Idt

Processor Family

BIOS Writers Guide

Version 0.9 of this document was released 27 January, 1999. © 1998 Centaur Technology, Inc. All Rights Reserved

Centaur Technology, Inc. reserves the right to make changes in its products without notice in order to improve design or performance characteristics.

This publication neither states nor implies any representations or warranties of any kind, including but not limited to any implied warranty of merchantability or fitness for a particular purpose. No license, express or implied, to any intellectual property rights is granted by this document.

Centaur Technology, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication or the information contained herein, and reserves the right to make changes at any time, without notice. Centaur Technology, Inc. disclaims responsibility for any consequences resulting from the use of the information included herein.

Trademarks

WinChip, IDT-C6, C6, and CentaurHauls are trademarks of Integrated Device Technology Corporation.

AMD, the AMD logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Am486 and AMD-K6 are registered trademarks, and AMD-3D is a trademark of Advanced Micro Devices, Inc.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Intel, the Intel logo, and combinations thereof are trademarks of the Intel Corporation. MMX and Intel486 are trademarks of the Intel Corporation. Pentium is a registered trademark of the Intel Corporation.

Cyrix, the Cyrix logo, and combinations thereof are trademarks of the Cyrix Corporation. Cyrix 6x86MX is a trademark of the Cyrix Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

INTRODUCTION

The IDT WinChip Processor Family is an x86-compatible processor family with unique features and restrictions. This guide should serve as a starting point for any BIOS programmers adapting their BIOS code to recognize and fully utilize 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 the detailed information.

This document discusses the IDT WinChip C6 and (mainly) the various versions of the IDT WinChip 2, with some advance information about the IDT WinChip 3.

Some programming information has changed (see sample code to enable byte combining). The new code is still backward-compatible with the IDT WinChip C6, but will now support future products.

GENERAL COMPATIBILITY

The IDT WinChip Processor Family processors are highly compatible with standard ("586-class") x86 implementations. Some members of the IDT WinChip family also support industry-compatible instruction set extensions that speed up certain graphics operations (AMD-3D Technology). The handling of SMI is compatible with the Intel Pentium (P54C). Different members of the IDT WinChip family support different clock speed multiples. Earlier members of the IDT WinChip family support multipliers of 2, 3, 4 and 5x only.

The WinChip C6 and earlier versions of the WinChip 2 offer no method through software for the programmer to determine the clock multiplier. This situation was remedied in the 100MHz bus versions of the WinChip 2. The versions can be distinguished through the use of the CPUID instruction, described in a later section.

The following table summarizes the IDT WinChip 2 usage of the BF pins as compared to the P54C. Please refer to the Data Sheet for more detailed information.

Bus Frequency Multipliers For WinChip 2 (original version)

Pins			Clock Multiplier	
BF2	BF1	BF0	IDT WinChip 2	Intel P54C
1	0	0	2x	2.5 x
1	0	1	3 x	3 x
1	1	0	2 x	2 x
1	1	1	4 x	1.5 x
0	0	0	4x	
0	0	1	5 x	
0	1	0	4 x	
0	1	1	5x	

The 100MHz bus frequency version, the IDT WinChip 2A, supports a different set of clock multipliers, including fractional multipliers not supported by previous versions. These multipliers are summarized in the following table.

Bus Frequency Multipliers WinChip 2A (100MHz Bus version) and WinChip 3

Pins			Clock Multiplier
BF2	BF1	BF0	IDT WinChip 2A
1	0	0	2.5x
1	0	1	3 x
1	1	0	3.33 x
1	1	1	3.5 x
0	0	0	4.5 x
0	0	1	2.33 x
0	1	0	4 x
0	1	1	2.66 x

In the WinChip 2A, the multiplier information can be read from machine-specific registers (MSR 0x10A and 0x147) as follows:

Ratio	MSR 0x147 [26:23]	MSR 0x10A [1:0]
2.5X	0011b	00b
3.0X	0100b	00b
3.33X	1000b	01b
3.5X	0101b	00b
4.5X	0111b	00b
2.33X	0101b	01b
4.0X	0110b	00b
2.66X	0110b	01b

Register Settings

Ratio = (MSR 0x147 [26:23] + 2) ÷ (MSR 0x10A[1:0] + 2)

The Machine-Specific Registers (MSRs) are not currently compatible with a P54C except for the Timestamp Counter (more on that later). When an IDT WinChip family CPU is recognized, writes to and reads from MSRs should be avoided (with the exception of MSR 10h, the Timestamp Counter). There are some performance-enhancing features that can and should be enabled. Write-combining, weak read ordering and linear burst ordering are such features (described later).

Although "MMX-compatible" instructions are supported, the string "MMX" should not be displayed with the processor name. Furthermore, the "-S" string that Intel uses to identify CPUs that support SMI should not be used, even though SMI is supported. The only identifiers that should be displayed are the processor name (e.g. "IDT WinChip 2") and a megahertz or PR-rating. This megahertz rating is the actual internal clock frequency while the PR rating is as given in the following table.

