Accessing North Bridge and South Bridge registers on the Vortex86

Summary

Application Note
AP0102 (v1)

How to write to the 32bit registers in the Vortex86SX and Vortex86DX processors using a 16bit DOS operating system

These DOS code examples were written and compile using Borland Turbo C++.
There is a download link for this compiler at http://cc.embarcadero.com/item/26014

The Borland Turbo C++ compiler and the Assembler (TASM) are 16bit whereas the North Bridge and South Bridge registers in the Vortex86 are 32bit so we need a method of accessing these registers using a Turbo C++ compiled program. This application note provides a method of achieving this.

We will create two functions outport32 and inport32 which will use in-line assembly code to perform the 32 bit read and writes. As the assembler is only 16bit there is no op-code for reading a 32bit processor register, for example EAX. To overcome this we can manually insert the 32bit operand 66h ahead of the AX command.

To do this first define the operand to make the program easier to read.

```
#define OPERAND32 asm db 66h
```

Using the operand, the following code will move a 32bit value into the EAX register.

```
OPERAND32
asm mov ax,Value
```

The following example program will demonstrate the use of these functions to read the processor ID. For the Vortex86SX the ID should be 031504D44(hex) and for the Vortex86DX it should be 03254D44(hex).

This value is stored in the Customer ID register (CID) which is a 32 bit register starting at North Bridge location 90h.

This example only uses the inport32() function, however the outport32() function has been included in the source for completeness.

To access the PCI configuration space we will use the ports at locations CF8h/CFCh: CF8 is the index port and CFC is the data port.

The mapping of the index port is:-

Bits	Description
31	Configuration Enable. 1=access enabled 0=Access disabled
30-24	Reserved
23-16	Bus number
15-11	Device number*
10-8	Function number
7-2	Register number
0-1	Reserved

To access a register, first we need to set the index port to the register location, remember the MSB must be set for the access to be enabled. Once this is done the register can be read or written by reading from or writing to the data port.

*The Northbridge configuration space register is device zero. The Southbridge configuration space register is device seven.

```
#include <dos.h>
#include <stdio.h>


#define PCI_ADDRESS 0xcf8
#define PCI_DATA 0xcfc

//------ 32 bit operations
#define OPERAND32 asm db 66h

// prototypes
void outport32(unsigned long, unsigned long);
unsigned long inport32(unsigned long);

void main (void)
{
 unsigned long Value;

 Value=inport32(0x80000090); // Config enable set, bus 0, device 0, function 0 , register 90h
printf("ID = %lX\n\r",Value);

} // end of main


void outport32(unsigned long Address, unsigned long Data)
{
 asm mov dx,PCI_ADDRESS
 OPERAND32
 asm mov ax,Address
 OPERAND32
 asm out dx,ax
 asm mov dx,PCI_DATA
 OPERAND32
 asm mov ax,Data
 OPERAND32
 asm out dx,ax
}

unsigned long inport32(unsigned long Address)
{
 long int Value;

 asm mov dx,PCI_ADDRESS
 OPERAND32
 asm mov ax,Address
 OPERAND32
 asm out dx,ax
 asm mov dx,PCI_DATA
 OPERAND32
 asm in ax,dx
 OPERAND32
 asm mov Value,ax
 return Value;
}
```

Disclaimer

All information contained in this application note is believed to be accurate and reliable. However, DSL Ltd assumes no responsibility for its use. Since conditions of product use are outside our control, we make no warranties express or implied in relation thereto. We therefore cannot accept any liability in connection with any use of this information. Nothing herein is to be taken as a license to operate under or recommendation to infringe any patents.

Whilst every effort has been made to ensure that this document is correct; errors can occur. If you find any errors or omissions please let us know so that we can put them right.