Application Note 112
Cyrix CPU Detection Guide

Applies to the full range of Cyrix CPUs from the Cx486SLC to the MII.

APPLICATION NOTE 112

Cyrix CPU Detection

1. Introduction

This application note explains how to use software to determine what type of CPU is present. This allows software vendors to create software applications and tools that can take advantage of the features in different processors.

As the Cyrix CPU architecture continues to expand with new processors, Cyrix provides different methods of software CPU identification. Three methods of Cyrix processor detection have evolved:

1) The processors are identified using standard features levels of the CPUID instruction.

2) The processors are identified using extended feature levels of the CPUID instruction.

3) The processors that do not recognize the CPUID instruction or have CPUID turned off as a default are identified by examination of the Device Identification Registers (DIR0 and DIR1).

Three tests may be performed. The CPU may be tested using the Support CPUID Test, Support Extended CPUID Test and the 5/2 Test. **The tests must be performed in a specific order**, otherwise the identification may be invalid.

Over the years, as the Cyrix CPU architecture has continued to expand, the following Cyrix processors may be encountered: Cx486SLC, Cx486DLC, Cx486SRx2, Cx486DRx2, Cx486S, Cx486DX, Cx486DX2, 5x86, 6x86, 6x86L, 6x86MX, MII, MediaGX, and GXm.

The Cx486SLC, Cx486DLC, Cx486SRx2, Cx486DRx2, Cx486S, Cx486DX, Cx486DX2, and 5x86 processors do not recognize the CPUID instruction and must be identified using the DIR0 and DIR1 register.

The 6x86, 6x86L and MediaGX processors do recognize the CPUID instruction, but these processors are initialized with CPUID turned off. With CPUID turned on, these processors support the standard level CPUID instruction.

The 6x86MX, MII and the GXm processors (and all future Cyrix CPUs) are initialized with CPUID turned on. The 6x86MX and MII processors support the standard-level CPUID instruction. The GXm processor and all future Cyrix CPUs supports the CPUID instructions at both the standard and extended-levels.

2. Cyrix CPU Identification and Inquiry Flow Chart

The Cyrix CPU Identification process (Figure 1 on page 3) consists of up to three tests and three inquiries. If CPUID is not supported, a 5/2 Test is performed to check if the CPU is a Cyrix part. If CPUID is supported, a second test is made to see if extended level CPUID is supported.

The numbers in parenthesis shown in Figure 1 refer to the supporting paragraphs in this manual.

