32Bytes	1
Read	64Bytes	64Bytes	1
Read	128Bytes	128Bytes	1
Read	256Bytes	256Bytes	1
Write	32Bytes	32Bytes	1
Write	64Bytes	64Bytes	1
Write	128Bytes	64Bytes	2
Write	256Bytes	64Bytes	4

4.1.2.3 Number of Isochronous Transactions Per Period

For each payload size the core-logic supports, it must report the maximum number of transactions it will accept in an isochronous period. This parameter, *Isoch_N,* is specified in the configuration register NISTAT. An AGP3.0 device *(or Fan-out Bridge)* may initiate up to the maximum number of isochronous transactions supported by the AGP3.0 core-logic. At a minimum, the core-logic must support enough transactions to provide at least 128MB/sec of isochronous bandwidth.

Table 50: Minimum Isochronous Bandwidth

Payload Size	Transactions Per Period (Isoch_N)
32 bytes	4
64 bytes	2
128 bytes	1
256 bytes	1

When the payload size is set at 128 or 256 bytes, the core-logic must implicitly allow for 2x or 4x the value of *Isoch_N* for isochronous write transactions. This requires the total number of pending transactions (isochronous and asynchronous) not exceeding the maximum request queue specified by the field AGPCMD.PRQ. It is the responsibility of the Master to ensure that the maximum request queue never overflows. The isochronous contract established by the Master must take this into account when programming the isochronous parameters PISOCH_Y and PISOCH_N in the configuration register NICMD.

4.1.2.4 Transaction Latency

Isochronous latency (L) is defined as the maximum time from an isochronous request to the beginning of a corresponding data transfer (to or from AGP3.0 Target) and is specified in integer multiples of the isochronous period, T[23]. To illustrate, assume that the isochronous latency is 5, and a transaction request begins crossing the AGP3.0 Port 11 common clocks after the beginning of an isochronous period (T). The AGP3.0 core-logic must initiate the corresponding data transfer no later than the 11 common clocks after the beginning of the fifth period following the one containing the request.

Chipsets may guarantee any value for the maximum latency between zero and five isochronous periods (L_{max} = 5 microseconds). The specific value supported by the core-logic is indicated in a PCI configuration register (NISTAT.isoch_L). Figure 30 shows how data transfers may occur in a system where the core-logic latency is $L_{core-logic}$ = 2. The figure also illustrates the required transaction ordering within a read or a write stream, and the lack of order between read and write streams.

---

[23] AGP3.0 does not require the core-logic to indicate its minimum latency.
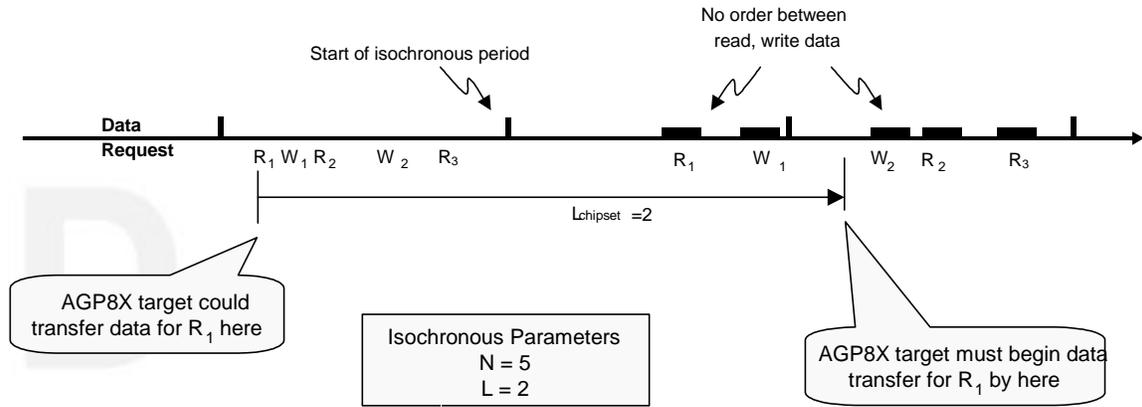
Figure 30: Isochronous Latency Determines Transfer Occurances

Figure 31 shows common values for the maximum number of transactions, latency and payload size for various computer platform classes.

Platform Classes	BW	N	L	Y	Usage
Perf WS	640 MB/s	5	10	128	2 HDTV streams
Entry WS	384 MB/s	3	10	128	1 HDTV stream
DT Enthusiast	320 MB/s	5	2	64	Video editing
DT Consumer	128 MB/s	2	2	64	Video capture

Figure 31: N, Y, and L Values Determine Platform Isochronous Class-level Service

Allowable choices of N, L and Y afford core-logic designers considerable freedom to adapt products to the needs of their markets.

4.1.3 Transaction Ordering

There is no ordering between the completion of isochronous memory transactions and the completion of any other memory traffic in the system. Further, there is no ordering between completions of isochronous read and write transactions.

4.1.3.1 Read Completion Order

AGP3.0 core-logics return isochronous read data in the same order as the corresponding isochronous read requests. The underlying memory read operations might be performed in an order different from the isochronous read request order.

An AGP3.0 memory read transaction is complete when all of the data have been returned to the device from system memory.

4.1.3.2 Write Completion Order

AGP3.0 core-logics transfer isochronous write data across the AGP3.0 Port in the same order as the isochronous write requests; however, this does not constitute completion of the write transaction. An

AGP3.0 "memory write" transaction is complete when the data is globally visible. There are two isochronous write transaction types that differ in their completion order.

Unfenced isochronous write transactions

Unfenced isochronous write transactions may complete in *a <u>different</u>* order from the corresponding unfenced write requests and <u>*cannot*</u> be relied upon for producer/consumer synchronization.

Consider two <u>*unfenced*</u> isochronous write operations to the same target memory address:

<u>*Unfenced*</u> **Isochronous Write A, <u>*Unfenced*</u> Isochronous Write B. B <u>*is not required*</u> to overwrite A**

Chipsets that reorder unfenced write transactions memory completions may be able to sustain higher overall memory performance. AGP3.0 core-logics that do not reorder unfenced isochronous write operations are allowed to substitute fenced writes instead.

Fenced isochronous write transactions

Fenced isochronous write transactions complete in memory after all previously requested write isochronous transactions (of either kind) are completed.

Consider two isochronous write operations to the same target memory address, the second transaction being a <u>*fenced*</u> isochronous write:

Isochronous Write A, <u>*Fenced*</u> Isochronous Write B. B will overwrite A

Fenced write transactions provide a convenient means for synchronizing an AGP3.0 device with the host processor using the producer/consumer model. The AGP3.0 Master device may write a block of data, then write an isochronous synchronization flag in system memory using a fenced write transaction. Once the flag becomes visible, the data is guaranteed to be visible. Isochronous data may be written with fenced or unfenced write transactions; however, completion flags must be written with a fenced write transaction.

Synchronizing the processor with an AGP3.0 Isoch. Write could proceed as follows:

AGP3.0 device writes data *(unfenced isoch write transaction)*
AGP3.0 device writes flag *(fenced isoch write transaction)*
AGP3.0 signals processor

Processor enters polling loop in response to signal
Processor breaks out of polling loop when flag comes true
Processor consumes data

Signaling the processor could be done via interrupts, or done implicitly by means of a periodic task scheduling service in the operating system. Since both signaling methods are unordered with respect to the arrival of data, the polling loop prevents premature processing of the data.

4.1.4 Guaranteed Global Visibility

An AGP3.0 core-logic must make isochronous write data globally visible to the system without the need for an explicit flush operation. This ensures that write data cannot remain invisible inside the core-logic for an indefinite amount of time.

4.1.5 Coherent Isochronous Requests

An isochronous request must not target coherent AGP3.0 address space. Coherent address space is defined as any region in the AGP3.0 address space where the core-logic enforces cache coherency in hardware. Any isochronous request into such a space will result in a platform specific exception condition. The method of disposition of such a request is also platform specific.

4.1.5.1 Memory Attributes for Isochronous Requests

Within the AGP3.0 aperture (see sec 5.3) the following memory types are supported:

1. ***Write-Combining/Non-Coherent***: Processor accesses into this region are treated as uncached, non-coherent and "write-combining". AGP3.0 accesses are also treated as non-coherent with no hardware enforced coherency checks. Isochronous requests are allowed in this region.
2. ***Write-Back/Coherent***: Processor accesses to this region are considered cache-coherent and core-logic will perform hardware enforced coherency checks for AGP3.0 accesses. Isochronous requests are disallowed in this region.
3. ***Write-Back/Non-Coherent***: In this region processor accesses are cache coherent (write-back space) while AGP3.0 accesses are treated as non-coherent with no hardware coherency checks. Isochronous requests are allowed in this region.

Note that isochronous accesses are ***not*** permitted outside the AGP3.0 aperture since such accesses are always treated as coherent by the core-logic

4.1.6 Isochronous Request/Status Encoding

The AGP3.0 specification includes four new isochronous transaction codes in addition to those required by the *AGP Interface Specification V2.0.* The AGP3.0 Master device must place one of these four new codes on the "CCCC" wires to initiate an isochronous transaction. The new isochronous transaction codes are shown in Table 51. Highlighting implies a change from AGP.

Table 51: AGP3.0 Interconnect Requests

CCCC	AGP3.0 Operation	Description
0000	Read	Asynchronous Read (formerly Low Priority Read)
0001	Reserved	AGP high priority read is unused in AGP3.0
0010	Reserved	
0011	ISOCH Read	Isochronous Read command – uses the Length Field LLL in Type 1 command to specify transaction size as follows: | L-L-L	Size
-------	------
0-0-0	32Bytes
0-0-1	64Bytes
0-1-0	128Bytes
0-1-1	256Bytes	
0100	Write	Asynchronous Write (formerly Low Priority Write)
0101	Reserved	AGP3.0 does not use AGP High Priority Write

0110	ISOCH Write/Unfenced	Isochronous Write with out-of-order completion. Uses Length Field LLL in Type 1 request to specify transaction size as follows: <table><tr><td>L-L-L</td><td>Size</td></tr><tr><td>0-0-0</td><td>32Bytes</td></tr><tr><td>0-0-1</td><td>64Bytes</td></tr><tr><td>0-1-0</td><td>Reserved</td></tr><tr><td>0-1-1</td><td>Reserved</td></tr></table> No byte masking allowed so Isoch Master must assert C#/BE[3:0] lines high during data transfer to indicate that all byte lanes are active.
0111	ISOCH Write/Fenced	Isochronous Write with ordered completion. Implicit Fence forces all preceding Isoch writes (fenced and unfenced) to complete first. Isoch Writes following it cannot bypass. Size is determined the same as in Isoch Write/unfenced.
1000	Reserved	AGP3.0 does not implement Long Read
1001	Reserved	AGP3.0 does not implement Long High Priority Read
1010	Flush	Similar to a Read. This request drives all low priority write accesses ahead of it to the point that all the results are fully visible to all other system agents, and then returns a single Quad-word of random data as an indication of its completion. The address and length fields are meaningless for this request.
1011	Reserved	
1100	Fence (For both read and writes)	All asynchronous writes prior to fence will become visible before any asynchronous reads or writes completion following fence.
1101	Reserved (used for PCI Frame based Dual Address Cycle)	
1110	ISOCH Align	Returns the time offset from AGP3.0 isochronous period of core-logic (or *AGP3.0 bridge*). The core-logic will treat this request as it does a Read request and return a single aligned 32Byte value to the Device. The length field for this transaction is 000. The core-logic will return the data in the first word (4 bytes) in a common clock cycle. The remaining 7 words in the common clock cycle will have undefined data.
1111	Reserved	

4.1.7 Identifying Isochronous Data

The AGP3.0 core-logic specifies the type of isochronous data to be transmitted over the AGP3.0 Port using the two status codes that were previously reserved. The new status codes are shown below. Highlighting implies a change from AGP2.0.

