Preliminary Information

Processor Family BIOS Writers Guide v4.0

This page is unintentionally left blank

Introduction

The IDT WinChip family of microprocessors has been designed to provide high performance at a low cost. They are x86-compatible processors with unique features and restrictions. This guide should serve as a starting point for any BIOS programmers adapting their BIOS code to recognize and support the IDT WinChip Processor Family CPUs. This is not a stand-alone document; it is meant as a companion to the IDT WinChip Processor Family Data Sheet***. The Data Sheet provides more detailed information***.

The following table outlines several features:

	Voltage	**MMX**	**3DNow!**	**100 MHz bus**	**L1 Cache**
WinChip C6	3.52V	Yes	No	No	64k
WinChip 2	3.52V	Yes	Yes	No	64k
WinChip 2A	3.52V	Yes	Yes	Yes	64k
WinChip 2B	2.8V	Yes	Yes	Yes	64k
WinChip 3	2.8V	Yes	Yes	Yes	128k

TABLE: IDT WinChip Features

This document will outline the necessary requirements for writing a WinChip compatible BIOS.

Identification

IDT WinChip processors can be identified with the CPUID instruction **(opcode 0x0FA2).**

Function 0

Input:	EAX=0	
Output	EAX=1	
	EBX=0x746E6543	"tneC"
	EDX=0x48727561	"Hrua"
	ECX=0x736C7561	"slua"

Note that Vendor ID string in EBX:EDX:ECX is "CentaurHauls".

Function 1

Input:	EAX=1
Output	EAX[3:0]=Stepping
	EAX[7:4]=Model
	EAX[11:8]=Family
	EAX[31:12]=*Reserved*
	EBX=*Reserved*
	ECX=*Reserved*

The following table displays the IDT WinChip values returned by CPUID function 1.

	Family	**Model**	**Stepping**
WinChip C6	5	4	0-1
WinChip 2	5	8	5
WinChip 2B	5	8	A-F
WinChip 2A	5	8	7-9
WinChip 3	5	9	0-F

The Stepping ID field encodes manufacturing process technology as shown in the following table:

Stepping	**Technology**
0-4	IDT 9.5 (0.35u)
5-9	IDT 10.5 (0.25u)
10-15	IBM 6SF (0.25u)

Extended Functions

EAX Input	Title	Output
0x80000000	Largest Extended Function Input Value	EAX=0x80000005 EBX,ECX,EDX=*Reserved*
0x80000001	Processor Signature and Extended Feature Flags	EAX=**Processor Signature** EBX,ECX= *Reserved* EDX=Extended Feature Flags
0x80000002	**Processor Name String**	EAX,EBX,ECX,EDX
0x80000003	**Processor Name String**	EAX,EBX,ECX,EDX
0x80000004	**Processor Name String**	EAX,EBX,ECX,EDX
0x80000005	TLB and Cache Information	EAX=*Reserved* EBX=TLB info ECX=L1 Data Cache info EDX=L1 Instruction Cache info

Processor Signature	EAX[3:0]=Stepping
	EAX[7:4]=Model
	EAX[11:8]=Family
	EAX[31:12]=*Reserved*
	EDX[31]=1 for AMD 3DNow! instructions supported EDX[31]=0 for no AMD 3DNow! instructions

Example:
The string “IDT WinChip 2” would be returned by extended function eax = 0x80000002 as follows:

Processor Name String	EAX= 0x20544449
	EBX= 0x436E6957
	ECX= 0x20706968
	EDX= 0x00000032

Since the string is only 13 bytes, the extended functions eax = 0x80000003 and eax = 0x80000004 return zero in eax, ebx, ecx and edx. A longer display string would require the use of functions eax = 0x80000003 and eax = 0x80000004

Clocks

WinChip family clock multipliers

BF[2]	BF[1]	BF[0]	AMD K6-2	WinChip C6 WinChip 2 WinChip 2B	WinChip 2A	WinChip 3
1	0	0	2.5x	2.0x	2.5x	2.5x
1	0	1	3.0x	3.0x	3.0x	3.0x
1	1	0	2.0x*	2.0x	3.33x/2.0X	3.33X/2.0x
1	1	1	3.5x	4.0x	3.5x	3.5x
0	0	0	4.5x	4.0x	4.5x	4.5x
0	0	1	5.0x	5.0x	2.33x	2.33x
0	1	0	4.0x	4.0x	4.0x	4.0x
0	1	1	5.5x	5.0x	2.66x	2.66x

TABLE: Clock Multiplier Values

*AMD K6-3 has replaced 2.0x with 6.0x

For WinChip 2, WinChip 2B, WinChip 2A, and WinChip 3, the CPU clock multipliers can be determined via software. **WinChip C6 clock ratio cannot be determined by software**. The method for determining WinChip 2 and WinChip 2B are different from WinChip 2A & 3. WinChip 2/2B reports its clock multipliers from register MSR 0x10A [1:0]. WinChip 2A and WinChip 3 are determined by two registers, MSR 0x147 [26:23] and MSR 0x10A [1:0]. The following table outlines this information:

WinChip 2 WinChip 2B		WinChip 2A WinChip 3		
Ratio	**MSR 0x10A [1:0]**	**Ratio**	**MSR 0x147 [26:23]**	**MSR 0x10A [1:0]**
		2.0x	0010b	00b
2.0x	00b	2.5x	0011b	00b
3.0x	01b	3.0x	0100b	00b
4.0x	10b	3.33x	1000b	01b
5.0x	11b	3.5x	0101b	00b
		4.5x	0111b	00b
		2.33x	0101b	01b
		4.0x	0110b	00b
		2.66x	0110b	01b

