

Connecting the Intel® PXA27x Processor Family to a Hard-Disk Drive via the VLIO Memory Interface

Application Note

October 2005

Order Number 308422-002



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at http://www.intel.com.

AlertVIEW, AnyPoint, AppChoice, BoardWatch, BunnyPeople, CablePort, Celeron, Chips, CT Connect, CT Media, Dialogic, DM3, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Create & Share, Intel GigaBlade, Intel InBusiness, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel Play, Intel Play logo, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel TeamStation, Intel Xeon, Intel XScale, IPLink, Itanium, LANDesk, LanRover, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PC Dads, PC Parents, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, RemoteExpress, Shiva, SmartDie, Solutions960, Sound Mark, StorageExpress, The Computer Inside., The Journey Inside, TokenExpress, Trillium, VoiceBrick, Vtune, and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © Intel Corporation, 2005. All Rights Reserved.

Contents

1.0 Introduction 5

2.0 Hardware Overview 5

2.1 Intel® PXA27x Processor Memory Interface 5
2.2 Intel® PXA27x Processor VLIO Interface 6
2.3 Intel® PXA27x Processor DMA Controller Interface 7
2.4 The CompactFlash True IDE Standard 8

3.0 Mapping Intel® PXA27x Processor VLIO to the True IDE CompactFlash Interface 9

3.1 CompactFlash HDD Connections to the Intel® PXA27x Processor 10
3.2 True IDE Multi-word DMA Mode Block Diagram 11
3.3 Timing Considerations 11
3.4 Intel® PXA27x Processor Register Configuration 12
3.4.1 MSC Register Setting 12
3.4.2 Intel® PXA27x Processor GPIO Register settings 12

4.0 HDD Device Driver 13

4.1 HDD Driver Overview 13
4.2 Setting the HDD Transfer Mode 13
4.3 I/O Address Mapping 14
4.4 PIO Mode and Interrupt Handling 14
4.5 DMA Mode and Interrupt Handling 14
4.6 Advanced Feature Setting 15
4.6.1 Enabling Write Cache: 15
4.6.2 Enabling Power Management (PM) 15
4.6.3 Setting Multiple Sectors 16
4.7 Linux Driver Considerations 16
4.7.1 I/O Address Mapping 16
4.7.2 Request DMA Channel 16
4.7.3 Request IRQ 17
4.7.4 DMA Buffer and Descriptor Allocation 17
4.8 Microsoft* Windows* CE Driver Considerations 17
4.8.1 I/O Address Mapping 17
4.8.2 Request DMA Channel 17
4.8.3 Request IRQ 17
4.8.4 DMA Buffer And Descriptor Allocation 17

5.0 Summary 18

Figures

1 Intel® PXA27x Processorr Block Diagram 6
2 True IDE PIO Mode Block Diagram 9
3 Block Diagram for True IDE DMA Mode 11

Tables

1 VLIO Memory Interface Signals 7
2 DMA Controller Interface Signals 8
3 Listing of the Significant CompactFlash True IDE Signals 8
4 Memory Mapping 10
5 Acronym Definitions 18

Revision History

Date	Revision	Description
June 2005	001	Initial draft of Application Note, *Connecting the Intel® PXA27x Processor Family to a Hard-Disk Drive via the VLIO Memory Interface.*
October 2005	002	Corrected Signal name for “Write enable for VLIO memory” Modified Timing considerations for Section 3.3 Updated MSC Register Setting in Section 3.4.1 Corrected the comments for Example 2 of Section 3.4.2

§§

1.0 Introduction

As handheld devices increase in functionality, greater in-system storage capacity is needed. At the same time, hard-disk drives, subsequently referred to as HDDs, are increasing in capacity while decreasing in price and size. Small HDDs such as the 1.8" and 1.0" versions, have become competitive to flash storage for some designs. Potential uses of these small HDDs include MP3 players, portable media player, portable gaming systems, PDAs, and phones.

The Intel® PXA27x Processor Family (PXA27x processor) offers an integrated system-on-a-chip design based on the Intel XScale® microarchitecture. The PXA27x processor integrates a 624 MHz Intel XScale®core with many on-chip peripherals enabling a wide variety of products for the handheld and cellular handset markets. The PXA27x processor memory controller provides several possible solutions to implement a HDD interface. These interfaces include the PCMCIA/ CompactFlash* interface and the variable latency input/output (VLIO) interface. This application note provides detailed suggestions for implementing a CompactFlash, true IDE mode, parallel ATA, HDD to a PXA27x processor using the VLIO memory interface. Configurations using programmed IO (PIO) and flow-through DMA are presented. The VLIO interface solution only requires a small number of additional components and produces low-cost and efficient DMA performance.