Table 52: Isochronous Status ID Codes

Status	Data Type
000	Asynchronous read or flush return
001	Reserved
010	Asynchronous write data
011	Reserved
100	Isochronous memory read data
101	Isochronous memory write data
110	Calibration Cycle
111	Master has been given permission to start a bus transaction

4.1.8 Flow Control

The AGP3.0 Master device may delay the transmission of returning isochronous write data up to one common clock by delaying the assertion of **IRDY#** for one clock. The AGP3.0 core-logic may delay the start of a read data transfer by up to one common clock by delaying the assertion of **TRDY#** by one clock[24]. Once an isochronous data transfer begins it goes to completion without interruption.

Asynchronous transactions allow flow control for every four common clocks during data transmission, whereas, isochronous data transmission is never interrupted. The longest duration isochronous data payload *(8X signaling and 256 byte data transfers)* takes eight clocks. No flow control is allowed during the entire data transmission phase. The non-isochronous flow control signals **RBF** and **WBF** have no effect on isochronous transactions. When the isoch device asserts these signals, the core-logic may choose to ignore them.

4.1.9 Isochronous Bridges

AGP3.0 systems may contain an optional Fan-out Bridge component for connecting two AGP3.0 Master devices to the system. Each AGP3.0 device may read or write main system memory using isochronous transactions as specified above. The processor may use PCI transactions to write data to either AGP3.0 device in the usual way. If two AGP3.0 devices must communicate, they must do it through a section of system memory to which they both have access.

---

[24] The same is true for AGP.
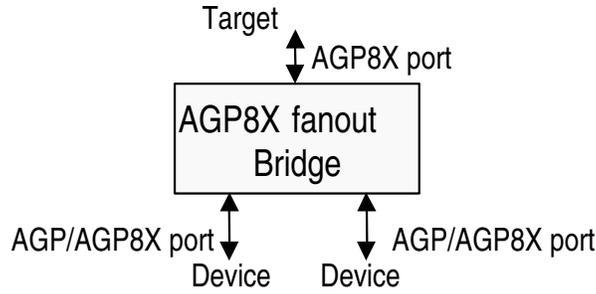
Figure 32: AGP3.0 Fanout Bridge Connections

4.1.10 Setting T, L, Y, N for AGP3.0 Core-logic and AGP3.0 Devices

Several isochronous parameters must be set to compatible values on the AGP3.0 core-logic *(target)* and the AGP3.0 Master device(s), including:

- Isochronous time period (T = 1 microsecond)
- Isochronous Data Payload size *(Y)*
- Number of isochronous transactions per period *(N)*
- Isochronous request queue size *(Q)*,

These parameters must be setup in the following order.

1. The AGP3.0 core-logic advertises its default isochronous data payload size *($Y_{core\text{-}logic}$)*, which may be adjusted upward, if necessary, to agree with the AGP3.0 Master device's isochronous payload size *(Y_{device})*.

$$Y = Max(Y_{chipset}, Y_{device})$$

2. Isochronous bandwidth is allocated in two steps:

   A. **Data Bandwidth:** The sum of AGP3.0 Master device transaction allocations must be less than or equal to the AGP3.0 core-logic's transaction capability *($N_{core\text{-}logic}$)*. The transaction capability may depend on the AGP3.0 core-logic's isochronous transaction size.

$$N_{chipset} \geq N_1 + N_2$$

   In the above equation N_1 and N_2 represent the isochronous transactions per time period T required by a one or more AGP3.0 devices. These include both read and write transactions.

   B. **Request Bandwidth:** An AGP3.0 core-logic must specify a maximum request queue capability between 1 and 256. AGP3.0 Master device(s) must make sure the AGP3.0 target's request queue is never overfilled.

An AGP3.0 Master's request queue depth must be sufficient to cover the AGP3.0 core-logic's isochronous pipeline latency (L). Plus, any additional isochronous latency contributed by an optional Fan-out Bridge, according to the following expression:

$$Q_{isoch} = \frac{BW * L}{Y}$$

Where BW/Y = N which is the number of transactions (read and write) per time period T (1 microsecond) of payload size Y.
Request queue size for an AGP3.0 Master device is the sum of its asynchronous and isochronous queue allocations.

$$Q_{device} = Q_{async} + Q_{isoch}$$

The core-logic request queue depth (specified in register AGPSTAT field RQ) must be greater than or equal to the sum of the AGP3.0 device request allocations. Note that the normal case is that of a single device on the AGP Interface.

$$Q_{chipset} \geq Q_{device\_1} + Q_{device\_2}$$

An exception to the above rule is in the case of isochronous writes with payload size greater than 64 bytes. In this case, the setup software that establishes the contract parameters must ensure that Q_{isoch} includes the increase in write transactions to satisfy the isochronous payload. The programmed value of the request queue size for the Master device must not exceed the Core-Logic's request queue depth as stated in AGPSTAT.RQ.

Take the case of Y=256 bytes.

$$N_{device} = N_{read} + N_{write} \qquad \text{in terms of payload size Y = 256 bytes.}$$

$$Q_{isoch} = N_{read} * L + N_{write} * L * \frac{Y}{64}$$

$$Q_{device} = Q_{async} + Q_{isoch}$$

And the software must program the parameters such that

$$N_{chipset} \geq N_{device}$$

$$Q_{device} \leq Q_{chipset}$$

Data (N) and request (Q) resources are shared between AGP3.0 Master devices connected to a Fan-out Bridge. This specification requires AGP3.0 setup software to establish these parameters in a compatible way between core-logic and AGP3.0 Master device(s) following the sequence of steps outlined above.

If a Fan-out Bridge is present, the transaction capability and request queue resources will be first allocated to the device with the smallest requirement, with any remaining resource allocated to the other device. Driver software for the AGP3.0 devices are free to change this allocation later, if desired.

4.1.11   Time Keeping

An AGP3.0 device must be aware of the isochronous time period (T) to ensure that it does not exceed the agreed-upon maximum number of isochronous transaction requests. AGP3.0 bridges must also be aware of the isochronous period for the same reason. Timekeeping is optional for AGP3.0 core-logics.

AGP3.0 Master devices must synchronize their isochronous periods to that of the bridge or core-logic by issuing a time alignment transaction. The bridge forwards the transaction to the core-logic, which responds with the time offset *(in common clocks)* between the initial data phase of the time alignment data, and the beginning of the core-logic's next isochronous period. The AGP3.0 Bridge shifts its isochronous period by this amount, which brings it into time alignment with the core-logic. The AGP3.0 Bridge then responds to the AGP3.0 devices that shift its isochronous periods by this amount, which then brings them into synchronization with the isochronous period of the AGP3.0 bridge and core-logic.

Isochronous alignment transactions resemble an isochronous read transaction except that the return data contains the time offset, in AGP3.0 common clocks, between the initial data phase of the return data and the beginning of the AGP3.0 core-logic's next isochronous period. The value ranges between 0 and 65 *(decimal)* or xxxxxx00 to xxxxxx41 *(hexadecimal)*. An AGP3.0 core-logic may return the value xxxxxxFF *(hexadecimal)* indicating it does not maintain on an internal isochronous period. Note that only the lowest significant byte has any meaning. All other bytes in the 32-bit word must be ignored.

The bridge may shift the isochronous periods of the AGP3.0 Master devices relative to its own isochronous period so the bridge can combine requests from two AGP3.0 Masters into a single upstream isochronous period. The amount of this shift is an implementation detail left to the bridge designer. Figure 33 illustrates the time offset introduced by a hypothetical AGP3.0 bridge.
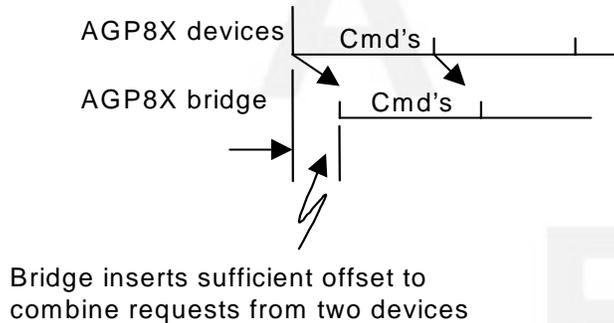
Figure 33: Potential Fan-Out Bridge Delays of Isochronous Periods for AGP3.0 Devices

AGP3.0 bridges may also delay data transfers provided the total delay of requests and data does not exceed one isochronous period. Isochronous delay experienced by an AGP3.0 Master device is the sum of contributions from the core-logic and bridge. The AGP3.0 Master device is required to tolerate the additional period of latency.

Future versions of this specification may require AGP3.0 core-logics to maintain an isochronous period and participate in the time alignment process described above to support a more rigorous definition of isochronous latency.

4.1.11.1   *Isoch Align Transaction*

An Isoch Align transaction is similar to a Isoch Read transaction except that it has a special meaning to the Core-Logic. If the Core-Logic supports time-keeping, it will return an offset value to allow the isoch device to align its isochronous period to that of the core-logic. The value of this offset is returned in the lower 8 bits of a 32 bit word. The remaining bits in the word should be ignored. The offset value ranges from xxxxxx00 through xxxxxx41 (hexadecimal). If the core-logic does not support time-keeping it will return a value of xxxxxxFF. All other values of the lower 8-bits are reserved.
The core-logic must return the data for an Isoch Align transaction in the first 32-bit word in a common clock cycle. The remaining 7 words of that common clock cycle are undefined and must be ignored by the isoch device. A core-logic may drive the remaining 7 words of this cycle with undefined data. Regardless of the transaction size selected for normal isoch reads, the length field for isoch align will be 000, which selects 32Bytes.

Since Isoch Align is an isochronous transaction, all isoch protocol requirements must be obeyed.

4.2 Flow Control Change

Since isochronous and asynchronous transactions may be concurrently processed on the AGP3.0 interface, it is necessary to ensure that a throttled (stalled) asynchronous transaction does not cause the isochronous latency specification to be violated. Based on the flow-control discussion in Section 2.3.3, the following points emerge:

- Block-level flow control will stall the interface. Buffer-Full flow control only stalls a new asynchronous data transfer from starting.
- AGP3.0 "Reads and Writes" cannot be flow-controlled at the block-level since the maximum transaction size is 64 bytes.
- AGP3.0 Fast Writes can be flow-controlled at the block-level since there is no limit to the size of the transaction.

Two modifications made to the asynchronous transaction flow control are required to ensure proper isochronous operation. These modifications are described in the following sections.

4.2.1 Isochronous Transactions and Buffer-Full Flow Control

Data transfers for isochronous read or write transaction will ignore buffer-full flow control applied using **RBF** and **WBF**. The core-logic that initiates these transfers may pass a stalled asynchronous data transfer with an isochronous data transfer. Simply put, the core-logic ignores **RBF** and **WBF** when it initiates an isochronous data transfer.

4.2.2 AGP3.0 Fast Write Flow Control

For AGP "Fast Writes", the maximum number of wait states introduced during each block-level stall was not specified. In AGP3.0, the maximum number of Wait states is limited to two common clock cycles. In other words, the AGP3.0 Master must not suppress **TRDY** for more than two cycles including the throttle point cycle. If the AGP3.0 Master needs additional time to receive the Fast Write data, it must use the Fast Write termination schemes that are defined in the AGP interface specification. The AGP3.0 Master is allowed to disconnect a stalled transaction with or without accepting any data. The target is required to resume where the disconnected transaction left off.