The actual CPU MHz should be displayed whenever the bus speed is below 83MHz. When the bus speed is either 83MHz or 100MHz, then the PR rating should be used. Note that the suffix "MHz" should not be used, as indicated in the tables below.

WinChip 2A PR Rating Table

Clock Speed MHz	PR Rating	Multiplier and Bus Frequency	Identifier
200	200	3x66MHz	IDT WinChip 2 – 200
200	233	2x100MHz	IDT WinChip 2 – 233
208	233	2.5x83MHz	IDT WinChip 2 – 233
233	233	3.5x66MHz	IDT WinChip 2 – 233
233	266	2.33x100MHz	IDT WinChip 2 – 266
250	266	3x83MHz	IDT WinChip 2 – 266
250	300	2.5x100MHz	IDT WinChip 2 – 300
266	266	4x66MHz	IDT WinChip 2 – 266
266	300	2.66x100MHz	IDT WinChip 2 – 300

WinChip 3 represents a significant improvement in performance over the WinChip 2. The next table shows the PR ratings of the WinChip 3. Again, please note that neither "PR" nor "MHz" appear in the identifier string.

WinChip 3 PR Rating Table

Clock Speed MHz	PR Rating	Multiplier and Bus Frequency	Identifier
200	233	3x66MHz	IDT WinChip 3 - 233
233	266	3.5x66MHz	IDT WinChip 3 - 266
233	300	2.33x100MHz	IDT WinChip 3 - 300
266	300	4x66MHz	IDT WinChip 3 - 233
266	333	2.66x100MHz	IDT WinChip 3 – 333
285	350	3x95MHz	IDT WinChip 3 - 350
300	333	4.5x66MHz	IDT WinChip 3 - 333
300	366	3x100MHz	IDT WinChip 3 - 366
316	380	3.33x95MHz	IDT WinChip 3 - 380
333	366	5x66MHz	IDT WinChip 3 – 366
333	400	3.33x100MHz	IDT WinChip 3 - 400

IDT WINCHIP RECOGNITION

The first step is to recognize the IDT WinChip and its revision level. Because each revision level can have unique features, both steps are necessary.

CPU recognition is accomplished through the use of the CPUID instruction. The CPUID instruction on IDT WinChip family CPUs is compatible with that of other x86-compatibles, but with added functionality. Refer to the Data Sheet for more detailed information on CPUID. We will summarize only the features we use in recognizing and working with the IDT WinChip Processor Family CPUs.

Operation of CPUID (Basic Functions):

Input value: eax: 0

Output values:

eax:	1
ebx:	0x746E6543
edx:	0x48727561
ecx:	0x736C7561

Note that this makes the vendor ID string "CentaurHauls"

Input value: eax: 1

Output values:

eax[3:0]	Stepping ID
eax[7:4]	Model
eax[11:8]	Family
eax[31:12]	Reserved
ebx	Reserved
ecx	Reserved
edx	Feature flags

The stepping ID, model, family and feature flags are defined compatibly. The IDT WinChip C6 returns an eax value of 0x54*z*, where *z* is the stepping number. The IDT WinChip 2 returns 0x58*z*. For the original WinChip 2, the value of z (the stepping ID) is 0x5 or 0xA and for the 100MHz version, the value of z (the stepping ID) is 7, 8 or 9. The WinChip 3 will return 0x59z.

Extended CPUID Functions

The extended functions are described in detail in the Data Sheet. We will describe only the one we use in detecting the CPU type and feature set.

The original IDT WinChip C6 supported one extended function.

Input value: eax: 0xC0000000

Output value:

eax: 0xC0000000

In other words, the input value must return unchanged. This indicates that only C6 features are supported.

The input value eax = 0x80000000 will also be returned unchanged for an IDT WinChip C6. This can be used instead of eax = 0xC0000000. For the IDT WinChip 2 and beyond, the input value eax = 0x80000000 produces the following output:

eax = 0x80000005 (highest extended cpuid function number)

ebx, ecx, edx reserved

The IDT WinChip 2 supports several extended functions in addition to the one described above. These should not be used unless it is previously determined that you are running on an IDT WinChip 2 per the above.

Input value: eax: 0x80000001

Pertinent output values:

eax[3:0]	Stepping ID
eax[7:4]	Model
eax[11:8]	Family
eax[31:12]	Reserved
edx[31]	1 if AMD-3D instructions supported
	0 if AMD-3D instructions not supported

This information can be used to select an identification

string to display at boot time. However, there is another architected means of determining the correct display string which might be simpler to implement, if that is all that is

desired. There is a set of three extended CPUID functions that returns the correct CPU Identifier String . They should be invoked sequentially to read the complete WinChip identifier.

Input values: eax: 0x80000002 – 0x80000004 (three values)

Output:

Returns the name of the processor, suitable for BIOS to display on the screen (ASCII). The string can be up to 48 characters in length. If the string is shorter, the rightmost characters are padded with zero. The leftmost characters go in eax, then ebx, ecx, and edx. The leftmost character goes in least significant byte (little endian).