TABLE: Clock Multiplier Software Values

Formula: (MSR 0x147[26:32]+2) / (MSR 0x10A[1:0]+2)

Example:

Winchip 2A, WinChip 3 (Ratio 2.33x) = (0101b + 2h) / (01b + 2h) = 7 / 3 = 2.33

Winchip 2, WinChip 2B (Ratio 3x) = (01 +2h) = 3

Clock Frequency Determination

The best algorithm for determining the clock frequency is to make use of the RDTSC instruction. This instruction yields a running 64-bit count of the number of processor clocks since powerup. This count can be run against either of two independent clocks in the PC architecture. These two clock sources can be programmed to generate interrupts and the TSC count can be compared between two successive interrupts. This count is accurate enough to place the CPU into one of several “megahertz bins.” A system will generally support several bus speeds, such as 66MHz, 75MHz, 83MHz or 100MHz. These frequencies, multiplied by the available multipliers, yield the aforementioned megahertz bin. The two sources are the real-time clock, which will generate the so-called “periodic interrupt”, and the system timer tick, with its frequency of close to 18.2 times per second.

Models

WinChip C6, WinChip 2, and WinChip 3 provide a variety of performance ranges, so it is necessary to differentiate the CPU's using a performance rating (PR). If PR Rating is not used, then please identify the processor speed by its real MHz.

	When to use PR Rating	**Bios Display**
WinChip C6	Never	IDT WinChip C6 – MHz value
WinChip 2, 2A, 2B	When Bus Frequency is >= 100Mhz	See PR table
WinChip 3	Always used	See PR table

WinChip processors have a variety of speed grades and bus frequencies. The following table enumerates our models:

Model Number	**Internal Frequency**	**Clock Multiplier**	**Bus Frequency**
IDT WinChip C6 - 180	180	3X	60 MHz
IDT WinChip C6 - 200	200	3X	66 MHz
IDT WinChip C6 - 225	225	3X	75 MHz
IDT WinChip C6 - 240	240	4X	60 MHz
IDT WinChip 2 - 200	200	3X	66 MHz
IDT WinChip 2 - 225	225	3X	75 MHz
IDT WinChip 2 - 233	200	2X	100 MHz
IDT WinChip 2 - 233	208	2.5X	83 MHz
IDT WinChip 2 - 233	233	3.5X	66 MHz
IDT WinChip 2 - 240	240	4X	60 MHz
IDT WinChip 2 - 266	233	2.33X	100 MHz
IDT WinChip 2 - 266	250	3X	83 MHz
IDT WinChip 2 - 266	266	4X	66 MHz
IDT WinChip 2 - 300	250	2.5X	100 MHz
IDT WinChip 2 - 300	266	2.66X	100 MHz
IDT WinChip 3 - 233	200	3X	66 MHz
IDT WinChip 3 - 266	200	2X	100 MHz
IDT WinChip 3 - 266	233	3.5X	66 MHz
IDT WinChip 3 - 300	233	2.33X	100 MHz
IDT WinChip 3 - 300	266	4X	66 MHz
IDT WinChip 3 - 333	250	2.5X	100 MHz
IDT WinChip 3 - 333	266	2.66X	100 MHz
IDT WinChip 3 - 333	300	4.5X	66 MHz
IDT WinChip 3 - 350	285	3X	95 MHz
IDT WinChip 3 - 366	333	5X	66 MHz
IDT WinChip 3 - 366	300	3X	100 MHz
IDT WinChip 3 - 380	316	3.33X	95 MHz
IDT WinChip 3 - 400	333	3.33X	100 MHz

TABLE: IDT WinChip Models

*Note that WinChip 2, WinChip 2A, and WinChip 2B should all display the same model number.

Jumperless Motherboard Considerations

Jumperless motherboards require that the system BIOS configure the CPU's supply voltage, clock multiplier, and bus frequency. It is important for the BIOS programmer to power the CPU in a safe configuration to prevent overclocking or driving too high/low a voltage. These values vary throughout the IDT WinChip product line.

	Family Model	Stepping	Voltage	Min Core Frequency	Min Bus Frequency	Min Clock Multiplier
IDT WinChip C6	54	All	3.52V	180 MHz	60 MHz	3X
IDT WinChip 2	58	0-6	3.52V	200 MHz	66 MHz	3X
IDT WinChip 2A	58	7-9	3.52V	200 MHz	66 MHz	3X
IDT WinChip 2B	58	A-F	2.8V	200 MHz	66 MHz	3X
IDT WinChip 3	59	All	2.8V	200 MHz	66 MHz	3X

Performance Optimizations

The IDT WinChip processors provide extensions over the P55 to define variable size memory ranges with associated special attributes. These Memory Configuration Registers (MCRs) are similar to the MTRRs of the Pentium Pro processor, and the Address Region Registers (ARRs) of the Cyrix 6x86MX/MII processor. All of these approaches perform similar functions but differ in specifics. In fact, there are differences in the MCR details between the IDT WinChip 2, IDT WinChip 3, and its predecessor, the IDT WinChip C6. To implement the features provided by these registers, one must configure the correct machine specific register (MSR).