Disconnecting a stalled Fast Write ensures that a pending isochronous transfer can complete in a timely manner.

4.3 Synchronization Models

This section presents a discussion of the synchronization models used for interchange of data between different agents in the platform. The basic reason for synchronization is to guarantee that when one agent performs a data movement in the platform for consumption by a second agent, the second agent receives the intended data.

There are several instances of this as presented in the following list:

1. Processor writes data to Main Memory and AGP3.0 device reads this data.

2. AGP3.0 device writes data to Main Memory and the processor reads this data.

3. A 3rd agent (say another PCI device) writes to Main Memory and AGP3.0 device reads this data.

4. AGP3.0 device writes to the Main Memory and a 3rd agent reads this data.

5. One AGP3.0 device behind a Fan-out Bridge writes to the Main Memory, while the other AGP3.0 devices behind the same bridge read it.

✍ NOTE

The mechanisms that apply to sections 1 and 2 above generally work for the rest. However, the method used to signal the other agent varies.

A few terms need to be defined before we proceed to synchronization schemes.

4.3.1 Global Visibility

A write to system memory reaches global visibility when all agents accessing that memory address get the updated data. As an example, consider an AGP write to system memory location A. If the write data is sitting in a write buffer within the AGP Port and the core-logic does not consider this write buffer as part of the globally visible state, a subsequent access to location A does not read the contents of this buffer. The "write data" then has not reached a state of global visibility. If the write data moves into a posted write buffer in the memory controller that is checked on every subsequent access from anywhere in the system, then the write data is now globally visible and no other (or older copies) of this data matters. Another way to define the point of global visibility is as a point of serialization of all requests from anywhere in the system. If the state being modified is in cacheable space, the global visibility is attained only after the cache coherency enforcement has been completed.

4.3.2 Request Completion

- A Write request is completed only after the state being modified is globally visible.
- A Read request has two levels of completion:

  1. When the data fetch has taken place implying that the data to be returned is now committed.
  2. The actual return of data to the requesting agent.

Since the second phase is eventually completed, the completion of the first phase (which decides on the data to be returned) is more important.

4.3.3 Write Completion

Since "Writes" from any agent (processor, or AGP3.0 for example) device may be posted, this data could potentially sit in write queues indefinitely. This is especially true if there are dependencies on subsequent events (such as more writes) to flush these queues.

The AGP3.0 specification then requires that all AGP3.0 initiated writes to system memory must be completed in a timely fashion, without the need for any other system event. While no actual time is being specified for this completion, it is expected that the core-logic shall ensure that these writes have a fair and timely opportunity to propagate to the point of global visibility.

This process applies to both asynchronous and isochronous writes. The guaranteed write completion ensures that a FLUSH is not required on the AGP3.0 interface forcing a write to completion.

4.3.4 Synchronization Case 1: Processor Data to AGP

The following scheme ensures correct operation.

Table 53: The Synchronization Sequence of Data from Processor to AGP

Processor Actions	Core-logic Actions	AGP Device Actions
Writes (or flushes cache) data to System Memory (either WC or WB Space).	Accumulates writes in posted write buffers. Performs any Coherency operations.	Polls for flag indicating buffer availability to set
Executes un-cached access such as an I/O space read or write.	Flushes all posted write buffers into system memory (or to global visibility) before completing un-cached transaction.	Polls for flag to set
[25]Signals to AGP3.0 Device by reading or writing I/O register or writing memory location to set flag	Guarantees that this write (to AGP Device or memory) will complete after write buffers are flushed.	Finds flag set and begins to read the data that processor wrote in memory.

---

[25] This signal to a AGP3.0 Device may be merged with Step 2 that causes core-logic to flush buffers.

4.3.5 Synchronization Scheme: Case 2 AGP Data to Processor

The following scheme ensures correct operation.

Table 54: The Synchronization Sequence of Data from AGP to the Processor

AGP3.0 Device Actions	Core-logic Actions	Processor Actions
Writes to memory (Could be low priority Async. or Isoch).	Writes may complete in any order in system memory.	Processor polls flag to set. (Optionally waits for signal to poll).
Executes a Fence for Asynchronous Writes (skip step for Isoch Writes).	Core-logic inserts Fence in low priority write queue to ensure order.	Processor polls flag to set (Optionally waits for signal to poll).
Writes flag in system memory to indicate completion. For Isoch writes this will be an Isoch Write/Fenced. This system memory is likely to be in non-snoop space.	Core-logic will guarantee that this write will complete following previous writes.	Processor polls flag to set (Optionally waits for signal to poll).
Optionally signals processor to poll flag (interrupt or cached write).	Core-logic delivers this signal if needed.	Processor finds flag set and proceeds to read AGP3.0 data.

✎ **NOTE**

Asynchronous and Isoch Writes may be completed in any order. It is essential to insert an ordering operation such as a Fence before setting a flag to signal the processor.

The write of the flag to indicate buffer availability is likely to be in un-cached space especially for Isoch Writes. If this is the case, the processor polling the flag generates a lot of undesirable system traffic. There are probably several ways around this problem. One of these is for the processor to not poll the flag until another signal is received. This signal could be an interrupt (from an AGP3.0 device or a periodic system event) or a setting of a secondary flag in cached space, which may be polled locally in the processor's cache without generating system traffic.

The exact signaling mechanism is implementation specific and beyond the scope of this specification.

4.4 Fan-Out Bridge

AGP3.0 is a point-to-point interface, necessitated by both the signaling scheme and the speed of the data transfer. This means that an AGP3.0 interface can support only one device slot. A Fan-out Bridge component could be designed to interconnect more than one AGP3.0 device to a single AGP3.0 port from the core-logic. Such a bridge would be located on the motherboard side, which, from a device configuration perspective, is different from a similar bridge located on the graphics card. A bridge located on the motherboard needs to ensure that it behaves exactly as the Core-Logic to any AGP device that connects to it.

While the design specification of the Fan-out Bridge component is beyond the scope of this document, a few basic requirements for such a component will be described here. These are listed as follows.

- The Fan-out Bridge features are required to be "transparent" to the software. This implies that the bridge does not require any PCI enumeration or configuration. Furthermore, it does not contain software-visible (OS, applications or drivers) programmable state. An exception to this might be a level of BIOS initialization of platform specific features. However, the AGP3.0 specification does not describe any platform specific features.

Since the Fan-out Bridge is transparent, it implies that the AGP Devices are logically on the primary segment and are indistinguishable to software from multiple devices on a traditional multi-drop bus.

- The Fan-out Bridge cannot have an impact on the AGP flow control protocol. This implies that it must supply adequate buffering to manage the flow control on each side of the bridge independently.
- No direct peer-to-peer capability should be assumed between devices behind a fan-out bridge. The core-logic is not required to comprehend such traffic and no ordering rules for such traffic with respect to the AGP traffic will be specified. Any peer-to-peer implementation scheme is outside the scope of this specification. Indirect peer-to-peer between such devices using a shared area in system memory is possible within the scope of this specification.
- AGP devices installed in the slots supported by the bridge are configured as Type 0 PCI devices. Since the Fan-Out Bridge is transparent, it must direct the PCI configuration cycles to the appropriate AGP3.0 device based on the **IDSEL**. At the AGP device level, **IDSEL** is always tied to AD16. Therefore, the Fan-Out Bridge must implement a scheme to direct configuration cycles with different **IDSEL**s from the core-logic to the appropriate AGP device with IDSEL remapped to AD16. Figure 34 illustrates a possible scheme.
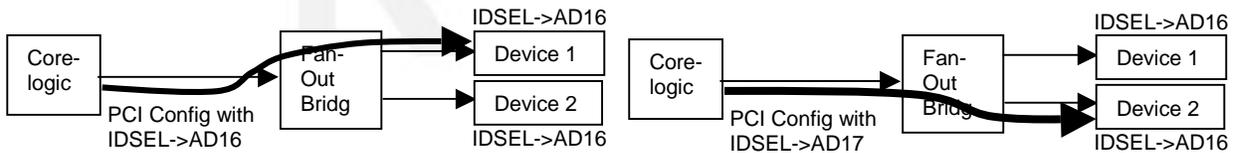

Figure 34: Fan-out Bridge Scheme Example

5 Appendix B: Workstation Programming Model

5.1 System Components
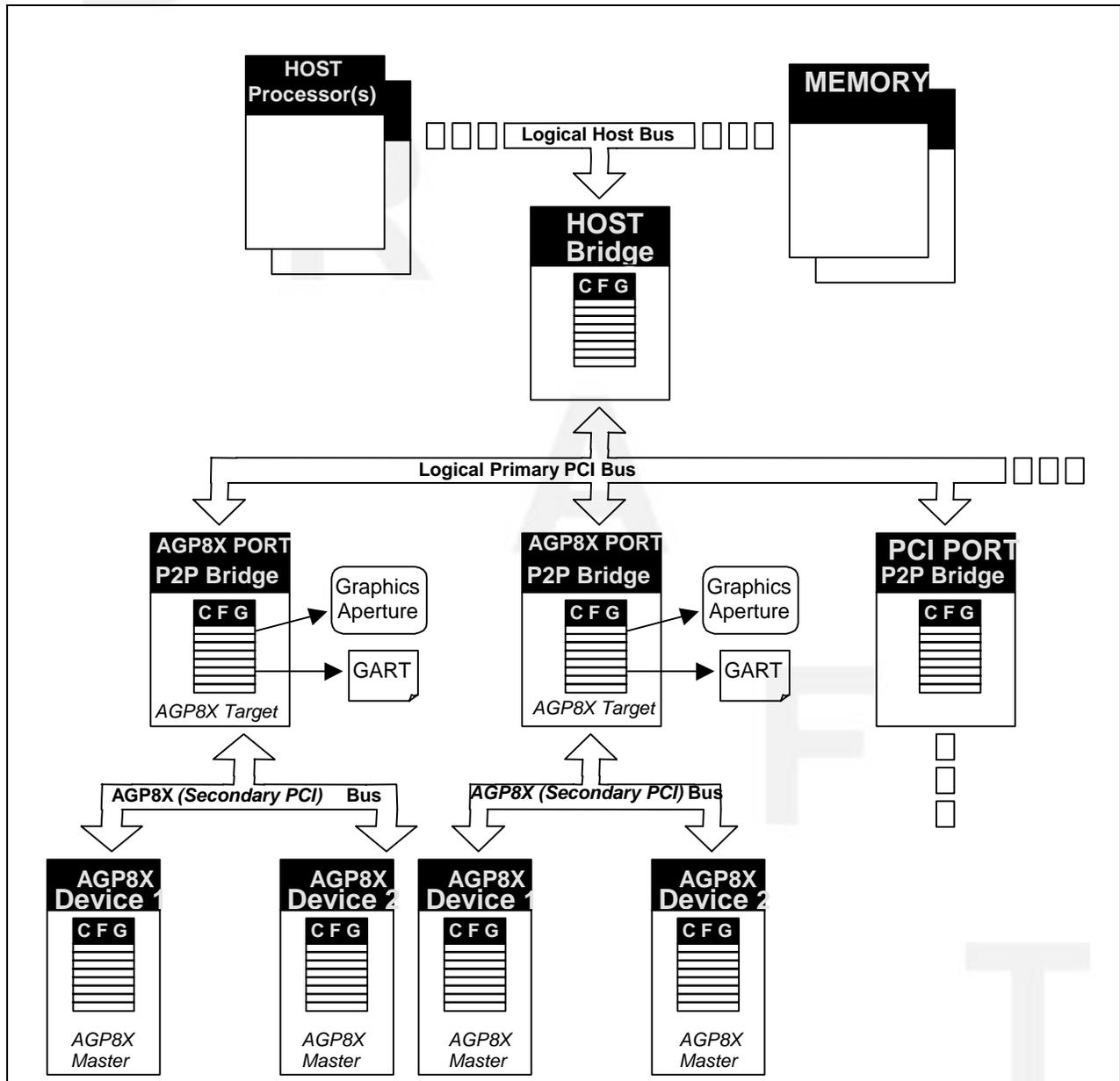
Figure 35: AGP3.0 Generalized Logical System Overview

Figure 35 illustrates a typical host Bus bridge (also known as core-logic) implementation that supports the AGP interface along with other typical interfaces (or ports). Ports, other than the AGP3.0 Port, of the core-logic are included for illustrative purposes only and are not required by this interface

specification. The example core-logic provides ports to the processor, system memory, the PCI Bus, and to AGP3.0.