2.0 Hardware Overview

This section describes the hardware requirements for implementing a connection of the PXA27x processor to a HDD using the VLIO interface.

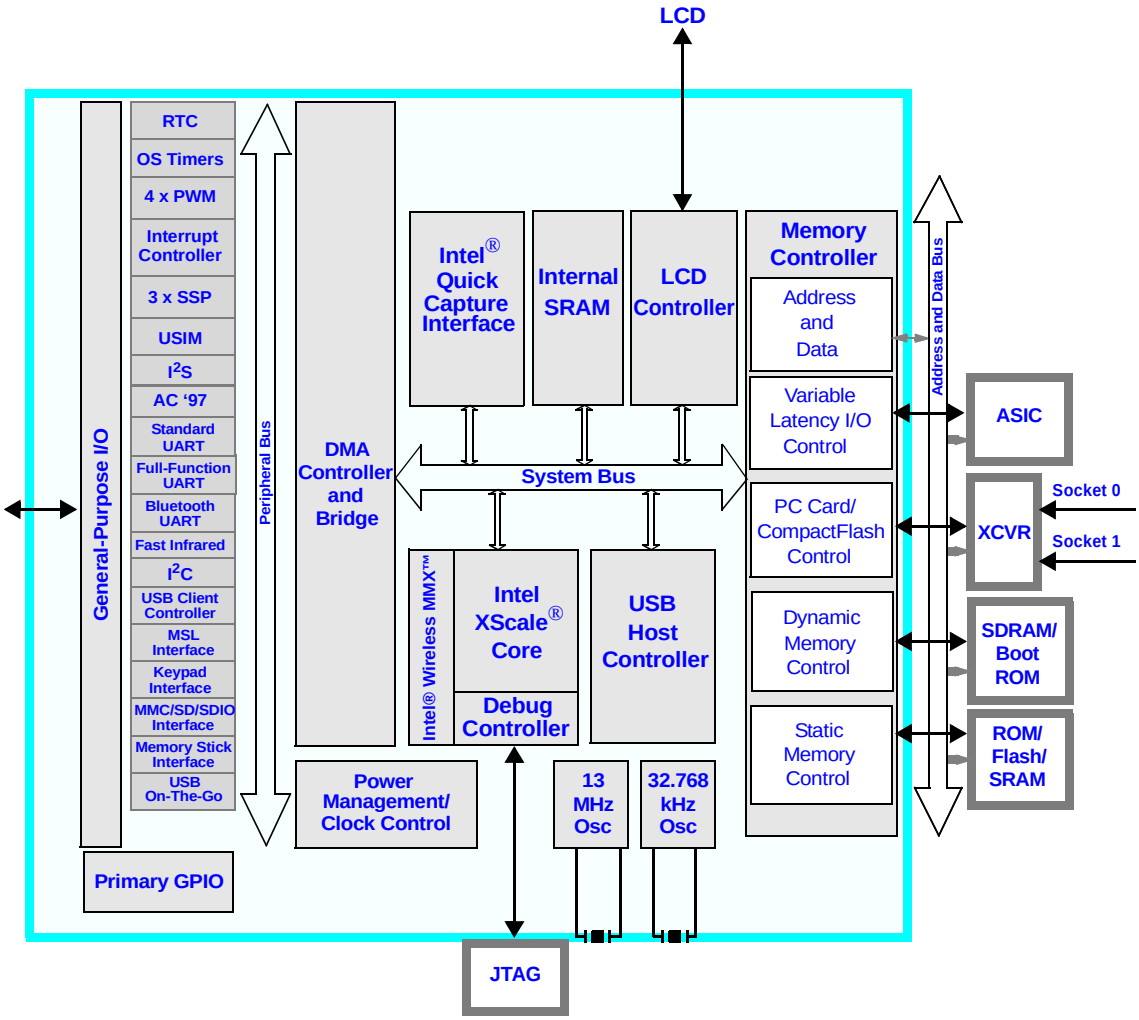
2.1 Intel® PXA27x Processor Memory Interface

The PXA27x processor memory controller supports many memory types and operations via following interfaces:

- SDRAM interface
- Flash memory interface (asynchronous/synchronous)
- ROM interface
- Variable latency input/output (VLIO) interface
- PC card (PCMCIA) interface
- Alternate bus master interface

Figure 1 illustrates the PXA27x processor and memory interface.