MSR	ECX	EDX	EAX	Type
MCR 0	110h	Base Address [31:12]	Address Mask [31:12] & Ctrl Value [11:0]	WO
MCR 1	111h	Same as above	Same as above	WO
MCR 2	112h	Same as above	Same as above	WO
MCR 3	113h	Same as above	Same as above	WO
MCR 4	114h	Same as above	Same as above	WO
MCR 5	115h	Same as above	Same as above	WO
MCR 6	116h	Same as above	Same as above	WO
MCR 7	117h	Same as above	Same as above	WO
MCR_CTRL	120h		Control Value	WO

TABLE: Memory Configuration Registers

The intent is to define memory regions and special attributes for these regions such as byte-combining and weakly ordered stores. These special attributes are deviations from formal x86 architectural behavior, but, in practice, work fine for specific memory regions in a PC architecture. The advantage of these special attributes is improved performance for the affected memory regions.

MCR_CTRL (MCR Control Register) MSR 0x120

The MCR_CTRL MSR controls various pervasive behaviors of byte-combining and write-ordering. The store combining definitions for string and non-stack and non-string are defined in the following table.

Bit	Description	Default	Notes
1:0	Store combining definition for non-stack & non-string. 00: Forward Combining 01: Forward and Overlapped Combining 10: Forward and Reverse Combining 11: Forward, Reverse, and Overlapped Combining	00	Read-Write
3:2	Store combining definition for string. 00: Forward Combining 01: Forward and Overlapped Combining 10: Forward and Reverse Combining 11: Forward, Reverse, and Overlapped Combining	00	Read-Write
4	Weak Write-Ordering Enable	0	Read-Write
5	Reserved	0	
8:6	Trait Mode Control, must match MCR_CTRL[19:17] in order to enable Memory Configuration Registers: 001: Enables IDT WinChip 2,and WinChip 3 MCRs	000	Read-Write

	other: Disables MCRs Before programming the MCRs system software should identify the processor version to ensure that the MCRs are programmed appropriately. MCR_CTRL[19:17] contains a unique value that identifies the version of the MCR traits supported by the processor. System software should copy MCR_CTRL[19:17] to MCR_CTRL[8:6] if, and only if, it recognizes the version.		
9	MCR 0 in use 0: MCR0[4:0] attributes are all zero 1: MCR0[4:0] attributes are non-zero	0	Read-Only (RO)
10	MCR 1 in use (see MCR 0) *Not supported on WinChip C6.*	0	RO
11	MCR 2 in use (see MCR 0) *Not supported on WinChip C6.*	0	RO
12	MCR 3 in use (see MCR 0) *Not supported on WinChip C6.*	0	RO
13	MCR 4 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO
14	MCR 5 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO
15	MCR 6 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO
16	MCR 7 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO
19:17	Trait Mode Key, must write this value to MCR_CTRL[8:6] in order to enable Memory Configuration Registers. System software should copy MCR_CTRL[19:17] to MCR_CTRL[8:6] if, and only if, it recognizes the version.	001	Read-Only
24:20	Reserved	11111	Read-Write

Further MCR_CTRL details

If system software is only concerned with programming the memory configuration registers, then it can read the MCR_CTRL register (MSR 0x120) and inspect the Trait Mode Key field (MCR_CTRL[19:17]). ***In the IDT WinChip 2 and WinChip 3, the Trait Mode Key must be written to the Trait Mode control field (MCR_CTRL[8:6]) in order to activate the memory configuration registers***. **WinChip C6 does not require this.** System software should determine whether it recognizes the value of the trait mode key before writing it to MCR_CTRL[8:6]. For the IDT WinChip 2 and WinChip 3 the value is 001b. If the value is not recognized by the software, the key *must not be written* to avoid serious problems.

Trait Mode Key (MCR_CTRL[19:17]) & Trait Mode Control (MCR_CTRL[8:6])

Family	Model	Trait Mode Key MCR_CTRL[19:17]	Processor Model
5	4	000	IDT WinChip C6
5	8	001	IDT WinChip 2,A,B
5	9	001	IDT WinChip 3

If the processor version is not recognized, then system software must not attempt to activate any machine-specific features.

MCR In Use

If system software such as the operating system or device drivers wish to reprogram any of the memory control registers, certain steps must be taken to avoid conflicts and unpredictable behavior. There is a field of 8 bits in the MCR Control Register (MSR 0x120). These bits indicate which memory control registers (MCRs) are in use (with non-zero traits).

The following table describes the relevant bits of the MCR Control Register

Bit	Description	Default	Notes
9	MCR 0 in use 0: MCR0[4:0] attributes are all zero 1: MCR0[4:0] attributes are non-zero	0	Read-Only (RO)
10	MCR 1 in use (see MCR 0) *Not supported on WinChip C6.*	0	RO
11	MCR 2 in use (see MCR 0) *Not supported on WinChip C6.*	0	RO
12	MCR 3 in use (see MCR 0) *Not supported on WinChip C6.*	0	RO
13	MCR 4 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO
14	MCR 5 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO
15	MCR 6 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO
16	MCR 7 in use (see MCR 0) *Not supported on WinChip C6*.	0	RO

Store-Combining (MCR_CTRL[0:3])

The IDT WinChip C6, WinChip2, and WinChip 3 processor's store-combining feature allows multiple stores to be combined into a single bus store. This is permissible if the stores are destined to the same 8-byte memory address.