5.2 Port and Device Definitions

1. **AGP3.0 Port:** The AGP3.0 Port is the target of requests from the AGP3.0 (Master) devices. It is provided by the core-logic implementation. AGP3.0 provides for multiple AGP3.0 Ports with multiple devices on each port.
2. **Processor Port:** The processor port is not required when supporting an AGP3.0 interface, but is typically supported by a Host Bus Bridge. The processor port provides a means for the processor to generate accesses to the PCI Ports, AGP3.0 Port, and to the memory controller. The core-logic determines to which port these accesses are routed by using information stored in the Host Bus Bridge Configuration Space block. This information is provided during the initialization process.
3. **PCI Port:** The PCI Port converts processor accesses into PCI transactions. Since the processor has no Configuration commands, the PCI Port generates PCI Configuration commands as described in the *PCI Local Bus Specification*. The PCI Port takes memory commands that address system memory and forwards them to the Memory controller. The AGP3.0 specification only requires the PCI Port controller to provide a means for PCI masters to have peer-to-peer *write* access to the PCI target that resides on the AGP3.0 Port.
4. **MEM Port:** This port provides a means to connect system memory to the core-logic. The memory controller is responsible for converting accesses that are initiated on other ports (processor, AGP3.0, and PCI) into memory commands.

When the core-logic supports an AGP3.0 Port, it requires new logic that has not been incorporated in previous core-logics, however no new functionality is required to boot the system. To enable the use of existing enumeration code to handle AGP3.0 devices, the core-logic will use functionality already defined by the PCI Local Bus Specification. The Configuration Space of the Bridge containing the AGP3.0 Port also contains configuration registers used to specify parameters associated with the GART and circuitry in the AGP3.0 interface.

5.2.1 AGP Master Device Memory Reference

AGP Master devices have a certain amount of memory resources that must be placed somewhere in the system memory address map using a PCI base address register. These memory resources fall into two categories, prefetchable and non-prefetchable, address regions. Prefetchable memory space is generally where the linear frame buffer is mapped to provide performance improvements. Non-prefetchable memory space is generally where control registers and FIFO-like communication interfaces are mapped. Each of these address regions should have its own base address register. See the *PCI Local Bus Specification* for a description of PCI base address registers.

5.2.2 AGP3.0 Port Requirements

1. The core-logic locates all AGP3.0 configuration registers within a single function of the AGP3.0 Port device; this allows System Software to easily support AGP3.0. The core-logic ensures that the configuration of all AGP3.0 Port resources, AGP3.0 aperture, and GART reside within this single function space. The registers are described in more detail later in this section.
2. The location of AGP3.0 configuration registers may either be in the core-logic's Host Bridge or all within a PCI-to-PCI bridge or both.
3. It may be possible for a core-logic implementation to place AGP3.0 configuration registers in a Host-Bridge device; it is not known at this time if this will work across various future operating systems or allow a portable software device driver. However, such hardware implementations may be unavoidable and so the AGP3.0 specification allows it.
4. Core-logic provides each AGP3.0 Port with a separate, independent Graphics AGP3.0 aperture and GART that is shared by all devices on that port.
5. Each AGP3.0 device on a given AGP3.0 Port resides on the bridge's PCI side; each AGP3.0 device has a unique PCI Device address on this interconnect.

AGP3.0 does not, in general, require the support of PCI Peer-to-Peer accesses. Devices needing to share data do it through shared buffers in system memory. Note that the AGP3.0 specification explicitly relaxes some peer-to-peer requirements of the *PCI-to-PCI Bridge Specification*. See the following below. Table 55 describes the requirements. In this table, "AGP3.0$_a$" represents one AGP3.0 port in the platform. A "non-AGP3.0$_a$" port could be another PCI or a different I/O port. The "type" field represents the transaction on the initiator's port.

Table 55: PCI Peer-to-Peer Access

Initiated From	Targeted To	Type	Support
Device(Port == AGP3.0a)	Device(Port != AGP3.0a)	AGP3.0 / PCI Read and Write	**Not Required**
Device(Port == AGP3.0a)	Device(Port == AGP3.0a)	AGP3.0 / PCI Read and Write	**Not Required**
Device(Port != AGP3.0a)	Device(Port == AGP3.0a)	AGP3.0 or PCI Read	**Not Required**
Device(Port != AGP3.0a)	Device(Port == AGP3.0a)	AGP3.0 Fast Write or PCI Write	**Required**

5.2.3 Multiple AGP3.0 Ports

AGP3.0 allows a core-logic implementation to provide multiple AGP3.0 Ports. Each AGP3.0 Port is a bridge device with multiple AGP3.0 devices hanging off the secondary bus. Each Port has a separate Graphics AGP aperture and GART that is independent and not shared with another AGP3.0 Port; however, these are shared across the devices within a single AGP3.0 Port.

5.3 AGP Graphics Aperture

In an AGP3.0 system, driver software and an AGP3.0 device can share large amounts of data through buffers placed in system RAM. A large buffer requires many host processor virtual pages; although host system software ensures that these pages appear contiguous to host software (*virtually contiguous*). It is often difficult for system software to map these virtual pages to contiguous physical pages in system memory (*physically contiguous*). Thus, in the absence of any sort of remapping mechanism, these pages appear non-contiguous to the rest of the system and require scatter/gather hardware in each device that will access the buffers to deal with the discontinuity.

Like AGP, AGP3.0 provides a solution to this problem in the form of an AGP Graphics aperture. The AGP aperture is a physically contiguous range of the physical address space where AGP3.0 Master accesses directed to it are re-mapped (translated) to potentially physically non-contiguous pages. AGP3.0 Master translation is accomplished through the AGP Graphics Aperture Re-mapping Table (GART). For the purposes of translation, the AGP aperture range is split into a series of aligned regions, each such region is termed an AGP aperture page. Each AGP aperture page has a corresponding translation in the GART.
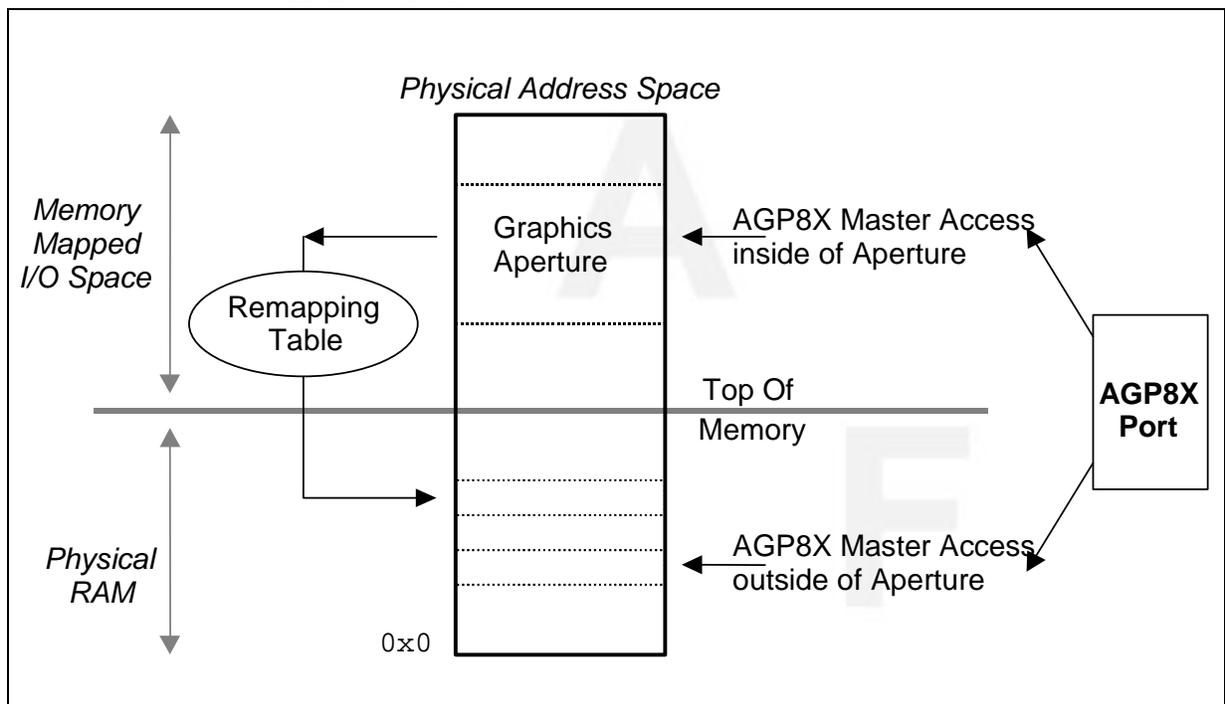

Figure 36: AGP Graphics Aperture Model

In general, system software places the AGP aperture above the top of the memory (above the highest byte of actual physical RAM in the system) in a hole that does not conflict with memory-mapped I/O registers of system devices.

AGP3.0 supports AGP3.0 aperture sizes of 4MB and larger; each core-logic implementation indicates which AGP3.0 aperture sizes are supported through the APSIZE register.

Translation (or re-mapping) through the GART is a physical-to-physical translation and is completely independent of virtual-to-physical translation performed by host processors in the system.

An AGP3.0 target (core-logic) may implement a number of Graphics Translation Look-aside Buffers (GTLB) to speed-up translation of AGP3.0 aperture page addresses to system memory locations.

5.3.1 AGP Aperture Page Size

The core-logic implementation is configured by system software to support one, common AGP aperture page size for all pages in the AGP aperture. The core-logic includes a mechanism that allows system software to select that common AGP3.0 aperture page size from a set of supported page sizes. At any given time, one and only one AGP3.0 aperture page size is used for all pages within the AGP aperture.

Core-logic indicates which AGP aperture page sizes are supported through the NEPG register. System software selects one page size from the set of supported AGP aperture page sizes by programming the value into the NEPG.SEL. A core logic implementation must always support an AGP aperture page size of 4 KB; implementations are encouraged, but not required to support other AGP aperture page sizes. (Note that NEPG defaults to selecting 4-KB AGP aperture pages.)

System software populates (maps) each AGP aperture page with a same-sized, naturally aligned region from physical memory. Note that there need be no correspondence between the host processor page size and the AGP aperture page size used by core logic. The only requirement is that each populated AGP aperture page translates to a fully allocated and resident physical memory region that is of equal size and is naturally aligned.

When given the choice, System Software should select the largest AGP aperture Page size that is compatible with operating system memory allocation algorithms. Larger AGP aperture page sizes reduce the size of the GART; allow more freedom in which system pages can be mapped into the AGP aperture; and can allow an implementation to improve the efficiency of GART translations.