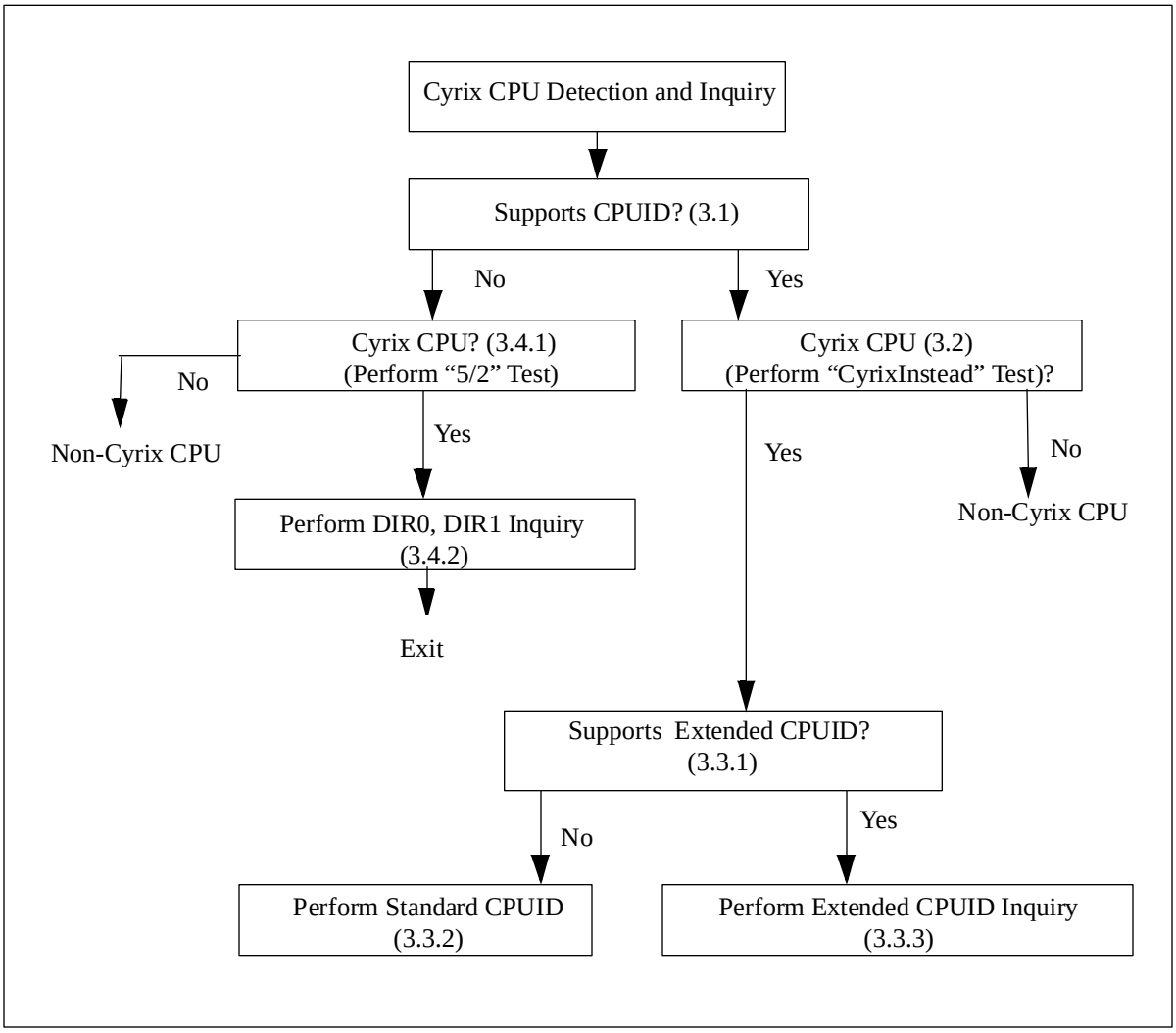

FIGURE 1. CPU Identification and Inquiry

Note: **The testing must be performed in the order shown in Figure 1 or testing may be invalid**.

3. CPU Detection Steps

3.1. CPUID Support Test

In order to avoid an invalid opcode exception on processors that do not support the CPUID instruction, software must first verify that the processor supports the CPUID instruction. The presence of the CPUID instruction is indicated by the ID bit (bit 21) in the EFLAGS register. If this bit can be toggled, the CPUID instruction is present and enabled on the processor. The following code will check for the presence of the CPUID instruction.

CPUID Support Test Sample Code*:

```
pushfd                  ; get extended flags
pop     eax             ; store extended flags in eax
mov     ebx, eax        ; save current flags
xor     eax, 200000h    ; toggle bit 21
push    eax             ; put new flags on stack
popfd                   ; flags updated now in flags
pushfd                  ; get extended flags
pop     eax             ; store extended flags in eax
xor     eax, ebx        ; if bit 21 r/w then eax <> 0
je      no_cpuid        ; can't toggle id bit (21) no cpuid here
```

*Note: It has been assumed that the tests for EFLAGS support has been complete prior to this point.

If CPUID is supported, it can be assumed that the CPU is an 80486 or above class processor.

3.2. *"CyrixInstead" Test*

The CPUID instruction level 0 provides vendor information. Following execution of the CPUID instruction with an input value of "0" in EAX, the EBX, ECX and EDX registers contain the vendor string of the CPU. To verify that the processor is a Cyrix CPU, the software checks for "CyrixInstead" in the return registers as shown in the sample code below:

"CyrixInstead" Test Sample Code

```
mov  eax, 0                ; CPUID standard level 0
cpuid
cmp  ebx, 'iryC'
jne  not_cyrix
cmp  edx, 'snIx'
jne  not_cyrix
cmp  ecx, 'daet'
jne  not_cyrix
```

3.3. *Standard and Extended CPUID Levels*

The CPUID instruction has been extended on recent processors so that additional information can be obtained from the CPU concerning items including stepping, model, family, type, TLB and cache information. The original levels of the CPUID instruction are termed "the standard CPUID levels" and the newer levels are termed the "extended CPUID levels."

The standard and extended CPUID levels differ in that the EAX register's most significant bit is set for the extended CPUID levels. Both the standard and extended CPUID levels may be executed at any privilege level. The EAX register provides the input value for the CPUID instruction to indicate what information should be returned by the instruction.

3.3.1 Extended CPUID Level Support Testing

This test is performed to determine if the CPU supports the extended CPUID levels.

Extended CPUID Instruction support testing consists of executing a CPUID instruction with the EAX register initialized to 8000 0000h and testing the return value in EAX. If a value greater than or equal to 8000 0000h is returned to the EAX register by the CPUID instruction, the CPU can execute extended CPUID instructions. The following sample code tests for Extended CPUID support.

Extended CPUID Instruction Test Sample Code:

```
mov     eax, 80000000h    ; try extended cpuid level
cpuid                     ; execute cpuid instruction
cmp     eax, 80000000h    ; check if extended levels are supported
jb      no_extended       ; extended cpuid functions not available
```

3.3.2 Standard CPUID Instruction Inquiry

The CPUID instruction provides processor and feature set information. This instruction may be executed at any privilege level. The standard CPUID instruction is defined as a CPUID instruction with the EAX register initialized to one of the following values:

0000 0000h - maximum standard levels supported and vendor string
0000 0001h - family, model and stepping information
0000 0002h - cache and TLB information

Table 1 summarizes the CPUID values returned by standard CPUID levels on Cyrix processors.