Store-combining can greatly reduce the memory bandwidth requirements of writes that miss the cache. However, if the associated writes are destined to memory mapped I/O locations, problems can arise.

For example, if an ISA bus 8-bit device is controlled with a data register at address 0 and a control register at address 1, and the control register must be written before the data can be written, it is possible for the order of writes to be changed if byte-combining is inappropriately configured.

If, for example, the device writes to address 1 and then to address 0 and the two are combined, a 16-bit word will go to the ISA bus where it will be split for the 8-bit device into two writes. Unfortunately, most ISA bridges split 16-bit operands into two transfers with the low byte first. Consequently, the order of the two writes are reversed.

To eliminate this problem, all WinChip processors are very configurable. Aside from disabling byte-combining, it is also possible to limit the type of instructions that are allowed to combine. Further, it is possible to prevent the processor from combining in reverse address order. Lastly, it is possible to prevent the processor from combine-matching (overlapping) byte addresses.

Weak Write-Ordering (MCR_CTRL[4])

The Pentium processor normally mandates that writes occur in the memory hierarchy in the same order as they occur in the code execution. This is termed strong write-ordering. This restriction can be a performance impact in that it blocks processor execution when a store hits an E or M line in the cache if another store is waiting to be retired on the bus

Normally systems do not require strong write-ordering unless they have bus-mastering I/O devices that use memory mapped I/O for control purposes. (Most DMA devices are slaves, and do not use memory-mapped I/O. The floppy controller, for example, is a DMA device, but does not use memory-mapped I/O.)

However, since there are devices that could not perform correctly with weak write ordering, this function should only be used in systems where the type of peripherals are tightly controlled and known to not require strongly ordered writes. ***Weak Write Ordering should never be turned on by a generic BIOS*****,** for example

MCR (Memory Configuration Registers

The IDT WinChip C6, WinChip 2, and WinChip 3 processors all have eight MCRs

Each MCR appears as a 64-bit MSR. The default value at reset is zero for all fields. This value causes all memory to have normal x86 attributes of no byte-combining and strongly ordered writes. **Note that the MCRs are *write-only*.**

EDX 31:12	EDX 11:0	EAX 31:12	EAX 11:5	EAX 4:0
Base Address of Region	*Res*	Mask Defined Region	*Res*	Attributes
20	12	20	7	5

MCR Format

Base Address of Region (EDX[31:12])

This is the starting physical address of the memory region. Each definable memory region starts at a 4-KB page boundary; thus the low order 12 bits of the address are ignored.

Mask Defined Region (EDX[11:0])

This is a bit mask defining the size of the memory region. A memory region hit exists for a MCR if: Mask address AND memory address = Base address AND mask address in all bit positions (31:12).

Example 1

For example, consider the memory range from 0x000A0000 to 0x000BFFFF. This is most efficiently done by masking off the low order bits that constitute the range.
Viewing the addresses in binary:

```
0x000A0000   =     0000  0000  0000  1010  0000  0000  0000  0000
0x000BFFFF   =     0000  0000  0000  1011  1111  1111  1111  1111
```

Notice that the upper 15 bits are identical, whereas the lower 17 bits define the address within the range. So a single MRD can describe this range as:

```
MCR Base     =     0000  0000  0000  1010  0000  or 0x000A0
MCR Mask     =     1111  1111  1111  1110  0000  or 0xFFFE0
```

Note that the lower twelve bits of the mask and base are ignored in the match calculation.

Example 2

Consider a more complex scenario where a range from 0x00080000 to 0x0017FFFF is required. Shown in binary as:

```
0x00080000   =     0000  0000  0000  1000  0000  0000  0000  0000
0x0017FFFF   =     0000  0000  0001  0111  1111  1111  1111  1111
```

In this case, the upper 11 bits are identical throughout the range. The next two bits vary, but not all combinations are part of the desired range. The range has to be broken down into two ranges that have common base bits where all combinations of the lower order bits are within the original region. This implies:

```
0x00080000   =     0000  0000  0000  1000  0000  0000  0000  0000
0x000FFFFF   =     0000  0000  0000  1111  1111  1111  1111  1111
```

and

```
0x00100000   =     0000  0000  0001  0000  0000  0000  0000  0000
0x0017FFFF   =     0000  0000  0001  0111  1111  1111  1111  1111
```

These map into MCRs:

```
 MCR0 Base   =     0000  0000  0000  1000  0000  or 0x00080
 MCR0 Mask   =     1111  1111  1111  1000  0000  or 0xFFF80
```

and

```
 MCR1 Base   =     0000  0000  0001  0000  0000  or 0x00100
 MCR1 Mask   =     1111  1111  1111  1000  0000  or 0xFFF80
```

Attributes (EAX[4:0])

There are five bits of attribute control defined for each memory region as described in Table . Note that this definition is different from the predecessor IDT WinChip C6. In order to avoid unintended programming by system software the IDT WinChip 2 defines the Trait Mode Control field in MCR_CTRL, which must be set to the appropriate value in order for the MCRs to be enabled

MCR Attribute bits for WinChip C6

Bit	Description	Default	Notes
0	Enables byte combining on non-stack non-string writes.	0	
1	Enables byte combining on string instruction writes.	0	
2	Enables byte combining on stack instruction writes.	0	
4:3	Defines write-ordering strategy for memory region. 00: All writes strongly ordering 01: String writes weakly ordered 10: Stack writes weakly ordered 11: All writes weakly ordered	00	