Figure 1. Intel® PXA27x Processor Block Diagram



2.2 Intel® PXA27x Processor VLIO Interface

Each of the six static memory locations (nCS<5:0>) can be configured to operate in VLIO mode by programming the corresponding MSCx[RTx] to 0b100. When a static chip select is used as a VLIO interface, its functionality is similar to that of an SRAM with some important differences.

- VLIO allows for the additional ability to insert a variable number of wait states through the RDY pin.
- VLIO read accesses also differ from SRAM read accesses in that nOE toggles for each beat of a burst.
- The first nOE assertion occurs two CLK_MEM cycles after the chip select signal (nCSx) is asserted.

- For VLIO writes, nPWE is used instead of nWE.

Table 1 lists the signals available to interface to VLIO memory devices.

Table 1. VLIO Memory Interface Signals

Signal Name	Direction	Polarity	Description
nCS<5:0>	Output	Active Low	Chip selects for static memory
MA<25:0>	Output	N/A	Output address to all memory types **NOTE:** Do not use MA0 for byte addressing because all VLIO devices must have a minimum bus width of 16 bits when interfacing to the PXA27x processor memory controller. An optional addressing mode is available to use MA0 to address the upper 64 Mbytes of memory within a 128 Mbytes partition.
MD<31:0>	Bidirectional	N/A	Bidirectional data for all memory types
DQM<3:0>	Output	Active High	Data byte mask control DQM<0> corresponds to MD<7:0> DQM<1> corresponds to MD<15:8> DQM<2> corresponds to MD<23:16> DQM<3> corresponds to MD<31:24> 0 = Do not mask out corresponding byte 1 = Mask out corresponding byte
nPWE	Output	Active Low	Write enable for VLIO memory
nOE	Output	Active Low	Output enable for Static Memory
RDY	Input	Active High	Variable Latency I/O signal for inserting wait states 0 = Wait 1 = VLIO is ready
Miscellaneous I/O Signals			
RDnWR	Output	Active High	Data direction signal to be used by output transceivers 0 = MD<31:0> is driven by the PXA27x processor 1 = MD<31:0> is not driven by the PXA27x processor

2.3 Intel® PXA27x Processor DMA Controller Interface

The PXA27x processor DMA controller transfers data to and from the memory system in response to requests generated by peripherals or companion chips. These peripheral devices and companion chips do not directly supply addresses and commands to the memory system. Instead, the addresses and commands are configured by software and maintained by each of the 32 DMA channels within the PXA27x processor DMA controller. The PXA27x processor DMA controller supports both flow-through and fly-by transfers. Refer to Section 5 of the Intel® *PXA27x Processor Family Design Guide* and to the *Intel® PXA27x Processor Family Developer's Manual* for the usage and constraints of flow-through and fly-by transfers and further detailed information about the PXA27x processor DMA controller. The HDD interface presented in this application note uses flow-through DMA to achieve high-throughput performance with reduced SW overhead due to the PXA27x processor DMA controller performing data transfer operations.

Table 2 lists the signals available to interface to DMA controller.

Table 2. DMA Controller Interface Signals

Signal Name	Type	Description
DREQ<2:0>	Input	External Companion Chip Request The external companion chip asserts the DREQ signal when a DMA transfer request is required. The DMA controller registers the transition from low to high to identify a new request. The signal must remain asserted for four MEM_CLK cycles to allow the DMA controller to recognize the low-to-high transition. When the signal is de-asserted, the signal must remain de-asserted for at least four MEM_CLK cycles. The external companion chip need not wait until the completion of the data transfer before asserting the next request. This companion chip has up to 31 outstanding requests on each of the DREQ<2:0> pins. The number of pending requests are logged in special status registers, DRQSRx. Requests on pins DREQ<1:0> are used for data transfers in either fly-by or flow-through modes. Requests on pins DREQ<2> are used for data transfers in flow-through mode only.
DVAL<1:0>	Output	External Companion Chip Valid The memory controller asserts DVAL to notify the companion chip. Either data must be driven or the output data is valid.

2.4 The CompactFlash True IDE Standard

The *CF+ and CompactFlash Specification, Revision 3.0*[1] defines three kinds of operating modes that could be used to connect to a HDD.

- PC Card ATA using I/O mode
- PC Card ATA using memory mode
- True IDE mode

Most parallel ATA HDD used in embedded applications work under true IDE mode, which has advantages of low-cost, ATA HDD compatibility, and good performance. Table 3 lists the CompactFlash signals used in the true IDE implementation.