5.3.2 AGP Graphics Aperture Requirements

1. To configure the size of the AGP aperture, use the software programmable APSIZE to select the desired AGPaperture size. The selected AGP aperture size causes the low-order bits of APBASE to be read-as-zero, thus forcing natural alignment of the AGP aperture.
2. Software sizes and locates the AGPaperture by programming the APBASE base registers (APBASELO and APBASEHI). Core-logic implementations that only support 32-bit physical addressing do not need to implement APBASEHI.
3. AGPCTRL.aperenb enables translations through the AGP aperture of AGP Master Accesses. When the AGP aperture is disabled, AGP3.0 Master accesses to the AGP aperture range are treated by the Core-logic exactly in the same manner as AGP3.0 Master accesses outside of the AGP aperture.
4. System software (and the BIOS) must ensure that the AGP aperture is located within the address range that can be addressed by the AGP device. This range can be determined by checking AGPSTAT.OVER4G and AGPCMD.OVER4G of both AGP Master and Target.
5. Core-logic implementations are strongly encouraged to support a range of AGP aperture sizes that span 16MB-256MB. Larger aperture sizes of 1GB or more would be appropriate for higher end workstation platforms.
6. All AGP Master accesses that fall within the AGP aperture are translated regardless of protocol when the AGP aperture is enabled; this is termed *AGP Master translation.*
7. The Core-logic must support AGP Master access to locations outside of the AGP aperture.
8. The Core-logic is not required by the AGP specification to translate accesses directed to the AGP aperture by a host processor – termed *host translation.* Portable AGP3.0 software should not rely upon the existence of host translation.

9. If core-logic supports host translation, it reports the capability in AGPSTAT.htrans#. Host-translation uses the GART for translation.
10. The operation of the AGP Port and AGP Master accesses to system memory function correctly regardless of whether host translation is present in the core-logic.
11. It is implementation-dependent whether the core-logic translates accesses directed to the AGP aperture by any system component NOT on the AGP Port – including PCI devices on other Buses. Software should not rely on this capability.
12. Core-logic indicates which AGP aperture page sizes are supported through the NEPG register. System software selects one page size from the set of supported AGP aperture page sizes by programming the value into the NEPG.SEL.
13. A core-logic implementation must always support an AGP aperture page size of 4KB; implementations are encouraged, but not required to support other AGP aperture page sizes. (Note that NEPG defaults to selecting 4KB AGP aperture pages).
14. Software must use the procedure described in Section 5.3.5 to relocate and resize the AGP aperture during normal system operation.

5.3.3 GART Overview

The GART is a re-mapping table of translations for accesses to the AGP aperture. The GART resides in system RAM. Each aligned AGP aperture page has a corresponding GART entry, which translates it.

AGP allows the GART to be organized in a DIRECT-MAPPED format; an offset into the AGP aperture is scaled down and used as an index to point directly to a GART entry in the GART.

A GART entry is marked as a valid translation when its corresponding valid bit is set. System software ensures that all GART entries needed to translate AGP aperture addresses are valid. If, when attempting to translate an AGP aperture address, core-logic accesses an invalid GART entry, it performs a platform specific exception action.

AGPCTRL.gtlben controls the caching of GART entries in an implementation-defined GTLB.

5.3.4 GART Requirements

1. The GART resides in the system physical address space and is pointed to by the address in the GARTLO and GARTHI configuration registers. It can be directly accessed by host software using memory or memory-mapped I/O addressing. For sake of discussion:

    ```
    GART_START = (GARTHI << 32) + ( GARTLO & 0xFFFFF000)
    ```

2. The starting location of the GART is always 4KB-aligned.
3. When the AGP aperture is enabled. The GART is always enabled; there is no way to turn off the GART without disabling the AGP aperture.
4. The core-logic implementation chooses one width for all GART entries; the width is either 32 or 64-bits and is reported by AGPSTAT.gart64b; this parameter cannot be programmed by software. The format of a GART entry is shown below. *(This format in conjunction with larger AGP aperture page sizes, allows an implementation to keep using 32-bit GART entries even as physical addresses grow larger than 40-bits. The format is also compatible with the i440BX style GART entries)*

    ```
    Let, SPFN[63:0] = Zero_Extend( Target_Physical_Address) >> log2( Aperture_Page_Size)
    ```

Table 56: Bit Positions

Bit Position	Field	Definition
0	Valid	When 1, the GART entry is a valid translation. When 0, the GART entry is invalid and accessing the entry for translation causes an implementation specific error condition..
1	Coh	This bit in combination with AGPSTAT.ITA_COH controls coherency support for accesses that use this GART entry. See section 5.4 for details.
3:2	Rsv	Reserved for future use by the AGP3.0 specification. SBZ[1].
11:4	SPFN1	SPFN[27:20]. In a 4KB-GART-page implementation, with 32-bit Physical Addressing, these bits will be '0'
31:12	SPFN0	SPFN[19:0]. In a 4KB-GART-page implementation, with 32-bit Physical Addressing, these bits will correspond to the bits[31:12] of the target Physical Address. All bits are valid and must be filled in by software.
63:32	SPFN2	SPFN[59:28]. Only present in a 64-bit width GART entry.

✎ NOTE

[1]*Software should write zero to these bits; AGP3.0 requires that a value of zero will always produce backward compatible behavior in all future implementations that define these bits.*

The GART must reside in a physically contiguous memory. The size of the entire GART is:

GART_Size = (Aperture_Size / Aperture_Page_Size) * 2 ^ (2 + AGPSTAT.gart64b)

5. Each GART entry maps an aligned AGP aperture Page address to an actual physical RAM location.
6. Let **M** be the log2 number of bytes in a GART entry (either 2 for 4-byte entries or 3 for 8-byte entries). For an AGP aperture address, A, the corresponding GART entry address is then:

$$\mathbf{entry\_address = GART\_START + (\ (\ (A\text{-}Aperture\_Base)\ >> log2(Aperture\_Page\_Size)\ )\ << M\ )}$$

7. Software may locate the GART anywhere within physical RAM addresses; the GART cannot be located in Memory-Mapped I/O space. The core-logic cannot place constraints on where the GART can be located within physical RAM addresses.
8. Core-logic accesses to the GART are not coherent with host processor caches. Portable system software should place the GART in a range of the physical address space that is considered un-cacheable by host processors. (A good example is mapping the GART as UC in an Intel® Pentium® II processor).
9. Attempting to translate an AGP aperture address through an invalid GART entry is a non-recoverable error condition. When this happens, core-logic does the following:
   a) It sets an implementation specific error bit and optionally records the invalid GART entry address in an implementation specific register. These remain latched until explicitly cleared by software or on Power-On-Reset.
   b) Core-logic causes a processor-specific action that is platform dependent.
   c) If the transaction is a read of the AGP aperture, core-logic may either cancel the request or complete the request by returning UNDEFINED data.

    d)   If the transaction is a write of the AGP aperture, core-logic cancels the request.
10. Core-logic supports modification of the GART during normal system operation provided that software follows the procedure outlined in Section 5.3.5.
11. The caching of GART entries in an implementation defined GTLB is controlled by AGPCTRL.gtlben.
12. Core-logic must support the aliasing (or mapping) of multiple AGP aperture pages to the same post-translated page. The corresponding gart_entry.coh bit, if supported, must apply coherency attributes.

5.3.5 Requirements for Modifying GART Entries

The following steps assume that the core-logic implements a GART Translation Look-aside Buffer (GTLB). A GTLB is not a required feature.

1. System software must ensure that all host processor and AGP Master transactions to the AGP Port and the AGP aperture that could potentially use the candidate GART entries are suspended. Once suspended, system software must ensure that all previous transactions have been flushed from all processor write queues that target AGP aperture locations mapped by the candidate GART entries.
2. System software writes '0' to AGPCTRL.gtlben using a strongly ordered, un-cacheable write.
3. System software is now free to modify GART entries using strongly ordered un-cacheable writes.
4. Using mechanisms specific to the operating-system and host processor architecture, system software needs to update any host processor translations that map the AGP Port or AGP aperture which are affected by changes to the GART entries.
5. When all modifications are finished, system software issues a strongly ordered, un-cacheable read for the last datum written to system memory; this ensures that all memory changes have propagated.
6. System software writes '1' to AGPCTRL.gtlben using a strongly ordered, un-cacheable write.
7. Using mechanisms specific to the operating system and host processor architecture, system software resumes AGP Port and AGP aperture transactions on all host processors.

5.4 Coherency Requirements

When the AGP target makes an access from an AGP Master coherent, the target ensures that:

- A read request receives the most current value of the datum from among all host processor caches, memory, and other coherent caching agents in the system;
- A write request's value becomes the most current value of the datum in the system and is made visible to all host processor caches, memory, and other coherent caching agents in the system.

When the AGP target does not make an access from an AGP Master coherent, the target need only update system memory without affecting the state of any caching agent in the system.

When the AGP target is not required to provide a specific coherency for an access, it is unpredictable if the above coherency guarantees are provided for the access; whether coherency is provided can change from moment to moment and implementation to implementation. When no specific coherency is required for an access, software should not rely upon coherency being provided.

Coherency is applied by the AGP Target to addresses generated after GART translations, rather than to the actual Graphics AGP aperture addresses, for access within the Graphics AGP aperture.

Table 57 identifies the actions of the AGP target that apply to any type of protocol access (AGP or PCI) made by an AGP Master:

Table 57: Coherency Table

Inside Graphics AGP Aperture	Protocol-Type	AGPSTAT.ita_coh (Support for Coherency inside AGP3.0 aperture?)	Gart_entry.coh	Coherency Action Ensured by AGP Target
NO	X	X	X	Access is coherent
YES	X	1	1	Access is coherent
YES	X	0	1	Specific coherency not required for Access
YES	Not PCI-Protocol	X	0	Access is not coherent
YES	PCI-Protocol	X	0	Specific coherency not required for Access

It is implementation-dependent and outside the scope of the AGP3.0 specification whether a core-logic implementation supports coherency for accesses to the graphics AGP aperture from masters not on the AGP Port.

5.4.1 Coherency of Host Processor Accesses

The coherency mechanisms described in the AGP3.0 specification do not control nor influence the coherency of any host processor access to any region of physical memory, including the Graphics AGP aperture. Thus, the coherency of a host processor access to a graphics AGP aperture page would ignore the value of the page's gart_entry.coh bit, and all other such mechanisms described above.

When the AGP Target supports host translation through the AGP aperture (AGPSTAT.htrans# is '0'), it is recommended that software not use GART entries with gart_entry.coh set to '1'. This mode of operation can result in unpredictable coherency for accesses. The unpredictable nature is because host processor accesses will have coherency applied to the pre-GART address using processor coherency controls. However, AGP Master accesses will have coherency applied to the post-GART address using gart_entry.coh. Table 58 describes the coherency behavior of the core-logic based on whether host processor accesses are translated or not. All accesses are assumed to target the AGP aperture.

Table 58: Host translation effect on Core-Logic behavior

Htrans#	ita_coh	gart_entry_coh	Target response to AGP Access
0	0	X	Non-Coherent access
0	1	0	Non-Coherent access
0	1	1	Coherent access. Core-logic uses Pre-GART address for coherency check.[26] This mode of operation is not recommended.
1	0	X	Non-Coherent access
1	1	0	Non-Coherent access
1	1	1	Coherent access. Core-Logic uses Post-GART address for coherency check.

5.5 System Software Initialization

The operating system initializes AGP3.0 features by performing the following operations:

1. Allocate memory for the AGP remapping table.
2. Initialize the AGP target's address remapping hardware.
3. Set the AGP target and master data transfer parameters.
4. Set host memory type for AGP memory.
5. Activate policy limiting the amount of AGP memory.