MCR Attribute bits for WinChip 2,A,B and WinChip 3

Bit	Description	Default	Notes
0	Enables store combining	0	
1	Accesses within this range will not disrupt the cache. This is useful for describing a video buffer. It is a hint to the processor that the memory controller will consider this non-cacheable (KEN# not asserted) and therefore the processor can potentially avoid a castout of a modified line.	0	
2	Reserved.	0	
3	Weak Write Ordering (WWO) bit. Allows writes to be reordered with respect to each other, but this may cause issues with DMA-ing devices.	0	
4	Weak Read Ordering (WRO) bit. This allows reads to be reordered in front of writes to different cache lines. This bit can usually be set for cacheable memory regions.	0	

Combining Ranges

It is possible to describe fairly complex ranges with a few descriptors. Generally, this does not involve overlapping MCRs. However, overlapping ranges are permitted and their behavior is useful in some cases.

The behavior of an access to a given memory location is defined by the logical OR of the attribute bits of the MCRs it matches. So, if a memory location does not match any MCRs, its aggregate attribute is 0. If, on the other hand, it matches two MCRs, one with an attribute of 0x10h and the other with an attribute of 0x01h, the aggregate attribute is 0x11h

For WinChip C6, an MCR attribute of 0x11h means “Enables byte combining on non-stack non-string writes” and “Stack writes weakly ordered”. WinChip C6 then looks into MCR_CTRL [1:0] to see how store combining is used for non-stack non-string writes, and checks if Weak Write-Ordering is Enable as defined by MCR_CTRL[4].

For WinChip 2 and WinChip 3, an MCR attribute of 0x11 means enabling weak read ordering on all accesses and allowing store-combining. WinChip 2 and WinChip 3 then looks into MCR_CTRL[0] to see if store combining is turned on.

BIOS Code

To summarize, there are three key features described in this guide that can improve the processor's performance. The following table will explain why the bios code in the next section will NOT take advantage of all these features.

	WinChip C6	WinChip 2 & WinChip 3
Weak Write Ordering	Not turned on because some devices don't work correctly with Weak Write Ordering enabled	
Weak Read Ordering	Not supported on WinChip C6	**On**
Store Combining	**On**	**On**

Store Combining and Weak Read Ordering

These are features that are described in detail in the Data Sheet. Briefly, When write-combining is enabled, the processor will attempt to accumulate memory writes of less than a full dword. This reduces the memory access overhead and improves performance. Weak read ordering allows read operations to be re-ordered ahead of writes to different cache lines, which can result in data being available to a program earlier

Remember the actual implementation of these features varies between IDT WinChip processors (In the future, WinChip 4 and 5 will have different methods of specifying the Memory Range). So it is ***imperative*** to check the ***Trait Mode Key (MCR_CTRL[19:17])*** before enabling the features

These concepts are better described in the sample source code below. This is a very basic algorithm designed for simplicity to illustrate the concepts. Refinements to this algorithm can be made easily to accommodate other memory capacities. Please refer to the data sheet for more detailed information. Areas of improvement to this very basic algorithm might include better support for memory sizes not a power of two megabytes. The algorithm shown treats anything greater than the closest power of two megabytes as non-write-combining space, so for example if the memory size is 40 MB then the region from 32-40MB is not enabled for write-combining. Better support for the so-called "OS/2 memory hole" is also possible. When enabled, this hole is from 15-16MB. The sample algorithm will create a non-write-combining, non-weak-read-ordered hole from 12-16MB.

Note the use of cpuid with eax = 0xC0000000 to determine which IDT WinChip is installed. The sample code will also set the EDCTLB bit in the Feature Control Register, required for WinChip C6. Note also the use of the Trait Mode Key as described above