Table 3. Listing of the Significant CompactFlash True IDE Signals (Sheet 1 of 2)

Signal Name	Direction	Description
D<0:15>	Bidirectional	Data lines
A<0:2>	Input	HDD address line <0:2>
IORD#	Input	Drive I/O read strobe
IOWR#	Input	Drive I/0 write strobe
ATA SEL#	Input	Enables true IDE mode
IOCS16#	Output	Indicates a 16 bit transfer
DMARQ#	Output	DMA Request

1. For more information about CompactFlash and the true IDE specification refer to the *CF+ and CompactFlash Specificatio*n Revision 3.0, which can be downloaded at the CompactFlash Association Web site at http://www.compactflash.org/.

Table 3. Listing of the Significant CompactFlash True IDE Signals (Sheet 2 of 2)

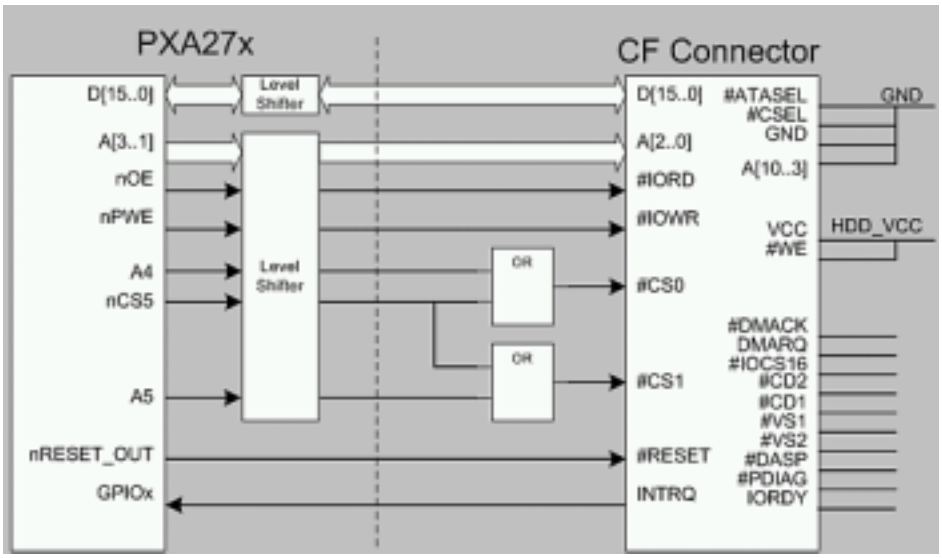
Signal Name	Direction	Description
DMACK	Input	DMA Acknowledge
INTRQ	Output	Interrupt Request
Reset#	Input	Reset
CSEL#	Input	Cable Select
CS#<0:1>	Input	Chips Select Signals
IORDY	Output	Input/Output Ready

3.0 Mapping Intel® PXA27x Processor VLIO to the True IDE CompactFlash Interface

This section describes the physical and logic interface of the CompactFlash card in true IDE mode to the PXA27x processor VLIO interface. The device driver example code presented in this application note assumes the CompactFlash HDD device is connected in this mode. A CompactFlash card is configured in a true IDE mode of operation only when the ATASEL# (OE#) input signal is grounded by the host during the power-up sequence.

In CompactFlash true IDE mode, the design discussed is designed to work under PIO mode 4 and DMA mode 2, but not Ultra DMA or any other working mode. Figure 2 shows a block diagram of a CompactFlash connection to the PXA27x processor VLIO interface in true IDE PIO mode.

Figure 2. True IDE PIO Mode Block Diagram



3.1 CompactFlash HDD Connections to the Intel® PXA27x Processor

The HDD in this application note is assumed to be using 3.3 V logic. The PXA27x processor is assumed to be running a 1.8 V memory bus, therefore level shifters are used to buffer logic to and from the HDD interface logic.

The following list describes the mapping of the CompactFlash true IDE signals to the PXA27x processor.