An AGP core-logic driver API will be used for the second item. Refer to the appropriate Operating System device driver interface kit for details.

---

[26] Core-logic must ensure that any buffers with post-GART addresses are also included in the coherency check.

The third item requires access to configuration registers defined later in this interface specification. Reading bit 4 set to 1 in the Status register at offset 6 indicates the device implements the New Capabilities mechanism as described in the *PCI Local Bus Specification*. The New Capabilities structure is implemented as a linked list of registers containing information for each function supported by the device. AGP status and Command registers are included in the linked list. The structure for the AGP specific ID and structure is illustrated in Figure 37.
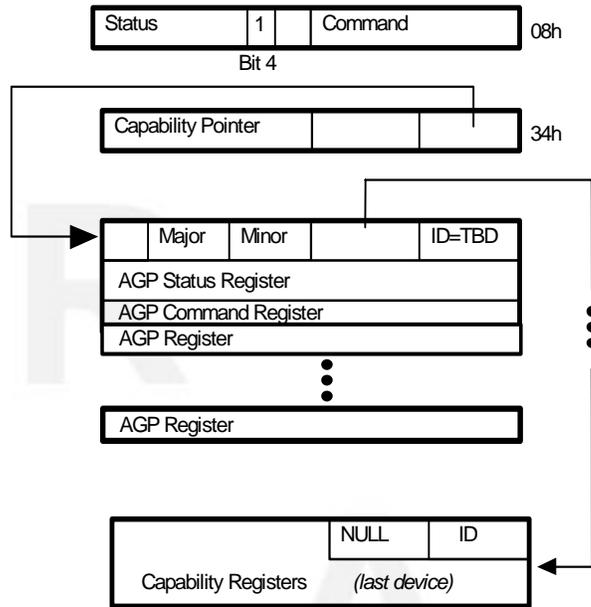

Figure 37: Location of AGP3.0 Capabilities

The operating system and BIOS use configuration registers to initialize AGP3.0 features. Both AGP Master and Target devices in the registers described in the following sections must support these features. The text describes the specific behavior of the target and master with respect to each function.

5.6 AGP Registers

The registers described below are a superset of the Core AGP Register set described in Section 2.5. The additional registers described in this Appendix support the following:

1. Programming for additional features such as isochronous transactions that are described in Appendix A and B.
2. Programming for the GART and AGP aperture setup.

These registers are located in AGP configuration space of the core-logic (also called AGP target) and AGP Device (also called AGP Master). For the core-logic, the AGP configuration space could be entirely in a Host-to-PCI Bridge function and/or a PCI-to-PCI Bridge function.

	byte 3	byte 2	byte 1	byte 0		
	apbaselo				:	10h
	apbasehi (optional)				:	14h
			CAPPTR		:	34h

AGP Capability Block

NCAPID	:	+	00h
AGPSTAT	:	+	04h
AGPCMD	:	+	08h
NISTAT (optional)	:	+	0Ch
AGPctrl	:	+	10h
nepg (optional) / apsize	:	+	14h
gartlo	:	+	18h
garthi (optional)	:	+	1Ch
NICMD (optional)	:	+	20h
reserved	:	+	24h
reserved	:	+	28h
reserved	:	+	2Ch
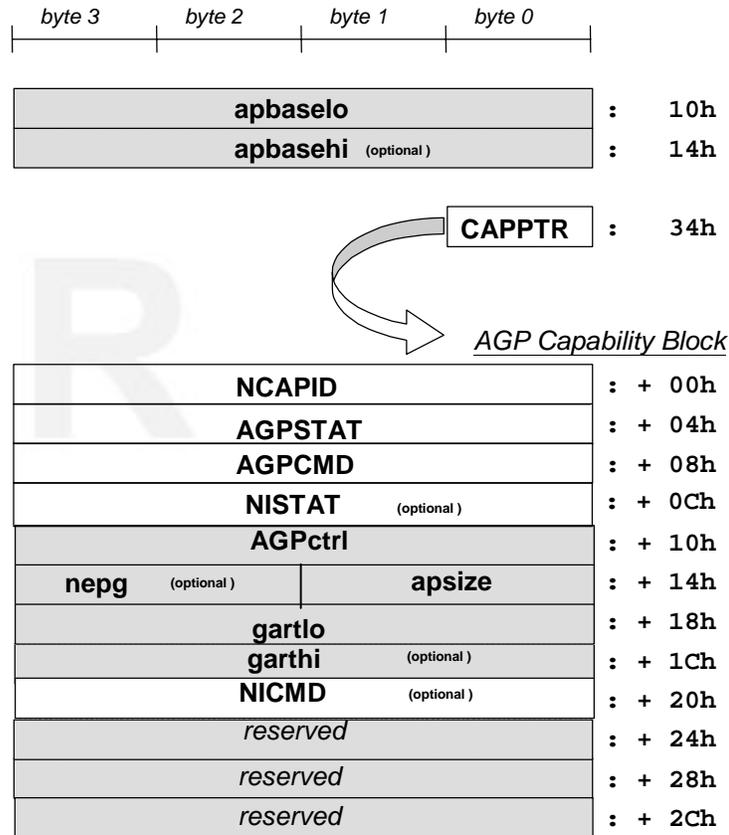
NOTES:

▱ = **Register is in target (core-logic) only. Use of these in the master (graphics chip) is undefined**.

□ = **Register is in both target and master**

Figure 38: AGP3.0 Configuration Register Space

5.6.1 Register Set Support Requirements

A Target or Master may choose to implement just the Core Specification Register set or the enhanced superset described in the Appendix. Even when implementing the superset, the implementation needs to only consider the registers that support the optional features being included in the design. Note that the Core Register Set described in Chapter 2 is a superset of the current AGP2.0 Register Set and is backward compatible from a legacy software perspective. The enhanced register set described here is a superset of both the AGP Core and the AGP2.0 register set. However, compatibility with legacy software environments cannot necessarily be assumed as GART and AGP aperture programming implementations have varied in the past. Furthermore, enabling of any new features described in the Appendices will require new software.

The Major and Minor Revision IDs in the NCAPID register are used to inform software of the register set being implemented. Table 59 shows this relationship.

Table 59: Major/Minor Revisions

Major Revision	Minor Revision	Register Set Support
2h	Don't Care	AGP2.0 Register Support
3h or greater	0h-4h	AGP3.0 Core Specification Register Support
3h or greater	5h-9h	AGP3.0 Appendix Specification Register Support
3h or greater	Ah-Fh	Reserved for future use

5.7 Register Table Format

Each Register is specified using the format below:

Table 60: Register Table

Bits	Access	Field	Description
HighBit : LowBit (Zero-origin)	**<ACCESS>**	*Name*	*Description*
Bit (Zero-origin)	**<ACCESS>**	*Name*	*Description*

The ACCESS column is of the form:

[READ] – [WRITE] – [DEFAULT]

Table 61 defines the value of each sub-field:

Table 61: Sub-field Table

Field	Value	Meaning
[READ]	**R**	Read Allowed
	RZ	Always Read-as-Zero
	R1	Always Read-as-One
	--	Read not allowed
[WRITE]	**W**	Write Allowed
	W1C	Write of "1" clears bit to zero
	IW	Writes Ignored
	MW	Must Write back what is Read (Write-Under-Mask)
	--	Write not allowed
[DEFAULT]	**D'0**	Power-on-Default is Zero
	D'1	Power-on-Default is One
	Dx	Power-on-Default is not specified by AGP
	D'??	Power-on-Default needs to be specified.
	D'xxxx	Power-on-Default is xxxx
	--	Hardwired-Value doesn't need Power-on-default

5.8 Required Master and Target Registers

5.8.1 PCISTS: PCI STATUS REGISTER

Offset: **06h**

Size: 2 bytes

Table 62: Status Register

Bits	Access	Field	Description
15:5			See the *PCI Local Bus Specification.*
4	R-IW	**CAP_LIST**	If the **CAP_LIST** bit is set, the device's configuration space implements a list of capabilities The capability pointer is located at 34h.
3:0			See the *PCI Local Bus Specification.*

5.8.2 CAPPTR: CAPABILITIES POINTER

Offset: **34h**

Size: 1 byte

Table 63: Capabilities Table

Bits	Access	Field	Description
7:0	R-IW	**CAP_PTR**	This field contains a byte offset into the device's configuration space containing the first item in the "capabilities list" and is a *Read Only* register.

5.8.3 NCAPID: AGP IDENTIFIER REGISTER

Offset: **CAPPTR**

Size: 4 bytes

Table 64: Identifier Table

Bits	Access	Field	Description	
31:24	RZ-IW	**Reserved**	Always returns 0 on read; write operations have no effect.	
23:20	R-IW-D'0011	**MAJOR**	Major revision number of AGP3.0 specification this device conforms to. A Value of **3 or greater** Indicates AGP3.0.	
19:16	R-IW-D'0101	**MINOR**	Minor revision number of AGP3.0 interface specification to which this device conforms. Range for minor revision for Appendix (Enhanced) register specifications is 5h-9h. First release will use 5h	
15:8	R-IW-Dx	**NEXT_PTR**	Pointer to next item in capabilities list. Must be NULL for final item in list.	
7:0	R-IW-D'CAPID Value	**CAP_ID**	The value of this field is determined as follows:	
			Location of Registers	CAPID Value
			AGP Target Host –to-PCI Bridge	00000010
			AGP Target PCI-to-PCI Bridge	NEW ID
			AGP Master	00000010

AGP CAP_ID is retained for backward compatibility for configuration registers located in the Host Bridge. A new CAP-ID will be used for configuration registers located in a P2P Bridge. The **NEXT_PTR** field contains a pointer to the next item in the list. The **NEXT_PTR** field in final list item must contain a NULL pointer.

5.8.4 AGPSTAT: AGP STATUS REGISTER

Offset: CAPPTR + 04h

Size: 4 bytes

Table 65: Status Register

Bits	Access	Field	Description
31:24	MST: R1-IW TGT: R-IW-Dx	**RQ**	**TARGET ONLY:** The RQ field contains the maximum number of AGP3.0 command requests (both asynchronous and isochronous) that can be enqueued to the target. "0" means a depth of 1 entry, while 0xFF means a depth of 256 entries.
23:18	RZ-IW	**Reserved**	Always returns 0 when read; write operations have no effect.