TABLE 1. SUMMARY OF RETURNED STANDARD CPUID VALUES

DESCRIPTION	INITIAL EAX VALUE	6x86*	6x86L*	MEDIAGX*	6x86MX	MII	GXm
Maximum Standard Value	0h	1h	1h	1h	1h	1h	2h
Stepping	1h	xx	xx	xx	xx	xx	xx
Model	1h	2h	2h	4h	0h	0h	4h
Family	1h	5h	5h	4h	6h	6h	5h
Type	1h	0h	0h	0h	0h	0h	0h
TLB/Cache	2h	-	-	-	-	-	xh**

Note: xx = stepping specific.

*Note: The CPUID instruction is disabled by default.

**Note: See Table 13 on page 17.

3.3.2.1 CPUID Instruction with EAX = 0000 0000h

Standard function 0h (EAX = 0) of the CPUID instruction returns the maximum standard CPUID levels supported by the current processor to the EAX register. The maximum standard CPUID level is the highest acceptable value for the EAX register input.

After the instruction is executed registers EBX through EDX contain the vendor string of the processor. Note that the middle section is placed (out of order) into the EDX register (Table 2).

TABLE 2. STANDARD CPUID WITH EAX = 0000 0000H

REGISTER*	CONTENTS
EAX	Max Standard Levels
EBX	Vendor ID String 1
EDX	Vendor ID String 2
ECX	Vendor ID String 3

*Note: The register order is correct.

3.3.2.2 CPUID Instruction with EAX = 0000 0001h

Standard function 01h (EAX = 1) of the CPUID instruction returns the Processor Type, Family, Model, and Stepping information of the current processor in EAX (Table 3). The Standard Feature Flags supported are returned in the EDX register. The other registers upon return are currently reserved.

TABLE 3. STANDARD CPUID WITH EAX = 0000 0001H

REGISTER	CONTENTS
EAX[3:0]	Stepping ID
EAX[7:4]	Model
EAX[11:8]	Family
EAX[15:12]	Type
EAX[31:16]	Reserved
EBX	Reserved
ECX	Reserved
EDX	Standard Feature Flags

Standard Feature Flags

The standard feature flags are returned in the EDX register when the CPUID instruction is called with standard function 01h (EAX = 1). Each flag refers to a specific feature and indicates if that feature is present on the processor. Some of these features require enabling or have protection control in CR4. Table 4 summarizes the standard feature flags.

Before using any of these features on the processor, the software should check the corresponding feature flag (Table 4). Attempting to execute an unavailable feature can cause exceptions and unexpected behavior. For example, software must check bit 4 before attempting to use the Time Stamp Counter instruction.

TABLE 4. STANDARD FEATURE FLAGS VALUES RETURNED IN EDX

FEATURE FLAG	EDX BIT	CR4 BIT	6x86*	6x86L*	MEDIAGX*	6x86MX	MII	GXM
FPU On-Chip	0	-	X	X	X	X	X	X
Virtual Mode Extensions (V86)	1	0,1	-	-	-	-	-	-
Debug Extension	2	3	-	X	-	X	X	-
4 MB Page Size	3	4	-	-	-	-	-	-
Time Stamp Counter	4	2	-	-	-	X	X	X
RDMSR/WRMSR Instructions	5	8	-	-	-	X	X	X
Physical Address Extensions	6	5	-	-	-	-	-	-
Machine Check Exception	7	6	-	-	-	-	-	-
CMPXCHG8B Instruction Support	8	-	-	X	-	X	X	X
On-chip APIC Hardware	9	-	-	-	-	-	-	-
Reserved	10	-	-	-	-	-	-	-
SYSENTER/SYSEXIT Instructions	11	-	-	-	-	-	-	-
Memory Type Range Registers (MTRR)	12	-	-	-	-	-	-	-
Page Global Enable (PTE-PGE)	13	7	-	-	-	X	X	-
Machine Check Architecture	14	-	-	-	-	-	-	-
Conditional Move Instruction (CMOV)	15	-	-	-	-	X	X	X
Page Attribute Table	16	-	-	-	-	-	-	-
36-Bit Page Size Extensions	17	-	-	-	-	-	-	-
Reserved	18-22	-	-	-	-	-	-	-
MMX™ Instructions	23	-	-	-	-	X	X	X
Fast FPU Save and Restore	24	-	-	-	-	-	-	-
Reserved	25-31	-	-		-	-	-	-

*Note: The CPUID instruction is disabled by default.