- **HDD D15-D0**—The data lines of HDD are connected to PXA27x processor D0 to D15 lines.
- **HDD A2-A0**—A[2:0] are connected to A[3:1] of the PXA27x processor, because all VLIO devices must have a minimum bus width of 16 bits when interfacing to the PXA27x processor memory controller. There is no need for byte select by A0 of the PXA27x processor.
- **HDD A10-A3** — These address lines on the HDD are grounded.
- **HDD IORD#** — The IO read strobe line from the CompactFlash card is connected to nOE of PXA27x processor memory controller.
- **HDD IOWR#**—The IO write strobe line from the CompactFlash card is connected to nPWE of PXA27x processor memory controller.
- **HDD RESET#** — The reset line from the CompactFlash card is directly connected to PXA27x processor nReset_Out or the power-on reset circuitry to reset the CompactFlash device.
- **HDD ATASEL#** —The signal is grounded to indicate true IDE mode.
- **HDD CSEL#** —The signal is grounded to indicate there is only one drive, the master drive, that can be connected.
- **HDD WE#** —The signal is tied to VCC.
- **HDD DMARQ, DMACK#** —These signals are specific for DMA transfers between the PXA27x processor and the HDD. They are not used under true IDE PIO mode. For PIO mode, the HDD DMACK# should be left floating or pulled up to VCC. Please refer to the CF+ and CompactFlash Specification for more details about all these signals and what to do with unused signals. For true IDE multi-word DMA mode, please refer to Section 3.2 of this application note for a description of the logic and connection of these signals.
- **HDD IOCS16#, CD2#, CD1#, IORDY, DASP#, PDIAG#, VS1#, VS2#** —These signals are not used.
- **HDD CS0#, CS1#** —The design presented here uses two address lines and nCS5 of PXA27x to get the two chip select signals HDD needs. Refer to Table 4 for the relationship of the PXA27x processor nCS5 memory mapping to the HDD CS0# and CS1# signals.

Table 4. Memory Mapping

nCS5	A4	A5	CS0#	CS1#	R/W Field	Address
0	0	1	0	1	Task file register	0x1400_0020
0	1	0	1	0	CTL Register	0x1400_0010
0	1	1	1	1	DMA	0x1400_0030
1	X	X	1	1	N/A	N/A

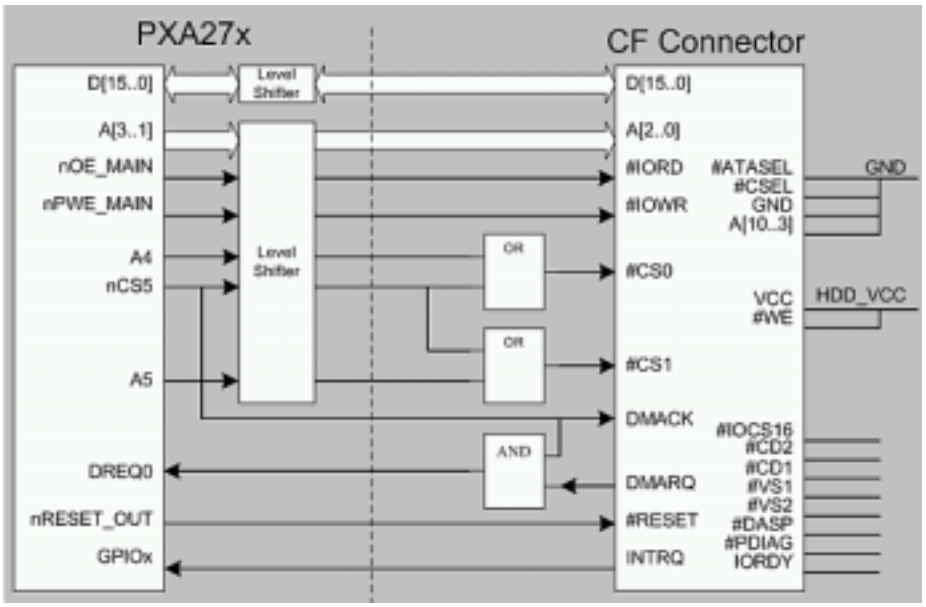
3.2 True IDE Multi-word DMA Mode Block Diagram

The flow-through mode of the PXA27x DMA controller can be used to transfer data between the PXA27x processor and the HDD. There are two challenges to overcome when implementing DMA support:

1. PXA27x processor DMA signals are edge-triggered, but HDD DMA signals are level-triggered; To address this issue, an AND gate has been added to the block diagram to combine the nCS5 signal and the DMARQ lines signals going to the PXA27x processor DREQ0 signal.
2. When the PXA27x processor DMA controller operates in flow-through mode, no DMA acknowledge signal is generated. To address this issue, the nCS5 pin is used as the DMACK signal the HDD needs.

Figure 3 contains a block diagram of a CompactFlash connection to the PXA27x processor VLIO interface while in true IDE DMA mode.

Figure 3. Block Diagram for True IDE DMA Mode



3.3 Timing Considerations

The CompactFlash specification defines timing requirement for PIO mode 4 and DMA mode 2 of CompactFlash true IDE. Please refer to Chapter 4 of CF+ and CompactFlash Specification for detail. Detailed timing comparisons should be made using both the *Intel® PXA27x Processor Family Developer's Manual* and the *Intel® PXA27x Processor Electrical, Mechanical, and Thermal Specification* and the applicable HDD documentation.