17	R-IW-Dx	**ISOCH SUPPORT**	**TARGET & MASTER:**
			'0' = Isoch transactions not supported
			'1' = Isoch transactions supported
			Note:
			   1. If '0', optional NISTAT and NICMD registers are not implemented and should be ignored by software.
			   2. Isoch is not supported when 4x speed operation is selected. When this happens this bit will be forced by hardware to 0

16	RZ-IW	**Reserved**	Always returns 0 when read; write operations have no effect.
15:13	MST: RZ-IW TGT: R-IW-Dx	**ARQSZ**	**TARGET ONLY**: LOG2 of the optimum asynchronous request size in bytes minus 4 to be used with the target. The MASTER should attempt to issue a group of sequential back-to-back asynchronous requests that total to this size and for which the group is naturally aligned to. Optimum_request_size = 2 ^ (ARQSZ+4) If ARQSZ is zero, then the target has no recommendation.
12:10	R-IW-Dx	**CAL_Cycle**	**MASTER & TARGET:** Specifies required period for core-logic initiated bus cycle for calibrating I/O buffers. | CAL_CYCLE	Period
---	---
000	4 ms
001	16ms
010	64 ms
011	256 ms
100-110	Reserved for future use
111	Calibration Cycle Not Needed	
9	R1-IW	**Reserved (SBA)**	Always returns 1 when read; write operations have no effect. AGP3.0 devices are required to support side band addressing, even when operating in AGP 2.0 compatibility mode.
8	MST: RZ-IW TGT: R-IW-Dx	**ITA_COH**	**TARGET ONLY**: When '1', the AGP3.0 target ensures that accesses from an AGP Master are made coherent with host processor caches and other caching agents in the system if the following are met: Access is Inside The AGP aperture (ITA) Access has corresponding gart_entry.coh bit set
7	MST: RZ-IW TGT: R-IW-Dx	**GART64B**	**TARGET ONLY**: When '1', Core-logic can support 64-bit and 32-bit GART entries; when '0', only 32-bit GART entries are supported.
6	MST: RZ-IW TGT: R-IW-Dx	**htrans#**	**TARGET ONLY:** When '0', Core-logic can translate host processor accesses through the AGP aperture. **NOTE:** *Core-logic translation of*

Bits	Access	Field	Description
			host processor accesses is a platform specific feature and is solely used for the purpose of supporting legacy OS. Drivers written to use this feature may not port to platforms that do not have legacy support requirements.
5	R-IW	**OVER4G**[27]	If set, this device supports addresses greater than 4 GB.
4	R-IW	FW	**MASTER & TARGET:** If set to a 1, this Master or Target supports Fast Writes. Fast-Write support is optional for both Master and Target.
3	R-IW	**AGP3.0_MODE**	'1' = AGP3.0 Mode and '0' = AGP Mode; Set by hardware on power-up reset, see section 2.4.2 for details. Note that when AGP3.0_MODE = 0, the RATE field (AGPSTAT [2:0]) and DRATE field (AGPCMD[2:0]) in both the master and target must function as defined by the AGP Interface Specification V2.0 for compatibility with existing software.
2:0	R-IW	**RATE**	Data Rate Support (RATE) - <table><tr><td>AGPSTAT[3]</td><td>Code</td><td>Speed Supported</td></tr><tr><td>0</td><td>xxx</td><td>See AGP2.0 Specs</td></tr><tr><td>1</td><td>001</td><td>4x</td></tr><tr><td>1</td><td>010</td><td>8x</td></tr><tr><td>1</td><td>011</td><td>4x, and 8x</td></tr><tr><td>1</td><td>All other codes</td><td>Reserved</td></tr></table>

---

[27] If Target (Core-Logic) sets OVER4G it must support >32bit address PCI cycles generated by the Master using DAC. See PCI Local Bus Specification V2.2 for details. However, support of outbound DAC transactions from the core-logic to the graphics card is optional.

5.8.5 AGPCMD: AGP COMMAND REGISTER

Offset: **CAPPTR + 08h**

Size: 4 bytes

Table 66: Command Register

Bits	Access	Field	Description

31:24	MST: R-W-D'0
	TGT: RZ-IW
		PRQ	**Master:** The **PRQ** field must be programmed with the maximum number of AGP command requests (both asynchronous and isochronous) that the master is allowed to "enqueue" in the target. "0" means a depth of one entry, while FFh means a depth of 256 entries.
			
			For a given AGP Port, the sum of all Masters' adjusted AGPCMD.PRQ must be less than or equal to the target's adjusted AGPSTAT.RQ:
			
			`(TARGET.RQ +1) >=(MASTER[i].PRQ + 1)  +`
			`                    (MASTER[i+1].PRQ + 1) +`
			`...`
			
			Target: IGNORED.

23:17	RZ-MW	**Reserved**	Always returns 0 when read; write operations have no effect.
16	RZ-MW	**Reserved**	Always returns 0 when read; write operations have no effect. Future use reserved.
15:13	MST: R-W-D'0 TGT: RZ-IW-D'0	**PARQSZ**	**MASTER ONLY**: Programmed based on ARQSZ in TARGET's AGPSTAT[15:13]. LOG2 of the optimum asynchronous request size in bytes minus 4 to be used with the target. The MASTER should attempt to issue a group of sequential back-to-back asynchronous requests that total to this size and for which the group is naturally aligned to. **Optimum_request_size = 2 ^ (PARQSZ+4)** If ARQSZ is zero, then the target has no recommendation.
12:10	MST: RZ-IW TGT: R-W-D'000	**PCAL_Cycle**	**TARGET ONLY**: Programmed with period for core-logic initiated bus cycle for calibrating I/O buffers for both master and target. The default value is based on chipset requirements. This value is updated with the smaller of the value in CAL_CYCLE from Master's/TARGET's AGPSTAT. The translations of the encoding remain the same as in CAL_CYCLE. PCAL_CYCLE is set to 111 by s/w when both Master and Target have AGPSTAT.CAL_CYCLE = 111.
9	R-W-D'0	**Reserved (SBA_ENABLE)**	Reserved. This must be a read/write bit for software compatibility with AGP interface specification V2.0. AGP3.0 devices are required to support side band addressing, so this must be set to 1 'b by the software when AGPSTAT[3] = 1.

Bits	Access	Field	Description
8	R-W-D'0	**AGP_ENABLE**	**Master:** Setting the **AGP_ENABLE** bit allows the master to initiate AGP operations. When cleared, the master cannot initiate AGP operations.
			Target: Setting the **AGP_ENABLE** bit allows the target to accept AGP operations. When cleared, the target ignores incoming AGP operations. Notes:
			1. The target must be completely configured and enabled before the master is programmed.
			2. A device can make no assumptions as to the sequence of command field programming, except that **AGP_ENABLE** is the last bit set. Concurrently setting all command fields and the **AGP_ENABLE** bit using a single 32-bit write is also permitted.
			3. This bit must not be cleared in the target until it has been cleared in the master. There should be no outstanding SBA requests in the target when the bit is cleared.
			The **AGP_ENABLE** bit is cleared by **power-on-reset**.
7	MST: RZ-IW TGT: R-W-D'0	**GART64B**	**TARGET ONLY:** When '1', Core-logic supports 64-bit GART entries; when '0' Core-logic supports 32-bit GART entries.
6	RZ-MW	**Reserved**	Always returns 0 when read; write operations have no effect.
5	R-W-D'0	**OVER4G**	**Master:** Setting the **OVER4G** bit allows the master to initiate AGP3.0 Requests to addresses above the 4 GB address boundary. When cleared, the master is only allowed to access addresses in the low 4 GB of the address space.
			Target: Setting the **OVER4G** bit enables the target to accept a Type 4 command and to utilize **A[35::32]** for a Type 3 command.
4	R-W-D'0	**FW_ENABLE**	**MASTER & TARGET**: When set to 1, FW is enabled in Master or Target.
3	RZ-MW	**Reserved**	Always returns '0' on reads. Writes are ignored.
2:0	R-W-D'000	**DRATE**	Data Rate Enable (DRATE) - The setting of these bits determines the data transfer rate. One (and only one) bit in this field must be set to indicate the desired data transfer rate. The same bit must be set on both master and target. The encoding below assumes AGP3.0_Mode is set in AGPSTAT

```
Encoding     Meaning
  001  4X Data Transfer Mode
  010  8X Data Transfer Mode
  100  Reserved
```

5.8.6 NISTAT: AGP ISOCHRONOUS STATUS REGISTER

This register is accessible in the Target only if the Target's AGPSTAT.isoch_support bit is set. Similarly, the register is accessible in the Master only if the Master's AGPSTAT.isoch_support bit is set. When the isoch_support bit is not set, reads to this register will return undefined data and writes will have no effect.

> **Offset:**   **CAPPTR + 0Ch**
>
> **Size:**   4 bytes

Table 67: Status Register

Bits	Access	Field	Description
31-24	RZ-IW	**Reserved**	Reserved. Reads return zeros; writes are ignored

23:16	R-IW-Dx	**MAXBW**[28]	Maximum Bandwidth (both asynchronous and isochronous) of the device per us in units of 32 bytes. If this field is all "0s," it should be ignored.
			
			    TARGET: Maximum bandwidth that the target can support.
			    MASTER: Maximum bandwidth required by the Master.

15:8	R-IW-Dx	**ISOCH_N**	Maximum number of isochronous transactions in a single isochronous period. **Target:** This is the maximum that the target can accept at its corresponding isochronous payload size (NICMD.PISOCH_Y). Must be at least 1. In the target, each ISOCH "Y" has a corresponding ISOCH "N". The value reported in NISTAT.ISOCH_N is controlled by the value in AGPCMD.PISOCH_Y (which defaults to NISTAT.ISOCH_Y). After software configures the target's NICMD.PISOCH_Y, it can then read the target's corresponding ISOCH "N" here. **Master:** This is the maximum number required by the master at its corresponding isochronous payload size (ISOCH_Y).
7:6	R-IW-Dx	**ISOCH_Y**	**Target and Master:** Isochronous payload sizes supported. Encoded as: '00' 32bytes, 64bytes, 128bytes, 256bytes '01' 64bytes, 128bytes, 256bytes '10' 128bytes, 256bytesand greater '11' 256bytes

---

[28] MAXBW should be used by configuration software for the allocation of PISOCH_N and PRQ parameters when dividing the available bandwidth among several devices.

Bits	Access	Field	Description
31-24	RZ-IW	**Reserved**	Reserved. Reads return zeros; writes are ignored
5:3	MST: RZ-IW TGT: R-IW-Dx	**ISOCH_L**	**Target:** Maximum isochronous data transfer latency in units of isochronous periods (1 us). Allowable values are 1 to 5. **Master:** IGNORED
2	RZ-IW	**Reserved**	Reserved
1:0	R-W1C/D'0	**Isoch-ErrorCode**	TARGET: 00 = No Error/ 01 = Isoch Req Overflow[29]/ 10 = Reserved/ 11 = Reserved MASTER: 00 = No Error/ 01 = Read Buffer under-run[30]/ 10 = Write Buffer Overflow[31]/ 11 = Reserved

The calculation of NICMD.PISOCH_Y, NICMD.PISOCH_N, and NICMD.PRQ are very dependent upon fields in AGP_STAT and NISTAT. Refer to Section 4.1.10 for possible methods in calculating these fields.

---

[29] Indicates that the number of isochronous requests from master has exceeded isoch request queue in target.

[30] Indicates that the target has failed to return isochronous read data in a timely manner resulting in a read buffer under-run.

[31] Indicates that the target has failed to fetch isochronous write data from the master in a timely manner resulting in this error.

5.8.7 NICMD: AGP ISOCHRONOUS COMMAND REGISTER

This register is accessible in the Target only if the Target's AGPSTAT.isoch_support bit is set. Similarly, the register is accessible in the Master only if the Master's AGPSTAT.isoch_support bit is set. When the isoch_support bit is not set, reads to this register will return undefined data and writes will have no effect.

Offset: **CAPPTR + 20h**

Size: 2 bytes

Table 68: Command Register

Bits	Access	Field	Description
15:8	MST: R-W-D'0 TGT: RZ-IW	**PISOCH_N**	**MASTER:** Software programs this field with the maximum number of isochronous transactions that the master is allowed to request from the target in a single isochronous period. It includes both Read and Write requests. **Target:** IGNORED
7:6	R-W	**PISOCH_Y**[32]	Software programs this field with the Isochronous payload size to be used by all AGP3.0 devices; software ensures it is the same value for all AGP3.0 devices. It should be set to the smallest size (NISTAT.ISOCH_Y) that is supported by all devices AND satisfies the bandwidth requirements for all ports. It is encoded in the same way as NISTAT.ISOCH_Y. **Master:** No default value. **Target:** Default value is NISTAT.ISOCH_Y In the target, each ISOCH "Y" has a corresponding ISOCH "N". The value reported in NISTAT.ISOCH_N is controlled by the value in AGPCMD.PISOCH_Y. After software configures the target's AGPCMD.PISOCH_Y, it can then read the target's corresponding ISOCH "N" in NISTAT.ISOCH_N.
5:0	RZ-IW	**Reserved**	Reserved

The calculation of NICMD.PISOCH_Y, NICMD.PISOCH_N, and NICMD.PRQ are very dependent upon fields in AGPSTAT and NISTAT.   Refer to Section 4.1.10 for possible methods in calculating these fields.