Sample Algorithm to Enable Write-Combining

```
; This sample source code is placed in the public domain.   ; Centaur
Technology, Inc. disclaims all warranties, express ; or implied,  and
all liability, including consequential   ; and other indirect damages,
for the use of this source     ; code, including the warranties of
merchantability and      ; fitness for a particular purpose.
; Centaur Technology, Inc. does not assume any              ;
responsibility for any errors which may appear in this     ; program
nor any responsibility to update it.


;=========================================================
; IDT WinChip 2 Write Combining Support
;=========================================================

; base/mask pairs for memory ranges

; 0 - 512K
base_0          equ     00000000000000000000000000000000b
mask_512K       equ     11111111111100000000000000000000b

; 512K 640K
base_512K       equ     00000000000010000000000000000000b
mask_640K       equ     11111111111111100000000000000000b
; 1M - 2M
base_1          equ     00000000000100000000000000000000b
mask_2          equ     11111111111000000000000000000000b

; 2M - 4M
base_2          equ     00000000001000000000000000000000b
mask_4          equ     11111111110000000000000000000000b

; 4M - 8M
base_4          equ     00000000010000000000000000000000b
mask_8          equ     11111111100000000000000000000000b

; 8M - 16M
base_8          equ     00000000100000000000000000000000b
mask_16         equ     11111111000000000000000000000000b
mask_12         equ     11111111100000000000000000000000b

; 16M - 32M
base_16         equ     00000001000000000000000000000000b
mask_32         equ     11111110000000000000000000000000b

; 32M - 64M
base_32         equ     00000010000000000000000000000000b
mask_64         equ     11111100000000000000000000000000b

; 64M - 128M
base_64         equ     00000100000000000000000000000000b
mask_128        equ     11111000000000000000000000000000b

; 128M - 256M
base_128        equ     00001000000000000000000000000000b
```

```
mask_256        equ     11111000000000000000000000000000b

; 256M - 512M
base_256        equ     00010000000000000000000000000000b
mask_512        equ     11111000000000000000000000000000b

; 512M - 1024M
base_512        equ     00100000000000000000000000000000b
mask_1024       equ     11100000000000000000000000000000b

; 1024M - 2048M
base_1024       equ     01000000000000000000000000000000b
mask_2048       equ     11000000000000000000000000000000b

; 2048M - 4096M
base_2048       equ     10000000000000000000000000000000b
mask_4096       equ     10000000000000000000000000000000b



; programming table for memory range registers

IDT_MCR_table   label   dword
                dd      base_2048
                dd      mask_4096

                dd      base_1024
                dd      mask_2048


                dd      base_512
                dd      mask_1024

                dd      base_256
                dd      mask_512

                dd      base_128
                dd      mask_256

                dd      base_64
                dd      mask_128

                dd      base_32
                dd      mask_64

                dd      base_16
                dd      mask_32

                dd      base_8
                dd      mask_16

                dd      base_4
                dd      mask_8

                dd      base_2
                dd      mask_4
```

```
                dd      base_1
                dd      mask_2

                dd      base_512K
                dd      mask_640K

                dd      -1           ;-1 means end of IDT_MCR_table


MCRbaseno       equ     110h   ;MSR of the MCR0
MCR_CTRL        equ     120h        ;MSR of MCR_CTRL

MCRvalue        equ     11111000000000000000001111b
                        ;Turn on Forward, Reverse and
                        ;overlapping combining for all
                        ;data type

FCRno                equ        107h     ;MCR of FCR
DTLOCKbit            equ        8        ;DTOCK is the 8th  bit of FCR



;=========================================================
;Name     : Winchip Memory Features
;
; ** This routine should be called ONLY AFTER an IDT
;    WinChip CPU has been identified
;
;Function : To enable the IDT WinChip write combining and
;            weak read ordering
;
;Inputs:  edi = memory size in MB
;         bp = 1 if memory hole exists from 15 to 16MB
;              0 if not
;
;=========================================================


WinChip_Memory_Features  Proc    near

                pushad

; bh has MCR attribut bits,
; bl has trait mode key bl = 000 if C6, 001 if WinChip 2

; first assume C6
; MCR attribute bh=111b, trait mode key bl=000b
; Turn on byte combining for all data types
                mov     ebx, 11100000000b

; now check trait mode key to see if which processor
; we have to skip the programming if we are dealing with the ;
                mov     ecx, MCR_CTRL
                rdmsr
                shr     eax 17      ;move [19:17] to [2:0]
```

```
                and     eax, 111b   ;mask all the other bit
                cmp     eax, 000b
                jz      sub_1M          ; if 000, it's C6
                 cmp     eax, 001b     ; if 001,
                                        ; it's WinChip 2 & 3
                jne     No_Memory_Features  ;skip if not 1


; here if WinChip 2
; traits to turn on are weak read ordering and byte
; combining

; MCR attribute=10001b, trait mode key bl=001b
                mov     ebx, 1000100000001b


sub_1M:
        ; always set up 0-512K region

               mov     ecx, MCRbaseno

               mov     edx, base_0
               mov     eax, mask_512K
               or      al, bh  ;OR the local attributes
               wrmsr

; now calculate offset into table to program the rest of
; the memory range registers
; bsr returns the bit index of the highest 1 (largest power
; of 2 in the number)
;edi = memory size in MB

                bsr     ecx, edi
                neg     ecx
                lea     esi, [8*ecx+offset IDT_MCR_table+96]


; esi now contains the offset to the beginning of largest
; memory region

                mov     ecx, MCRbaseno+1
program_one_mcr:
                mov     edx, cs:dword ptr [si]

                cmp     edx, -1
                je      short mcr_done

                mov     eax, cs:dword ptr [si+4]

                cmp     edx, base_8
                jne     short around_hole_check
                or      bp, bp
                jz      short around_hole_check
                mov     eax, mask_12 ;hole enabled 12-16MB
around_hole_check:
                or      al, bh  ;OR in the local trait bits
```

```
                wrmsr

                add     si, 8
                inc     ecx
                cmp     ecx, MCRbaseno+8
                jb      program_one_mcr
mcr_done:

        ; finally set up MCR_CTRL
; no trait mode control for c2c
                mov     edx, 0
                mov     ecx, MCR_CTRL
                test    bl, bl    ; check for WinChip 2
                jz      c2cwc     ; if C6


; if WinChip 2, need to get trait mode control bits
                rdmsr     ; bits 19:17 trait mode control

; isolate these bits
                and     eax, 111b shl 17
                shr     eax, 11      ; move them to 8:6
                mov     edx, eax   ; save the isolated bits

c2cwc:
                mov     eax, MCRvalue  ; get control value
                or      eax, edx  ; OR in trait mode control
                mov     edx, 0    ; prepare to write back
                wrmsr

                mov     ecx, FCRno
                rdmsr
                or      ax, 1 shl DTLOCKbit
                wrmsr

No_Memory_Features:
                popad
                ret

WinChip_Memory_Features  endp
```

Feature Control (FCR) MSR 0x107

The FCR controls the major optional feature capabilities of the IDT WinChip processor. It is analogous to the Pentium processor TR12 (actually MSR 0Eh) which controls things like BTB enable, cache enable, and so forth. The Cyrix 6x86MX/MII processor's CCRs (Configuration Control Registers) perform a similar function, as does the AMD-K6 processor's HDCR MSR.