The PXA27x VLIO interface is designed to interface to some high-speed memory devices, such as Flash and SRAM. This capability requires attention to timing parameters. Analysis of HDD vendor devices shows that the "CS Valid to -IORD/-IOWR" is close to the specified limits of the PXA27x processor. Perform timing analysis and component selection. In some cases, it may be necessary to take one of the following actions:

- Use glue logic to adjust the timing.
- Select an alternate HDD with different timing requirements.
- Contact the HDD manufacturer to confirm whether a small timing difference is acceptable.

3.4 Intel® PXA27x Processor Register Configuration

3.4.1 MSC Register Setting

To connect PXA27x nCS5 to an IDE interface, the MSC2 register must be set up to meet the true IDE interface timing requirements. MSC2 register covers nCS5 and nCS4, to configure nCS5 only the higher half word of this register need modification.

- Bit 31 [RBUFF]: set this bit to indicate a slow memory device.
- Bit 30-28 [RRR]: defines the interval from the time nCS is de-asserted (after a read/write) until nCS is asserted. Set this field to 0b0001 when the memory clock is both104 MHz and 208 MHz.
- Bit 27-24 [RDN]: the MSCx[RDN] field controls subsequent read access times to burst flash memory for address hold on nPWE/nOE. Set this field to 0b0010 when the memory clock is 104 MHz and to 0b0011 when the memory clock is 208 MHz.
- Bit 23-20 [RDF]: for MD/DQM setup to nPWE de-asserted and nPWE/nOE assert period between writes. Set this field to 0b0111 when memory clock is 104 MHz memory and 0b1100 when memory clock is 208 MHz.
- Bit 19: set this bit to 1 to indicate a 16-bit data bus width.
- Bit 16-18: set this field to 0b100 to indicate a VLIO device.

Example 1. Sample Code to Set MSC2 Register

To set the MSC2 register:

```
    unsigned long msc2=MSC2;
    msc2&=0x0000ffff;

#ifndef VLIO_MEM208M
    msc2|=0 << 31 |
             1 << 28 |
             2 << 24 |
             7 << 20 |
             1 << 19 |
             4 << 16;
#else
    msc2|=0 << 31 |
             1 << 28 |
             3 << 24 |
             12<< 20 |
             1 << 19 |
             4 << 16;
#endif
    MSC2= msc2;
```

3.4.2 Intel® PXA27x Processor GPIO Register settings

To enable VLIO and DMA signal requirements, the following GPIOs must be configured:

- DREQ0 GPIO20: configure as the DMA request signal. Set this pin to be an input and set it to alternative function 1
- RDY GPIO18: set to input and alternative function 1
- nCS5 GPIO33: set it to output and alternative function 2
- nPWE GPIO49: set to output and alternative function 2

Example 2. Sample Code to Set GPIO Registers

To set GPIO registers:

```
// pin 18 20
pins[0]=2;fn[0]=2;
pins[1]=XLLP_GPIO_RDY;
pins[2]=XLLP_GPIO_MBREQ;
fn[1]=XLLP_GPIO_ALT_FN_1;
fn[2]=XLLP_GPIO_ALT_FN_1;
XllpGpioSetDirectionIn((P_XLLP_GPIO_T) v_pGPIOReg, pins);
XllpGpioSetAlternateFn((P_XLLP_GPIO_T)v_pGPIOReg, pins, fn);
// pin 33 49
pins[0]=2;fn[0]=2;
pins[1]=XLLP_GPIO_nPWE;
pins[2]=XLLP_GPIO_nCS5;
fn[1]=XLLP_GPIO_ALT_FN_2;
fn[2]=XLLP_GPIO_ALT_FN_2;

XllpGpioSetOutputState1((P_XLLP_GPIO_T)v_pGPIOReg, pins);
XllpGpioSetDirectionOut((P_XLLP_GPIO_T)v_pGPIOReg, pins);
XllpGpioSetAlternateFn((P_XLLP_GPIO_T)v_pGPIOReg, pins, fn);
```

4.0 HDD Device Driver

4.1 HDD Driver Overview

The true IDE HDD driver is registered to the OS as a block driver for both Linux* and Microsoft* Windows* CE. In both cases, the driver needs to implement read/write and IOCTL operations. Both PIO mode and DMA mode are supported in the driver with the same read/write interface to upper layer applications. For PIO mode, all data read from or written to the HDD interface is handled by the CPU core directly. CPU use will be very high when accessing HDD data. For DMA mode, the CPU core participates only in preparing DMA buffers and descriptors. The DMA controller is responsible for transferring data between SDRAM and the HDD, which frees the CPU core to handle other system tasks.

