42.2.3.5 中断悬挂寄存器（FGGB_INTPENDING）

当 CPU 从 3D 图形内接收一个中断时，CPU 必须调查 3D 图形内的哪个功能块产生中断。CPU 可以通过读取 FGGB_INTPENDING 找出产生中断的模块。

在从 3D 图形清除中断的服务惯例内，必须向 FGGB_INTPENDING 写入任何值。通过向 FGGB_INTPENDING 写入值，FGGB_INTPENDING 将会自动清零，3D 图形可以产生另一个中断。向 FGGB_INTPENDING 写入的数值是什么并不重要，凡是写入 FGGB_INTPENDING 的写操作都会清除此值。

当前，FGGB_PIPESTATE 在 HI 内可以产生中断。一旦 3D 图形产生一个中断，在不读取 FGGB_INTPENDING 的情况下，CPU 会知道 FGGB_PIPESTATE 是中断源。

寄存器	地址	读/写	描述	复位值
FGGB_INTPENDING	0x72000040	读/写	中断悬挂寄存器	0x00000000

FGGB_INTPENDING	位	描述	初始状态
Major	[31:1]	主要版本	0
Pipeline-State	[0]	读：产生管道状态中断 1= 发生中断， 0=没有中断 写：清除值到 0.写入的数值并不重要	0b

42.2.3. 6 中断屏蔽寄存器（FGGB_INTMASK）

FGGB_INTMASK 可以使 3D 中断的信号使能或禁止。当前，只有通过 HI 才能产生中断。因此，FGGB_INTMASK 的 LSB 用于使中断使能或禁止。

注：有另一种从管道状态禁止中断的方法。当 FGGB_PIPEMASK 是位运算控制时，FGGB_INTMASK 是全局控制。

寄存器	地址	读/写	描述	复位值
FGGB_INTMASK	0x72000044	读/写	使中断能活不能	0x00000000

FGGB_INTMASK	位	描述	初始状态
Reserved	[31:1]	保留	0
Pipeline-State	[0]	管道状态产生中断 1=使能中断 0=使中断不能	0b

42.2.3.7 管道屏蔽寄存器（FGGB_PIPEMASK）

FGGB_PIPEMASK 指定产生中断的 3D 图形模块。

寄存器	地址	读/写	描述	复位值
FGGB_PIPEMASK	0x72000048	读/写	指定 3D 图形内用于产生中断的块。每个块的位位置与 FGGB_PIPESTATE 相同	0x00000000

FGGB_PIPEMASK	位	描述	初始状态
Reserved	[31:17]	保留	0
PF0	[16]	0b=不考虑 1b=用于产生中断	0b
Reserved	[15:13]	保留	0
PS0	[12]	0b=不考虑 1b=用于产生中断	0b
Reserved	[11]	保留	0
RA	[10]	0b=不考虑 1b=用于产生中断	0b
TSE	[9]	0b=不考虑 1b=用于产生中断	0b
PE	[8]	0b=不考虑 1b=用于产生中断	0b
Reserved	[7:5]	保留	0
VS	[4]	0b=不考虑 1b=用于产生中断	0b

VC	[3]	0b=不考虑 1b=用于产生中断	0b
HVF	[2]	0b=不考虑 1b=用于产生中断	0b
HI	[1]	0b=不考虑 1b=用于产生中断	0b
HOSTFIFO	[0]	0b=不考虑 1b=用于产生中断	0b

42.2.3. 8 管道目标状态寄器（FGGB_PIPETGTSTATE）

如前所述，FGGB_PIPEMASK 定义产生中断的 3D 图形模块。FGGB_PIPETGTSTATE 定义中断发生时的管道状态值。需要注意的是 FGGB_PIPEMASK 内的 0 值模块的 FGGB_PIPEGTSTATE 值可忽略。

寄存器	地址	读/写	描述	复位值
FGGB_PIPETGTSTATE	0x7200004C	读/写	指定发生中断时的管道状态值	0x00000000

FGGB_PIPETGTSTATE	位	描述	初始状态
Reserved	[31:17]	保留	0
PF0	[16]	0b=当 PF0 不工作时中断（空） 1b=当 PF0 工作时中断（非空）	0b
Reserved	[15:13]	保留	0
PS0	[12]	0b=当 PS0 不工作时中断（空） 1b=当 PS0 工作时中断（非空）	0b
Reserved	[11]	保留	0
RA	[10]	0b=当 RA 不工作时中断（空） 1b=当 RA 工作时中断（非空）	0b
TSE	[9]	0b=当 TSE 不工作时中断（空） 1b=当 TSE 工作时中断（非空）	0b
PE	[8]	0b=当 PE 不工作时中断（空）	0b