---

[32] PISOCH_Y applies to Isoch Read and Writes. For Isochronous Writes, the AGP Master must queue 2 and 4 consecutive 64 byte requests to handle PISOCH_Y of 128 and 256 bytes respectively.

5.9 Required Target Registers

5.9.1 APBASELO: AGP APERTURE BASE ADDRESS REGISTER

Offset: **10h**

Size: 4 bytes

Table 69: Address Register

Bits	Access	Field	Description
31:22	R-W-D'0	**AddressLo**	Address[31:22] of the AGP3.0 aperture address*.* ***The actual AGP Aperture Base Address is determined by using this register along with the Aperture size programmed in APSIZE register. APSIZE[11:8] and APSIZE[5:0] are used to force a '0' into corresponding bits in APBASE[31:28] and APBASE[27:22] such that the AGP3.0 aperture is always naturally aligned for a given aperture size. See 5.3.2.***
21:4	RZ-IW	**Zero**	Bits are hardwired to zero. This forces minimum AGP3.0 aperture size to be 4MB or greater.
3	R1-IW	**Prefetchable**	Hardwired to '1' to identify the Graphics AGP3.0 aperture range as prefetchable -- i.e. *there are no side effects on reads, the device returns all bytes on reads regardless of the byte enables, and Core-logic may merge host processor writes into this range without causing errors.*
2:1	R-IW-Dx	**Type**	AGP3.0 allows the target to support either a 64-bit or a 32-bit Base Address Register for APBASE. This is encoded as: '00' 32-bit Base Address Register; AGP3.0 aperture can be located anywhere within a 32-bit address space '10' 64-bit Base Address Register ; AGP3.0 aperture can be located anywhere within a 64-bit address space
0	RZ-IW	**Memory**	Hardwired to zero to indicate that the Graphics AGP3.0 aperture must reside in "Memory" space – as defined by the PCI Local Bus Specification.

5.9.2 APBASEHI: AGP APERTURE BASE ADDRESS HIGH REGISTER

This register is only available when AGPSTAT.over4G bit is set. . When the AGPSTAT.over4G bit is not set, reads to this register will return undefined data and writes will have no effect.

Offset: **14h**

Size: 4 bytes

Table 70: Base Address

Bits	Access	Field	Description
31:0	R-W-D'0	**AddressHi**	Address[63:32] of the AGP aperture address*. **Used only if APBASELO[2:1] select a 64-bit Base Address Register; otherwise this register is ignored.**

5.9.3 AGPCTRL: AGP CONTROL REGISTER

Offset: **CAPPTR + 10h**

Size: 4 bytes

Table 71: Control Register

Bits	Access	Field	Description
31:24	RZ-MW	**Reserved**	Reserved For Future AGP3.0 Architectural Core-logic Features
23:16	R-MW-Dx	**Reserved**	Reserved For Implementation-Specific Core-logic Features
15:10	RZ-MW	**Reserved**	Reserved For Future AGP3.0 Architectural Core-logic Features
9	R-W-D'0	**CAL_CYCLE_DIS**	When set to '1', calibration cycle operation is disabled by the core-logic. Note that calibration cycle should be automatically disabled by core-logic when not in AGP3.0 mode of operation (see AGPSTAT[3])
8	R-W-D'0	**APERENB**	This bit controls the enabling of the graphics AGP aperture for the AGP3.0 Port. When this bit is '1', the AGP aperture is enabled; when '0' the AGP aperture is disabled.
7	R-W-D'0	**GTLBEN**[33]	GTLB Enable (GTLBEN) - **This bit is available only if the core-logic implements a Graphics Translation Look-aside Buffer (GTLB). If no GTLB is implemented this bit is hardwired to '0'.** When set to 1 this bit enables normal operations of the Graphics Translation Lookaside Buffer. If it is zero (default) the GTLB is flushed by clearing the valid bits associated with each entry. In this mode of operation all accesses that require translation bypass the GTLB. All requests that are positively decoded to the graphics AGP aperture force the Core-logic to access the translation table in main memory before completing the request. Translation table entry fetches will not be cached in the GTLB.
6:0	R-MW-Dx	**Reserved**	Reserved For Implementation-Specific Core-logic Features

5.9.4 APSIZE: AGP APERTURE SIZE

Offset: **CAPPTR + 14h**

Size: 2 bytes

Table 72: Aperture

Bits	Access	Field	Description
15:12	RZ-MW	**Reserved**	Reserved
11:0	R-W-D'111100000000	**APSIZE**	APSIZE[11:0] determines the status of APBASE[31:22] as follows: APSIZE[n] = 0 forces APBASE[22+n] to 0 when 0<=n=<5 APSIZE[n] = 0 forces APBASE[22+(n-2)] to 0 when 8<=n=<11 APSIZE[n] = 1 allows APBASE[22+n] is R/W programmable when 0<n<5 APSIZE[n] = 1 allows APBASE[22+(n-2)] is R/W programmable when 8<=n=<11 For legacy compatibility, APSIZE[7:6] should always be 00 and are not used. APSIZE[11:0] have the following legal codes: Bits: 11 10 9 8 7 6 5 4 3 2 1 0 Aperture Size 1 1 1 1 0 0 1 1 1 1 1 1 4MB 1 1 1 1 0 0 1 1 1 1 1 0 8MB 1 1 1 1 0 0 1 1 1 1 0 0 16MB 1 1 1 1 0 0 1 1 1 0 0 0 32MB 1 1 1 1 0 0 1 1 0 0 0 0 64MB 1 1 1 1 0 0 1 0 0 0 0 0 128MB 1 1 1 1 0 0 0 0 0 0 0 0 256MB Default 1 1 1 0 0 0 0 0 0 0 0 0 512MB 1 1 0 0 0 0 0 0 0 0 0 0 1024MB 1 0 0 0 0 0 0 0 0 0 0 0 2048MB 0 0 0 0 0 0 0 0 0 0 0 0 4096MB A given core logic implementation may have a partial range of aperture sizes from the above table. The following rules apply: • There must be a single contiguous range of aperture sizes from the table above. –i.e. no gaps. • Max Aperture Size: If this is different from the max in the above table, the core logic must hardwire the "1s" for the APSIZE encoding for its maximum aperture

Bits	Access	Field	Description
			size. For e.g. if core logic supports 256MB max, it must hardwire APSIZE[11:8] to '1'. Writes to these bits will have no effect.
			• Min Aperture Size: If this is different from the table above, the core logic must hardwire the "0s" in the encoding for its min aperture size. For e.g. if min supported size is 16MB, APSIZE[1,0] must be hardwired to '0. Write will have no effect.
			The above scheme forces the aperture to be always naturally aligned to the selected aperture size.

5.9.5 NEPG: AGP ENABLED APERTURE PAGE SIZE

Offset: **CAPPTR + 16h**

Size: 2 bytes

Table 73: Enabled Page

Bits	Access	Field	Description
15:12	R-W-D'0000[20]	**SEL**	Size Select (SEL) selects the AGP aperture Page Size from field **N.** For example, if SEL = 0, N=0 which is default. If the following requirements are not met for the value of NEPG.SEL it is unpredictable which AGP aperture Page size, if any, is selected: • SEL ranges from b'0000 to b'1010 (all other values undefined) • NEPG[SEL}] must be '1'.
11	RZ-MW	**Reserved**	Reserved
10:0	R-IW	**N**	NEPG[bit N] indicates whether the Core-logic implementation supports an AGP3.0 aperture Page size of **(2 ^ (N+12))**. If NEPG[bit N] is '1', then the corresponding AGP aperture page size is supported; if '0' then the size is not supported. For example, if NEPG bit 7 is set, then the implementation supports an AGP aperture page size of of (2 ^ (7+12)) or 512KB. NEPG[6] must be '1' by all implementations to support 4 KB page size. The largest page size supportable is 4MB. 4KB page must always be supported. This means NEPG[0] must = 1 for all core-logic implementations.

---

[20] If an implementation supports only 1 page size (4KB) , then this field may be implemented as Read-Only.

5.9.6 GARTLO: AGP GART POINTER

Offset: **CAPPTR + 18h**

Size: 4 bytes

Table 74: GART Pointer

Bits	Access	Field	Description
31:12	RW-D'0	**baselo**	Bits[31:12] of the start of the Graphics AGP aperture Remapping Table (GART). As a minimum the GART must be aligned on a 4KB boundary
11:4	RZ-IW	**Reserved**	Reserved
3:0	RZ-IW	**Reserved**	Reserved

5.9.7 GARTHI: AGP GART POINTER HIGH

Offset: **CAPPTR + 1Ch**

Size: 4 bytes

Table 75: AGP Gart Pointer

Bits	Access	Field	Description
31:0	R-W-D'0	**basehi**	Bits[63:32] of the start of the Graphics AGP aperture Remapping Table (GART). Software writes all zeroes in this register for systems that only support 32-bit physical addresses.

6 Appendix C: Glossary of Terms

Term	Definition
Common Clock Transfer	A transfer across an interface between two agents that is synchronized to a clock that is common to both and is matched in phase and frequency.
Core-logic	The core-logic that has the host platform's AGP3.0 bridge.
Motherboard	This is a general term referring to the PC board that includes the components of the Core-logic and the AGP connector.
MT/s	Million Transfers/second where the size of the transfer is determined by the width of the interface. So, in AGP3.0 the data bus does a maximum of (8x)533 MT/s, which on a 4-byte interface translates to 2 Gbytes/s.
AGP3.0 Card, Graphics Card	These refer to the Graphics Card that is installed into the AGP3.0 Connector.
AGP3.0 Master, AGP3.0 Device	The graphics chip that initiates AGP transactions.
AGP3.0 Mode	The mode in which the AGP3.0 signaling and features are enabled.
AGP3.0 Target	The core-logic during any AGP transaction. The graphics chip can also function as target during core-logic initiated PCI transactions.
AGP3.0 Transactions	Transactions on the AGP Interface using the AGP protocol semantics initiated by the AGP Master using the Side-Band Address Interface. The core-logic initiated Fast-Write transaction also falls under the category of AGP Transaction.
DBI	Dynamic Bus Inversion. Scheme using dynamic inverting of data bits based on transition history in order to limit the maximum number of simultaneous transitions.
PCI Transactions	Any transaction on either the AGP or PCI interfaces that use the PCI protocol semantics.
Source Synchronous Transfer	A transfer across an interface between two agents where the sender transmits a strobe along with the data to capture and hold the data at the receiver.
Universal AGP3.0 Card	A graphics card that supports both AGP3.0 and AGP1.5 V signaling and protocols.
Universal AGP3.0 Motherboard	A motherboard that supports both AGP3.0 and AGP1.5 V signaling and protocols.
Block-Level Flow Control	This is used by the AGP Master to control the flow of data to or from the AGP target during the actual data transfer phase. The mechanism used is suppression of TRDY during cycles known as "throttle points" in the data phase. For a description of throttle points refer to the AGP2.0 interface specification.
Buffer-Full Flow Control	This is used by the AGP Master or Target (in the case of Fast Writes) to stall the start of new data transfer transaction. The signals used for this purpose are RBF and WBF. For a detailed description of using RBF and WBF refer to the AGP2.0 interface specification.