4.2 Setting the HDD Transfer Mode

At the beginning of initializing a driver, set the HDD transfer mode, for example, to PIO mode4 or DMA mode2. Example 3 illustrates the use of the SET_FEATURE(0xEF) command to set the HDD transfer mode.

Example 3. Sample Code to Set HDD Transfer Mode

To set the HDD transfer mode:

```
#define ATA_CMD_SETFEATURES 0xEF
```

```
#define SETFEATURES_XFER 0x03
#define XFER_MW_DMA_2 0x0c
#define XFER_PIO_4 0x22
//set transfer modes
   VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_DRV_HEAD,0xa0);
#ifdef   VLIODISK_PIOMODE4
   VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_SECT_CNT, XFER_PIO_4);
#else
   VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_SECT_CNT, XFER_MW_DMA_2);
#endif
   VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_FEATURE, SETFEATURES_XFER);
   VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_COMMAND, ATA_CMD_SETFEATURES);
```

4.3 I/O Address Mapping

For nCS5, the physical address is a 64 Mbytes slot starting from 0x1400_0000. To access this physical I/O space in a full-function operation system, this space must be mapped to a virtual address space.

4.4 PIO Mode and Interrupt Handling

The true IDE HDD interrupt line is high-level triggered. For PIO mode the interrupt happens in following manner:

Read Operation:

1. Host first sets parameter to HDD task file registers, then sends read command.
2. Host waits for HDD interrupt to happen.
3. When the HDD prepares some data in its internal buffer, the HDD pulls up the interrupt line to trigger the interrupt.
4. Host is notified by HDD, trapping into interrupt mode, then reads data from HDD through memory bus.
5. HDD clears interrupt and status when all data is read.

Write Operation:

1. Host first sets parameter to HDD task file registers, then sends write command.
2. HDD sets DRQ status when it is ready to receive data.
3. Host writes data to the HDD.
4. HDD triggers interrupt when all required data are received.
5. Host is notified by HDD, trapping into interrupt mode, then clears status.

4.5 DMA Mode and Interrupt Handling

For DMA mode, the interrupt happens in following manner:

Read Operation:

1. Host first sets parameter to HDD task file registers, prepares DMA and its descriptors and then sends a read command.

2. HDD requests DMA by asserting nDMARQ.
3. Host DMA controller responds with nDMACK and reads data from memory bus.
4. When all data is read out, HDD triggers interrupt.
5. Host is notified by HDD, trapping into interrupt mode, then clears interrupt and status.

Write Operation

1. Host sets parameter to HDD file register, prepares DMA and its descriptors, and then sends write command.
2. HDD requests DMA by asserting nDMARQ.
3. Host DMA controller responds with nDMACK and writes data to the HDD.
4. When all data is received, HDD triggers interrupt.
5. Host is notified by HDD, trapping into interrupt mode, then clears interrupt and status.

4.6 Advanced Feature Setting

4.6.1 Enabling Write Cache:

Almost all HDDs have an internal write buffer, but by default the buffer is always disabled. To enable write cache, use SET_FEATURE(0xEF) command with feature parameter 2

Example 4. Sample Code Enabling Write Cache

To enable write cache:

```
#define ATA_CMD_SETFEATURES 0xEF
#define SETFEATURES_EN_WCACHE 0x02

VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_DRV_HEAD,0xa0);
VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_FEATURE, SETFEATURES_EN_WCACHE);
VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_COMMAND, ATA_CMD_SETFEATURES);
```

4.6.2 Enabling Power Management (PM)

Power management feature enables HDD to enter standby status automatically when idle for a few seconds.

To enable PM feature, use ATA_CMD_SET_IDLE command with Sector Count register to indicate idle seconds (*5) to enter standby.

Example 5. Sample Code Enabling Power Management

```
#define ATA_CMD_SET_IDLE 0xE3
VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_DRV_HEAD,0xa0);
VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_SECT_CNT, 2); // set 2*5 seconds to idle
mode
VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_COMMAND, ATA_CMD_SET_IDLE);
```

4.6.3 Setting Multiple Sectors

When using PIO mode, by default, each sector read/write operation triggers one interrupt, which is not necessary. The ATA specification defines multi-sectors to make this more efficient. This command can set 2 or more sectors together to trigger just one interrupt.

