ATA

The ATA module is a driver for a PC-standard dual ATA interface, bridged over PCI. It supports up to four devices, two on each 'bus'. The devices may be disk or CDROM drives. The driver attempts to select the 'best' operating mode for each device, between PIO and DMA.

Caveat

Although most of this code is quite generic for ATA devices, there are two chip-dependent portions in the code that would need changing for another chipset configuration. These are:

1. the DMA and IDE timings are set in the PCI-IDE bridge controller chip during initialisation. This code is specific to the I400BX chipset used on the SPB450 motherboard.
2. the DMA code relies on the the DMA driver support. In this system, this is provided by the PCI bridge controller. These interface calls (*pci_dma_start* etc.) could be macro-defined for another API providing the same functionality.

Apart from the support for ATAPI functions, the code should work with almost any modern ATA controller chipset.

The support for ATAPI devices is minimal (only the 'packet' interface) and would need more work to be usable for real devices.

Process Information

Prototype Name	ata
Process Priority	driver-level
Process Name	does not matter as long as it matches the URLs used within the system to open the device.

Target File Definitions

ATA_CLEAR_INT	This macro defines the means by which the interrupt handler can clear the pending ATA interrupt(s).
ATA_MASTER	The index to the select the master drive on a cable (usually 0).
ATA_PRIMARY	The index of the primary ATA channel (cable) in the system (usually 0).

ATA_PRIMARY_ALTSTATUS	The address of the alternate-status register for the primary channel.
ATA_PRIMARY_IOSPACE	The address of the main register set for the primary channel.
ATA_PRIMARY_IRQ	The interrupt number generated by the primary channel.
ATA_SECONDARY	The index of the secondary ATA channel (cable) in the system (usually 1).
ATA_SECONDARY_ALTSTATUS	The address of the alternate-status register for the secondary channel.
ATA_SECONDARY_IOSPACE	The address of the main register set for the secondary channel.
ATA_SECONDARY_IRQ	The interrupt number generated by the seconday channel.
ATA_SLAVE	The index to select the slave drive on a cable (usually 1).

Process Operation

The initialisation routine searches for up to four ATA devices (two on each channel), and probes each device for its capacilities (ATA or ATAPI, and PIO or DMA, and maximum transfer speed). It then sets the modes and timings for the bridge chip and tells each device which mode will be used. In the case of DMA operations, the data are copied between the (uncached) DMA memory and the (cached) mblkd buffer within the driver layer.

Dataflow handling

Each channel (of the two possible ATA channels) maintains a queue of pending operations. Unlike network drivers, all operations to the driver are synchronous (both send and receive requests) and are processed in the order in which they are queued. An operation is started when it is queued to an idle channel. Subsequent requests are queued behind the active request. In the interrupt handler, the active request is completed and the reply returned to the caller. If there is another request in the queue, it is started. Only two operations will run in parallel, one for each channel. The four possible types of operation are read (PIO or DMA), write (PIO or DMA), packet-read (PIO only) or packet-write (PIO only). If the device is DMA-capable, a DMA channel is used for the transfer, otherwise the request is handled 'inline' in the interrupt handler.

For read and write requests, the *b_immed* field of the mblk is used to contain the logical block number of the first block in the request. Internally, all devices use absolute LBA addressing; the block address is converted to CHS if needed before programming the device registers.

For ATAPI packet operations, the six-byte packet is sent followed either by sending data (for a write) or reading data (for a read).

Messages

The module has a queue handler and main process. and accepts the *ATAPI* messageset and the messages defined in the *Standard* messageset for dataflows into and out of the driver with the following processing:

CLOSE messages are handled by the main process. It is assumed that the close request is not sent while there are outstanding I/O requests to the driver. Any outstanding event requests are returned and the context information block is freed before the reply is returned to the sender.

EVENT messages are added to the file's event queue. Currently, the driver does not generate any events (for example for disk removal), so the messages are just held until the file is closed.

FETMBLK messages are added to the queue of operations for the appropriate device.

FSLIB_DEV_ATTR messages are handled in the main process. The *port* field in the supplied URL selects the device for which information is to be returned. The *maxblk* and *blksize* parameters for the device are returned in the reply.

FLUSH messages are added to the queue of operations for the appropriate device. When the FLUSH message reaches the head of the queue, it is replied to from the interrupt handler, as all previously-queued operations must have completed by that time.

GETMBLK messages are added to the queue of operations for the appropriate device.

NEWMBLK messages return a buffer, of at least 512 bytes, to the caller from the queue handler.

OPEN messages are handled in the main process. The *port* field of the URL contains the index number of the device (in the same order as in *ata_find_devices*, below). The process returns a value in the *dest_context* field which must be supplied on subsequent messages.

OUTMBLK messages are added to the queue of operations for the appropriate device.

PACKETR messages are added to the queue of operations for the appropriate device.

PACKETW messages are added to the queue of operations for the appropriate device.

PUTMBLK messages with zero length cause the message's buffer to be freed in the queue handler, otherwise the message is added to the queue of operations for the appropriate device.

RETMBLK messages free the allocated block from within the queue handler and are replied to immediately.

Shared Library Macros and Routines

ata_find_devices

void *ata_find_devices*(
int **istatus*)

The *ata_find_devices* routine returns four status indications in the integer array *istatus* for the primary-master, primary-slave, secondary-master and secondary-slave devices respectively. Each indication is one of *ATA_DEVICE_ABSENT*, if no device was detected, *ATA_DEVICE_IDE*, if an IDE (standard ATA) device was found, or *ATA_DEVICE_ATAPI* if a packet-mode device was found.