The table below contains the bit values for the FCR. The default settings shown for the FCR bits are not necessarily exact within a WinChip model line. The actual settings can be changed as part of the manufacturing process and thus a particular IDT WinChip processor version can have slightly different default settings than shown here. All reserved bit values of the FCR must be preserved by using a read-modify-write sequence to update the FCR.

WinChip C6 has different definitions for the FCR from WinChip 2,A,B, and Winchip 3.

WinChip C6 FCR Bit Assignments

Bit	*Name*	*Description*	*Default*
0		*Reserved*	0
1	ECX8	Enables CPUID reporting CX8	0
2	EIERRINT	Enables INT18 (Machine Check) for internal errors	0
3	DPM	Disable dynamic power management	0
4	DMCE	Disables Machine Check Exception	0
5	DSTPCLK	Disables supporting STPCLK	0
6	ELINEAR	Enables Linear Burst Mode	0
7	DSMC	Disables strict cache coherency (self-modifying-code)	0
8	EDCTLB	Enables D-Cache for updates to PDE/PTE	0
9	EMMX	Enables MMX-compatible instructions.	1
10		*Reserved*	0
11	DPDC	Disables Page Directory cache	0
12		*Reserved*	0
13	DIC	Disables I-Cache.	0
14	DDC	Disables D-Cache.	0
15	DNA	Disables bus pipelining (NA response)	0
16	ERETSTK	Enables CALL-RET Stack operation	1
17		*Reserved*	0
18		*Reserved*	0
19		*Reserved*	0
20		*Reserved*	0
21		*Reserved*	0
22:25	SID	Stepping ID	0
26		*Reserved*	0
27		*Reserved*	0
28	ADIVFLG	Selects alternate EFLAGS results for divide	0
29	DCPUID	Disables CPUID instruction	0
30	EMOVTR	Enables move-to-test register instructions	0
31		*Reserved*	0

ECX8: 0 **=** The CPUID instruction does not report the presence of the CMPXCHG8B instruction (CX8 = 0)**.** The instruction actually exists and operates correctly, however.

1 = The CPUID instruction reports that the CMPXCHG8B instruction is supported (CX8 = 1).

EIERRINT: 0 = Normal internal error behavior (IERR and possible Shutdown**).**

1 = Causes INT18 instead of Shutdown for internal error.

DPM: 0 = Normal dynamic power management behavior.

1 = Disables all dynamic power management.

DMCE: 0 = Machine Check exception enabled.

1 = Disable Machine Check exception; a bus check or internal error condition does not cause an exception.

DSTPCLK: 0 = STPCLK interrupt properly supported.

1 = Ignores SPCLK interrupt.

ELINEAR: 0 = Interleaved burst ordering is enabled.

1 = Linear burst ordering is enabled.

DSMC: 0 **=** Strict cache coherency is enabled to support Pentium processor style self-modifying code.

1 = Disables strict cache coherency. I-cache/D-cache coherent only if branch is taken after store instruction which modifies instructions to be executed subsequently.

EDCTLB: 0 **=** Updates to the accessed and dirty bits in PDE/PTE entries are performed using the locked read-modify-write semantics which flushes the data from the D-Cache (like Pentium processor).

1 = Enables D-Cache for updates to accessed and dirty bits in PDE/PTE.

EMMX: 0 = Disables MMX-compatible instructions: they decode as invalid instructions.

1 = Enables MMX-compatible instructions.

DPDC: 0 = Enables use of internal Page Directory Cache.

1 = Disables use of internal Page Directory Cache.

DIC: 0 = Enables use of I-Cache.

1 = Disables use of I-Cache: cache misses are performed as single transfer bus cycles, PCD is de-asserted. This overrides any setting of CR0.CD and CR0.NW.

DDC: 0 = Enables use of D-Cache.

1 = Disables use of D-Cache: same semantics as for DIC except for D-Cache.

DNA: 0 = Enables bus pipelining operation.

1 = Disables bus pipelining operation: bus signal NA is ignored.

ERETSTK: 0 **=** Disables CALL-RETurn stack function.

1 = Enables CALL-RETurn stack function: RET branch target prediction is performed.

ADIVFLG: 0 = The EFLAGS setting after integer divide instructions is unique to the IDT WinChip C6 processor. These flag settings are undefined in the x86 architecture.

1 = The EFLAGS setting after integer divide of 5 by 2 is the same as on a Pentium processor (this is used by some old code to identify the processor)

DCPUID: 0 **=** The CPUID instruction is supported.

1 = The CPUID instruction is disabled and causes an invalid instruction exception.

EMOVTR: 0 = The Intel486 move-to/from-test register instructions are not supported and their behavior is the same as on a Pentium processor (invalid instruction exception).

1 = The test register instructions do not cause an invalid instruction exception but rather are treated as NOPs.