Example 6. Sample Code Setting Multiple Sectors

To set multiple sectors:

```
VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_SECT_CNT, pDisk->d_MaxMultiSectors);
VLIO_WRITE_PORT_UCHAR(TRUE_IDE_ATA_REG_COMMAND, ATA_CMD_SETMULTI);
```

The parameter MaxMultiSectors can be found in the HDD's identity data.

4.7 Linux Driver Considerations

When creating a Linux driver, the following issues must be addressed:

- Mapping I/O address
- Requesting DMA channel
- Requesting interrupt
- Completing DMA buffer and descriptor allocation

4.7.1 I/O Address Mapping

Linux can use mmap api to map physical address to a virtual space. Another choice is to add the VLIO address space to the system static mapping table that is in the file, arch\arm\mach-pxa\generic-bvd.c.

Example 7. Example Static Mapping Table

An example of a static mapping table with VLIO address space added:

```
static struct map_desc standard_io_desc[] __initdata = {
 /* virtual     physical    length      domain     r  w  c  b */
  { 0xf5000000, 0x14000000, 0x01000000, DOMAIN_IO, 0, 1, 0, 0 }, /* VLIO IO */
  { 0xf6000000, 0x20000000, 0x01000000, DOMAIN_IO, 0, 1, 0, 0 }, /* PCMCIA0 IO */
  { 0xf7000000, 0x30000000, 0x01000000, DOMAIN_IO, 0, 1, 0, 0 }, /* PCMCIA1 IO */
  { 0xf8000000, 0x40000000, 0x01800000, DOMAIN_IO, 0, 1, 0, 0 }, /* Devs */
  { 0xfa000000, 0x44000000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 }, /* LCD */
  { 0xfc000000, 0x48000000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 }, /* Mem Ctl */
  { 0xfe000000, 0x4c000000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 }, /* USB host */
  { 0xff000000, 0x00000000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 }, /* UNCACHED_PHYS_0
*/
  LAST_DESC
};
```

Line 1 in the example static mapping table shows the MMU attributes for the newly-added VLIO device.

4.7.2 Request DMA Channel

Example 8. Sample DMA Channel Request

```
vlio_dma_channel=pxa_request_dma("VLIODMA",DMA_PRIO_HIGH, vlio_dma_finished,
NULL);
```

4.7.3 Request IRQ

Example 9. Sample IRQ Request

```
request_irq(hwif->irq,&ide_intr,sa,hwif->name,hwgroup)
```

4.7.4 DMA Buffer and Descriptor Allocation

Example 10. Sample DMA Buffer and Descriptor Allocation

```
consistent_alloc( GFP_KERNEL, VLIO_DMA_DESC_NUM * sizeof(pxa_dma_desc), (void
*)&vlio_dma_descriptors_physical, 0);
```

4.8 Microsoft* Windows* CE Driver Considerations

As with the Linux driver, the following issues must be addressed when creating a Microsoft* Windows* CE Driver:

- Mapping I/O address
- Requesting DMA channel
- Requesting interrupt
- Completing DMA buffer and descriptor allocation

4.8.1 I/O Address Mapping

Example 11. Mapping VLIO Address Space

Windows* CE can map the VLIO address space like this:

```
v_VlioBase = (char *)MmMapIoSpace(ioPhysicalBaseVlio,0x100,FALSE);
```

4.8.2 Request DMA Channel

Example 12. Sample DMA Channel Request

```
XllpDmacAllocChannel(&v_DMAChannel, XLLP_DMAC_CHANNEL_PRIORITY_HIGH)
```

4.8.3 Request IRQ

Example 13. Sample IRQ Request

```
InterruptInitialize(SYSINTR_EXPBD, pDisk->d_IRQEvent, NULL, 0)
```

4.8.4 DMA Buffer And Descriptor Allocation

Example 14. Sample DMA Buffer and Descriptor Allocation

```
v_VlioDMADescVirtual = (char *) HalAllocateCommonBuffer(&Adapter, TOTAL_BUF_SIZE,
&PA, FALSE);
```

5.0 Summary

This application note provides suggestions and details on how to connect a PXA27x processor to a CompactFlash true IDE mode HDD. The application note includes the HDD connection in both PIO and DMA mode, as well as HDD driver development using both Linux and Windows* CE operating systems.

Appendix A Acronyms

Table 5. Acronym Definitions

Acronym	Description
ATA	Advanced Technology Attachment
CF	CompactFlash
CS	Chip Select
DMA	Direct Memory Access
DRQ	Data Request
GPIO	General Purpose I/O
HDD	Hard-Disk Drive
IDE	Integrated Device Electronics
PDA	Personal Digital Assistant
PIO	Programmed I/O
PM	Power Management
VLIO	Variable Latency Input/Output

§§