		1b=当 PE 工作时中断（非空）	
Reserved	[7:5]	保留	0
VS	[4]	0b=当 VS 不工作时中断（空） 1b=当 VS 工作时中断（非空）	0b
VC	[3]	0b=当 VC 不工作时中断（空） 1b=当 VC 工作时中断（非空）	0b
HVF	[2]	0b=当 HI 和 VS 间的 FIFO 为空时中断 1b=当 HI 和 VS 间的 FIFO 为非空时中断	0b
HI	[1]	0b=当 HI 不工作时中断（空） 1b=当 HI 工作时中断（非空）	0b
HOSTFIFO	[0]	0b=当主机 FIFO 不工作时中断（空） 1b=当主机 FIFO 工作时中断（非空）	0b

42.2.3.9 管道中断状态寄存器（FGGB_PIPEINTSTATE）

当发生中断时，FGGB_PIPEINTSTATE 捕捉管道状态。当同时发生几个中断时，FGGB_PIPEINTSTATE 掌握第一个管道状态。

需要注意的是 FGGB_PIPEINTSTATE 取决于 FGGB_PIPEMASKE。

寄存器	地址	读/写	描述	复位值
FGGB_PIPEINTSTATE	0x72000050	读	发生几个中断时，掌握第一个管道状态	0x00000000

FGGB_PIPEINTSTATE	位	描述	初始状态
Reserved	[31:17]	保留	0
PF0	[16]	0b=发生中断时，PF0 为空 1b=发生中断时，PF0 非空	0b
Reserved	[15:13]	保留	0
PS0	[12]	0b=发生中断时，PS0 为空 1b=发生中断时，PS0 非空	0b
Reserved	[11]	保留	0

RA	[10]	0b=发生中断时，RA 为空 1b=发生中断时，RA 非空	0b
TSE	[9]	0b=发生中断时，TSE 为空 1b=发生中断时，TSE 非空	0b
PE	[8]	0b=发生中断时，PE 为空 1b=发生中断时，PE 非空	0b
Reserved	[7:5]	保留	0
VS	[4]	0b=发生中断时，VS 为空 1b=发生中断时，VS 非空	0b
VC	[3]	0b=发生中断时，VC 为空 1b=发生中断时，VC 非空	0b
HVF	[2]	0b=发生中断时， HI 和 VS 间的 FIFO 为空 1b=发生中断时， HI 和 VS 间的 FIFO 为非空	0b
HI	[1]	0b=发生中断时，HI 为空 1b=发生中断时，HI 非空	0b
HOSTFIFO	[0]	0b=发生中断时，主机 FIFO 为空 1b=发生中断时，主机 FIFO 非空	0b

42.3.主机接口

42.3.1.概述

主机接口单元的主要功能是接收 CPU 内转换为浮点数据格式的数据。主机接口可以向 CPU 转换状态数据（SFR 值， VS isnt-memory 等等）。

根据数据的特点，对从 CPU 向 3D 图形转换的数据进行分类，主要可分为状态数据和几何数据。CPU 首先设置状态数据，然后向 3D 图形转换几何数据。3D 图形用状态数据返回转换的几何数据。对于被返回

的下一个几何数据，相应的状态数据必须转换给 3D 图形。新的状态数据可以影响到先前的几何数据。因此，只有当状态数据不影响先前几何数据时才可以向 3D 图形转换状态数据。管道状态寄存器（FGGB_PIPESTATE）在前几章内有介绍， 用 FGGB_PIPESTATE 询问 3D 图形内处理几何数据的地点。假设 3D 图形内有 A, B, C 和 D 四个模块，C 块的状态数据已被更新。如果 B 块处理几何数据， CPU 不更新 C 块的数据，因为 C 块新的状态可以影响到 B 块的几何数据。CPU 必须等到 B 块的几何数据穿过 C 块且到达 D 块时，CPU 才可以更新 C 块的状态，因为 A, B 和 C 块都为空。