For example, the string “IDT WinChip 2” would be returned by extended function eax = 0x80000002 as follows:

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.

CLOCK FREQUENCY DETERMINATION

The best algorithms for determining the clock frequency 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 of 2, 3, 4 and 5, 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.

DTLOCK

DTLOCK prevents the table walk state machine from performing a 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). This bit must be set to 1 for normal operation.

LINEAR BURST MODE

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 is enabled for the IDT WinChip 2 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.*

TRAIT MODE KEY

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 later versions of the processor family 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. 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.

In general system software can determine the processor version by comparing the Family and Model Identification fields returned by the CPUID standard function EAX=1.

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

The following table indicates how to interpret the results of both methods.

Family	Model	Trait Mode Key MCR_CTRL[19:17]	Processor Version
5	4	0	IDT WinChip C6
5	8	1	IDT WinChip 2
5	9	1	IDT WinChip 3

WRITE-COMBINING AND WEAK READ ORDERING

These are features that are described in detail in the Data Sheet. Briefly, when write-combining is enabled, the IDT WinChip 2 will attempt to accumulate memory writes of less than a full dword. This reduces the memory access overhead and improves performance. This feature should be enabled when the IDT WinChip 2 (or the WinChip C6) is recognized. 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.

The actual implementation of these features varies with the Feature Set Level, so it is imperative that this be checked 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, as required. Note also the use of the Trait Mode Key as described above.

Simple 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     11111111111110000000000000000000b

; 512K 640K
base_512K       equ     00000000000010000000000000000000b
mask_640K       equ     11111111111111100000000000000000b
; 1M - 2M
base_1          equ     00000000000100000000000000000000b
mask_2          equ     11111111111100000000000000000000b

; 2M - 4M
base_2          equ     00000000001000000000000000000000b
mask_4          equ     11111111111000000000000000000000b

; 4M - 8M
base_4          equ     00000000010000000000000000000000b
mask_8          equ     11111111110000000000000000000000b

 ; 8M - 16M
 base_8          equ     00000000100000000000000000000000b
```

```
mask_16         equ     11111111100000000000000000000000b
mask_12         equ     11111111110000000000000000000000b

; 16M - 32M
base_16         equ     00000001000000000000000000000000b
mask_32         equ     11111111000000000000000000000000b

; 32M - 64M
base_32         equ     00000010000000000000000000000000b
mask_64         equ     11111110000000000000000000000000b

; 64M - 128M
base_64         equ     00000100000000000000000000000000b
mask_128        equ     11111100000000000000000000000000b

; 128M - 256M
base_128        equ     00001000000000000000000000000000b
mask_256        equ     11111000000000000000000000000000b

; 256M - 512M
base_256        equ     00010000000000000000000000000000b
mask_512        equ     11110000000000000000000000000000b

; 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


MCRbaseno       equ     110h
MCR_CTRL        equ     120h

MCRvalue        equ     11111000000000000000001111b

BCRno           equ      145h
TLOCKbit        equ      3



;=======================================================
;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

 ; first assume C6
```

```
; see memory range descriptors for details

; bh has trait bits, bl = 0 if C6, 1 if WinChip 2
; traits=111b, bl=0
                mov     ebx, 11100000000b

; now check feature level to see if we understand what to do
; we have to skip the programming if we are dealing with the
; wrong levels
                mov     ecx, MCR_CTRL
                rdmsr
                and     eax, 111b shl 17   ; check [19:17]
                jz      sub_1M         ; if 0, it's C6
                cmp     eax, 1 shl 17  ; if 1, it's WC2
                jne     No_Memory_Features  ;skip if not 1


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

; traits=10001b, bl=1
                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 in the local trait bits
                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)

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


; esi now contains the offset

                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    ; if hole enabled

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, BCRno
                rdmsr
                or      ax, 1 shl TLOCKbit
                wrmsr

No_Memory_Features:
                popad
                ret

WinChip_Memory_Features  endp
```

SYSTEM SOFTWARE CONSIDERATIONS

Most of the time, the system BIOS will set up the memory control registers as suggested in the sample code above. 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.

MCR Control Register (MCR_CTRL) MSR 0x120

Bit	*Description*	*Default*	*Notes*
9	MCR 0 in use 0: MCR0[4:0] attributes is all zero 1: MCR0[4:0] attributes is non-zero	0	Read-Only (RO)
10	MCR 1 in use (see MCR 0)	0	RO
11	MCR 2 in use (see MCR 0)	0	RO
12	MCR 3 in use (see MCR 0)	0	RO
13	MCR 4 in use (see MCR 0)	0	RO
14	MCR 5 in use (see MCR 0)	0	RO
15	MCR 6 in use (see MCR 0)	0	RO
16	MCR 7 in use (see MCR 0)	0	RO

A zero in any position indicates that that MCR is unused. This means that a system programmer (for example, a video device driver writer) may enable memory traits for a special memory buffer. The traits are additive if there are overlaps in regions. This can cause unpredictable behavior in case of range overlaps. In debugging such problems the best thing to do is zero all of the MCRs first.