NAME

agp - generic interface to the Accelerated Graphics Port (AGP)

SYNOPSIS

device agp

DESCRIPTION

The **agp** driver provides uniform, abstract methods for controlling the following devices:

Ali:	M1541, M1621 and M1671 host to AGP bridges
AMD:	751, 761 and 762 host to AGP bridges
ATI:	RS100, RS200, RS250 and RS300 AGP bridges
Intel:	i820, i840, i845, i850, and i860 host to AGP bridges
Intel:	i810, i810-DC100, i810E, i815, 830M, 845G, 845M, 852GM, 852GME, 855GM, 855GME, 865G, 915G and 915GM SVGA controllers
Intel:	82443BX, 82443GX, 82443LX, 82815, 82820, 82830, 82840, 82845, 82845G, 82850, 82855, 82855GM, 82860, 82865, 82875P, E7205 and E7505 host to AGP bridges
NVIDIA:	nForce and nForce2 AGP controllers
SiS:	530, 540, 550, 620, 630, 645, 645DX, 648, 650, 651, 655, 661, 730, 735, 740, 741, 745, 746, 760 and 5591 host to AGP bridges
VIA:	3296, 82C597, 82C598, 82C691, 82C694X, 82C8363, 8235, 8237, 8361, 8367, 8371, 8377, 8501, 8601, 862x, 8633, 8653, 8703, 8753, 8754, 8763, 8783, KT880, PM800, PM880, PN800, PN880, PT880, XM266 and XN266 host to PCI bridges

The most common application of **agp** is for running X(7) (*ports/x11/xorg-docs*) on the Intel i81x controllers.

IOCTLS

The following ioctl(2) operations can be performed on */dev/agpgart*, which are defined in <*sys/agpio.h*>:

AGPIOC_INFO

Returns state of the **agp** system. The result is a pointer to the following structure:

```
typedef struct _agp_info {
        agp_version version;  /* version of the driver       */
        uint32_t bridge_id;   /* bridge vendor/device        */
        uint32_t agp_mode;    /* mode info of bridge         */
        off_t aper_base;      /* base of aperture            */
        size_t aper_size;     /* size of aperture            */
        size_t pg_total;      /* max pages (swap + system)   */
```

```
        size_t pg_system;     /* max pages (system)          */
        size_t pg_used;       /* current pages used          */
} agp_info;
```

AGPIOC_ACQUIRE

Acquire control of the AGP chipset for use by this client. Returns EBUSY if the AGP chipset is already acquired by another client.

AGPIOC_RELEASE

Release control of the AGP chipset. This does not unbind or free any allocated memory, which is the responsibility of the client to handle if necessary.

AGPIOC_SETUP

Enable the AGP hardware with the relevant mode. This ioctl(2) takes the following structure:

```
typedef struct _agp_setup {
        uint32_t agp_mode;    /* mode info of bridge */
} agp_setup;
```

The mode bits are defined in <*sys/agpio.h*>.

AGPIOC_ALLOCATE

Allocate physical memory suitable for mapping into the AGP aperture. This ioctl(2) takes the following structure:

```
typedef struct _agp_allocate {
        int key;              /* tag of allocation           */
        size_t pg_count;      /* number of pages             */
        uint32_t type;        /* 0 == normal, other devspec  */
        uint32_t physical;    /* device specific (some devices
                                 * need a phys address of the
                                 * actual page behind the gatt
                                 * table)                     */
} agp_allocate;
```

Returns a handle to the allocated memory.

AGPIOC_DEALLOCATE

Free the previously allocated memory associated with the handle passed.

AGPIOC_BIND

Bind the allocated memory at given offset with the AGP aperture. Returns EINVAL if the memory is already bound or the offset is not at AGP page boundary. This ioctl(2) takes the following structure:

```
typedef struct _agp_bind {
        int key;        /* tag of allocation          */
        off_t pg_start; /* starting page to populate  */
} agp_bind;
```

The tag of allocation is the handle returned by AGPIOC_ALLOCATE.

AGPIOC_UNBIND

Unbind memory from the AGP aperture. Returns EINVAL if the memory is not bound. This ioctl(2) takes the following structure:

```
typedef struct _agp_unbind {
        int key;            /* tag of allocation       */
        uint32_t priority;  /* priority for paging out */
} agp_unbind;
```

FILES

/dev/agpgart AGP device node.

SEE ALSO

X(7) (*ports/x11/xorg*)

HISTORY

The **agp** driver first appeared in FreeBSD 4.1.