如果帧缓冲区的内容应用于纹理，CPU 必须向纹理复制帧缓冲区的数据。下面部分描述了如何向主机接口提供几何数据。

42.3.2.操作模式

主机接口可以用两种方法决定如何转换几何数据：索引方法和非索引方法。

索引方法：CPU 向主机接口内的顶点缓冲区储存几何数据后，CPU 向储存的几何数据转换索引。这个方案消耗低总线的带宽。

非索引方法：CPU 直接转换几何数据。当顶点缓冲区内没有空间时这种方案非常有效。很少使用此种方法向主机接口转换几何数据。

42.3.3.索引模式

主机接口内索引模式使用顶点缓冲区。索引模式有两种操作方式：自动增加方式和索引转换方式。注意：所有的几何数据必须在顶点缓冲区内。索引范围与 VB 的尺寸有关：索引计算的 VB 地址必须是有效的 VB 地址。因此，应用程序中必须小心的控制索引。

自动增加模式（FGHI_CONTROL.EnVB=1,FGHI_CONTROL.AutoInc=1）：CPU 发送两个 DWORD，一个表示计数和一个表示索引。主机接口使用顶点缓冲区内被转换的索引，然后下一个索引将自动计算；FGHI_IDXOFFSET.VALUE 增加先前的索引数值（通常情况下设置为 0），这个进程重复计数。每对 DWORD 表示索引的设置。因此，这种方法在转换几何数据中可有效的提高性能。自动增加模式如图 42-4 所示。

图 42-4 自动增加模式

索引转换模式（FGHI_CONTROL.EnVB=1,FGHI_CONTROL.AutoInc=0）：在索引转换模式下，CPU 发送个人索引。在 CPU 发送索引计数的数目后，将转换一个随机索引设置。这些索引用于索引顶点缓冲区。索引转换模式如图 42-5 所示。

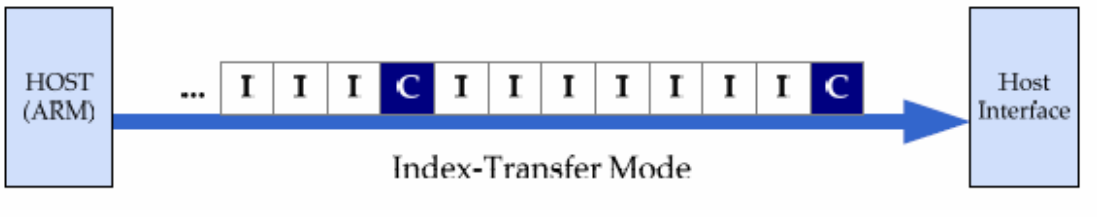

图 42-5 索引转换模式

非索引模式（FGHI_CONTROL.EnVB=0,FGHI_CONTROL.AutoInc=1）：在非索引模式下，不使用顶点缓冲区。CPU 发送所有的几何数据。与索引模式一样，必须首先转换顶点数和索引。在这种情况下，索引使用模拟数值（0xFFFFFFFF）.非索引模式如图 42-6 所示。

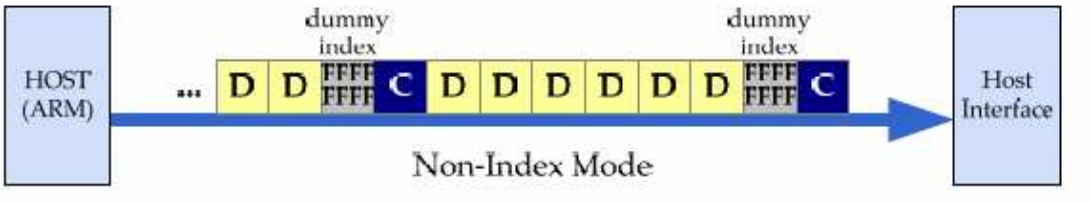

图 42-6 非索引模式

42.3.4.如何设置主机接口的特殊寄存器

主机接口的特殊寄存器设置以后，可以从 CPU 或顶点缓冲区内转换几何数据。几何数据是顶点着色器输入属性的一个设置。换句话说，顶点的属性序号可以从 CPU 或从顶点缓冲区内转换。在

FGHI_ATTRIB0~FGHI_ATTRIB9 内定义一个顶点属性的构成设置。可以通过 FGHI_ATTRIBn.NumComp 决定每个属性组成部分的数量。FGHI_ATTRIBn 内的最后属性位表示 FGHI_ATTRIBn 是否被使用。例如，FGHI_ATTRIB0.LastAtrr= FGHI_ATTRIB1.LastAtrr = FGHI_ATTRIB2.LastAtrr=0，FGHI_ATTRIB3.LastAtrr=1 的意思是顶点由四个属性组成。