WinChip 2,A,B, and WinChip 3 FCR bit assignments

BIT	*NAME*	*DESCRIPTION*	*DEFAULT*
0		Reserved	0
1	ECX8	Enables CPUID reporting CX8	0
2	EIERRINT	Enables INT18 (Machine Check) for internal errors	0
3	DPM	Disable dynamic power management	0
4	DMCE	Disables Machine Check Exception	0
5	DSTPCLK	Disables supporting STPCLK	0
6	ELINEAR	Enables Linear Burst Mode	0
7	DSMC	Disables strict cache coherency (self-modifying-code)	0
8	DTLOCK	Disables locking of updates to accessed or dirty bits in page directory/table entries	1
9	EMMX	Enables MMX-compatible instructions	1
10		Reserved	0
11	DPDC	Disables Page Directory cache	0
12	EBRPRED	Enables Branch Prediction	1
13	DIC	Disables I-Cache.	0
14	DDC	Disables D-Cache.	0
15	DNA	Disables bus pipelining (NA response)	0
16	ERETSTK	Enables CALL-RET Stack operation	1
17		Reserved	0
18		Reserved	1
19	E2MMX	Enables pairing of MMX-compatible instructions	1
20	EAMD3D	Enables AMD-3D compatible instructions	1
21		*Reserved*	1
22:25	SID	Stepping ID	0
26		*Reserved*	0
27		*Reserved*	0
28		*Reserved*	0
29	DCPUID	Disables CPUID instruction	0
30	EMOVTR	Enables move-to-test register instructions	0
31		*Reserved*	0

ECX8: 0 **=** The CPUID instruction does not report the presence of the CMPXCHG8B instruction (CX8 = 0)**.** The instruction actually exists and operates correctly, however.

1 = The CPUID instruction reports that the CMPXCHG8B instruction is supported (CX8 = 1).

EIERRINT: 0 = Normal internal error behavior (IERR# and possible Shutdown**).**

1 = Causes INT18 instead of Shutdown for internal error.

DPM: 0 = Normal dynamic power management behavior.

1 = Disables all dynamic power management.

DMCE: 0 = Machine Check exception enabled.

1 = Disable Machine Check exception; a bus check or internal error condition does not cause an exception.

DSTPCLK: 0 = STPCLK interrupt properly supported.

1 = Ignores SPCLK interrupt.

ELINEAR: 0 = Interleaved burst ordering is enabled.

1 = Linear burst ordering is enabled.

DSMC: 0 **=** Strict cache coherency is enabled to support Pentium processor style self-modifying code.

1 = Disables strict cache coherency. I-cache/D-cache coherent only if branch is taken after store instruction which modifies instructions to be executed subsequently.

DTLOCK: 0 **=** Updates to the accessed and dirty bits in PDE/PTE entries are performed using the locked read-modify-write semantics which flushes the data from the D-Cache (like Pentium processor).

1 = Updates to the accessed and dirty bits in PDE/PTE entries are performed without locking the bus or flushing data from the D-Cache.

EMMX: 0 = Disables MMX-compatible instructions: they decode as invalid instructions.

1 = Enables MMX-compatible instructions.

DPDC: 0 = Enables use of internal Page Directory Cache.

1 = Disables use of internal Page Directory Cache.

EBRPRED: 0 **=** Disables branch prediction function.

1 = Enables branch prediction function.

DIC: 0 = Enables use of I-Cache.

1 = Disables use of I-Cache: cache misses are performed as single transfer bus cycles, PCD is de-asserted. This overrides any setting of CR0.CD and CR0.NW.

DDC: 0 = Enables use of D-Cache.

1 = Disables use of D-Cache: same semantics as for DIC except for D-Cache.

DNA: 0 = Enables bus pipelining operation.

1 = Disables bus pipelining operation: bus signal NA is ignored.

ERETSTK: 0 **=** Disables CALL-RETurn stack function.

1 = Enables CALL-RETurn stack function: RET branch target prediction is performed.

E2MMX: 0 **=** Disables pairing of MMX instructions.

1 = Enables pairing of MMX instructions.

EAMD3D: 0 **=** Disables AMD 3D-compatible instructions.

1 = Enables AMD 3D-compatible instructions.

DCPUID: 0 **=** The CPUID instruction is supported.

1 = The CPUID instruction is disabled and causes an invalid instruction exception.

EMOVTR: 0 = The Intel486 move-to/from-test register instructions are not supported and their behavior is the same as on a Pentium processor (invalid instruction exception).

1 = The test register instructions do not cause an invalid instruction exception but rather are treated as NOPs.

Important FCR bits

DTLOCK FCR[8])

DTLOCK prevents the table walk state machine from performing a LOCK read-modify-write sequence on updates to the Accessed and Dirty bits in the page directory/tables. DTLOCK is controlled by bit 8 of the Feature Control Register (MSR 0x107). Setting this bit to 1 improves the performance of the chip because LOCK operation tends to be slow and is not necessary because IDT WinChip processors do not support multi-processing. WinChip C6 has this feature turned off, so system BIOS should set this bit to 1.

Linear Burst Mode FCR[6]

Linear burst mode improves memory performance in the cases where the first address of a burst read cycle is not a multiple of 16. Linear Burst Mode can be enabled for the IDT WinChip 2 and WinChip 3 by setting bit 6 of the Feature Control Register (MSR 0x107) to 1. *This feature must be supported by the chipset in order to be used. Please refer to the individual chipset documentation for how to enable this feature*. Note that WinChip C6 does not support this feature.