只有 CPU 可以转换顶点的数量。（计数写入主机接口的 FGHI_DWENTRY）。CPU 可以向同一个 FGHI_DWENTRY 内转换索引或新的几何数据。顶点缓冲区只能向主机接口提供几何数据。顶点缓冲区的用途由 FGHI_CONTROL.EnVB 决定。如果使用了顶点缓冲区，将需要索引来索引查找顶点缓冲区内的几何数据。需要的索引可以通过 CPU 发送，或在主机接口内产生。在 CUP 的 DWORD 内有一些索引取决于 FGHI_CONTROL.IdxType(索引数据类型)。在下列途径中应用转换的或产生的索引。

```
1.  Get a count from CPU
2.  Get an index from CPU or previous index;   // depending on FGHI_CONTROL.AutoInc. In Non-Index mode, 0xFFFFFFFF is used.
3.  Add FGHI_IDXOFFSET.VALUE to the index   for Index mode // In Non-Index mode, this step is skipped.
4.  for each n from 0x0 to 0xF
5.      if(FGHI_CONTROL.EnVB == 1 && FGHI_ATTRIB[n]_VBCTRL.Range != 0 && index < FGHI_ATTRIB[n]_VBCTRL.Range)
6.          Use DWORDs fetched from
                VertexBuffer[FGHI_ATTRIB[n]_VBBASE.Addr + index*FGHI_ATTRIB[n]_VBCTRL.Stride];   // Index mode
7.      else
8.          Fetch DWORDs from CPU and use them as the geometry data;     // Non-Index mode
9.          Transform DWORDs into floating point using FGHI_ATTRIB[n].Dt and send them to vertex shader.
10.     if (FGHI_ATTRIB[n].LastAttrib == 1) break;
11. end for
12. repeat step 2~11 (count) times. // the (count) value in step 1 is used.
```

在第八步和第九步中，DWDORD 的数目由 FGHI_ATTRIBn.Dt 和 FGHI_ATTRIBn.NumComp 决定。在第八步，主机接口不能正确的认识 DWORD。主机接口从 CPU 只能得到 DWORD 的需要的数据。这就是 CPU 必须正确传输索引和几何数据的原因。

在第九步，FGHI_ATTRIB[n].Dt 决定如何用浮点数格式传输转换的 DWORD。同时，在相同的步骤内，每个属性组成部分的顺序可以用每个 FGHI_ATTRIBn 内的 SrcX~SrcW 交换。SrcX~SrcW 的初始化值为 0，与转换到 SrcX~SrcW 内的第一个值映射。如果（x，y，z，w）属性按照顺序被转换，SrcX~SrcW 必须设置值为 2’b11，2’b10，2’b01，2’b00。这个构造在颜色值转换为 BGRA 而不是 RGBA 时非常有用。如果每个组成部分的数目不是 4 个，那么 0.0，0.0 和 1.0 将自动附加上。例如，如果(x，y)数据被转换，那么（FLOAT(x),FLOAT(y)，0.0，1.0）被发送到顶点着色器。

需要注意的是，从 CPU 或顶点缓冲区转换的顶点属性的 DWORD 必须为 DWORD aligned。如果转换（8 位 x，8 位 y，8 位 z），将忽略下一个 8 位数据。

42.3.5.如何从 CPU 向主机接口发送 DWORDS

在主机接口内有一个主机 FIFO 。CPU 只能向主机接口转换 DWORD 数据。如果 CPU 写入的数据比主机 FIFO 内的数据多，并且能够正确的储存，主机接口使 HREADY, 一个 AMBA 总线信号为低电平。这种情况下，AMBA 总线授予 3D 图形，在同一 AMBA 总线上的其他 IP 不能得到使用此总线的权利，这是不希望发生的情况。只读 FGHI_DWSPACE 寄存器用于解决此种情况。FGHI_DWSPACE 控制主机 FIFO 内的空闲的 DWORD 空间的数量。主机 FIFO 在主机接口内，并且可以储存 32 个 DWORD。无论 CPU 在什么时候发送计数，索引或几何数据，CPU 必须得到 FGHI_DWSPACE 值，转换 DWORD 的数量于 FGHI_DWSPACE 的值一样多。

一般而言，FGHI_DWSPACE 不告诉 CPU 准确的空闲空间，因为提供给 3D 图形和 AMBA 总线的时钟信号可以不同。(如果时钟信号相同，FGHI_DWSPACE 有准确的值)。因此，FGHI_DWSPACE 受 AMBA 总线和内部情况的影响。

如果读取的 FGI_DWSPACE 值比主机 FIFO 内的实际自由空间小，写操作将会完成，并且不会出现任何问题。另一方面，如果读取的 FGHI_DWSPACE 值比写入的 DWORD 的数量多，主机接口使 HREADY 信号变为低电平，并扩展转换。然而，读取的 FGHI_DWSPACE 值于实际值之间的不同通常很小。

CPU 读取 FGHI_DWSPACE 和转换 DWORD 与读取值一样多以后，CPU 可以做其他工作和处理，或者继续发送几何数据的其他部分，重复相同的处理。CPU 可以使用 3D 图形的中断方案。

当转换几何数据的时候，中断和顶点缓冲是非常有用的方案。

42.3.6. CPU 的索引转换类型

FGHI_CONTROL 内的 IdxType 控制着 CPU 的 DWORD 内存在的索引的数量。如果 IdxType 为无符号的整数类型，在转换的 DWORD 内只有 32 位的索引。如果为无符号的短型，在一个 DWORD 内有两个 16 位的索引。如果为无符号的字节类型，将有 4 个八位的索引。

当所有索引被使用时，DWORD 内剩下的索引被忽略。例如，如果转换了三个无符号字节类型索引的顶点，将用到一个 DWORD 数据。这种情况下，忽略 DWORD 内剩下的无符号字节。

42.3.7.顶点缓冲区的数据转换

在使用顶点缓冲区的内容之前，几何数据必须在顶点缓冲区内。首先，将顶点缓冲区内的 16 字节对齐的目标地址设置为 FGHI_VBADDR。然后，写入 FGHI_VBDATA 内的一串 DWORD 被储存到顶点缓冲区内。当 4 个 DWORD 写入 FGHI_VBDATA 内时，FGHI_VBADDR 的地址自动增加 16.因此，每向 FGHI_VBDATA 写入 DWORD 时，不需要更新目标地址。需要注意的是，写入 FGHI_VBDATA 的 DWORD 数量必须是 4 的倍数。只有当从 CPU 内转换 4 个 DWORD 时，这 4 个 DWORD 储存在顶点缓冲区内。

如果几何数据的尺寸不是 4 的倍数，那么向顶点缓冲区发送增加的 DWORD.DWORD-aligned 增加的 DWORD 不影响 FGHI_ATTRIBn_VBCTRL.Range 的值。实际的索引范围值必须写入 FGHI_ATTRIBn_VBCTRL.Range。

注意：NAN 或无限浮点数或半浮点值必须写入顶点缓冲区，像非索引模式一样。

同时注意顶点缓冲区内的几何数据必须是 DWORD-aligned。

42.3.8. 如何使用顶点缓冲区作为一个时间缓冲使用中断

主机 FIFO 内有 32 个 DWORD 空间。如果 CPU 发送大量的 FGHI_DWORDS 轮流检测的 DWORD，CPU 将耗费大量的周期读取 FGHI_DWSPACE,并没有做任何其他有用的工作，直到所有的 DWORD 转换完毕。这是不希望发生的情况。

顶点缓冲区和中断方案可以用在此种情况下。这种情况下的顶点缓冲区的一次几何应用：顶点缓冲区经常储存几何数据，假设进程中多次使用数据。

CPU 将几何数据的 DWORD 的一部分发送到顶点缓冲区内，而不是发送到主机 FIFO 内。保存 DWORD 到顶点缓冲区内后，当主机 FIFO 的 FGGB_POPESTATE 值和主机接口值变为 0 的时候，CPU 设置 3D 图形中断使中断单元向 CPU 发送中断。此时，CPU 可以做其他有用的工作，如运行系统或探测相关处理，等待 3D 图形的中断。如果 3D 图形发生中断，将允许 CPU 处理几何发送进程，CPU 用相同的步骤继续发送几何数据的剩下的部分。

当所有 3D 图形管道状态变为空时可以发生中断。可以自己决定什么时候发生中断。这里有一个需要注意的事情：必须发送准确的顶点的数量。例如，当图元引擎，就是顶点缓存的下一个模块，被假设得到