3.3.2.3 CPUID Instruction with EAX = 0000 0002h

Standard function 02h (EAX = 02h) of the CPUID instruction returns information that is specific to the Cyrix family of processors. Information about the TLB is returned in EAX. Information about the L1 Cache is returned in EDX. This information is to be looked up in a lookup table. See Table 13 on page 17.

TABLE 5. STANDARD CPUID WITH EAX = 0000 0002H

REGISTER	CONTENTS
EAX	TLB Information
EBX	Reserved
ECX	Reserved
EDX	L1 Cache Information

3.3.3 Extended CPUID Levels

The extended CPUID Instruction is defined when the EAX register is initialized to one of the following values (Table 6):

8000 0000h - Maximum Levels
8000 0001h - Processor Information/Extended features
8000 0002h - Processor Marketing Name
8000 0003h - Processor Marketing Name
8000 0004h - Processor Marketing Name
8000 0005h - TLB/Cache Information

Each of the extended CPUID levels reports information that is specific to the Cyrix family of processors.

TABLE 6. SUMMARY OF RETURNED EXTENDED CPUID VALUES

DESCRIPTION	INITIAL EAX VALUE	6x86*	6x86L*	MEDIA GX*	6x86MX	MII	GXM
Extended Levels	8000 0000h	-	-	-	-	-	8000 0005h
TLB Info	8000 0005h	-	-	-	-	-	0000 7001h
Cache Info	8000 0005h	-	-	-	-	-	0000 0080h

*Note: The CPUID instruction is disabled by default.

TABLE 7. SUMMARY OF CPUID FUNCTIONS

EXTENDED FUNCTION	DESCRIPTION	6x86	MEDIA GX	6x86MX	MII	GXM
8000 0000h	Extended Levels	-	-	-	-	X
8000 0001h	Extended Processor Info. Extended Feature Flags	-	-	-	-	X
8000 0002h	Processor Marketing Name	-	-	-	-	X
8000 0003h	Processor Marketing Name	-	-	-	-	X
8000 0004h	Processor Marketing Name	-	-	-	-	X
8000 0005h	TLB & Cache Information	-	-	-	-	X

3.3.3.1 CPUID Instruction with EAX = 8000 0000h

Extended function 8000 0000h (EAX = 8000 0000h) of the CPUID instruction returns the maximum extended CPUID levels supported by the current processor in EAX (Table 8). The other registers are currently reserved.

TABLE 8. MAXIMUM EXTENDED CPUID LEVEL

REGISTER	CONTENTS
EAX	Maximum Extended Levels
EBX	Reserved
ECX	Reserved
EDX	Reserved

3.3.3.2 CPUID Instruction with EAX = 8000 0001h

Extended function 8000 0001h (EAX = 8000 0001h) of the CPUID instruction returns the Processor Type, Family, Model, and Stepping information of the current processor in EAX (Table 9). The Extended Feature Flags supported are returned in EDX. The other registers are currently reserved.

TABLE 9. PROCESSOR SIGNATURE AND EXTENDED FEATURE FLAGS

REGISTER	CONTENTS
EAX[3:0]	Stepping ID
EAX[7:4]	Model
EAX[11:8]	Family
EAX[15:12]	Processor Type
EAX[31:16]	Reserved
EBX	Reserved
ECX	Reserved
EDX	Extended Feature Flags

Extended Feature Flags

The extended feature flags are returned in the EDX register when the CPUID instruction is called with extended function 8000 0001h (EAX = 8000 0001h). Each flag refers to a specific feature and indicates if that feature is present on the processor. Some of these features require enabling or have protection control in CR4. Table 10 summarizes the extended feature flags.

TABLE 10. EXTENDED FEATURE FLAGS

FEATURE FLAG	EDX BIT	CR4 BIT	CPUS PRIOR TO GXM	GXM
Floating Point Unit	0	-	-	X
Virtual Mode Extensions (V86)	1	0,1	-	-
Debug Extension	2	3	-	-
Page Size Extensions (4 MByte)	3	4	-	-
Time Stamp Counter	4	2	-	X
Cyrix Model-Specific Registers (MSR)	5	8	-	X
Reserved	6	-	-	-
Machine Check Exception	7	6	-	-
CMPXCHG8B Instruction	8	-	-	X
SYSCALL and SYSRET Instructions	11	-	-	-
Reserved	12	-	-	-
Global Paging Extension (PTE-PGE)	13	7	-	-
Reserved	14	-	-	-
Integer Conditional Move Instructions (CMOV)	15	-	-	X
Floating-Point Conditonal Move Instructions	16			
Reserved	17-22	-	-	-
MMX™ Instructions	23	-	-	X
Cyrix 6x86MX Multimedia Extensions	24	-	-	X
Reserved	25 - 30	-	-	-
3DNow!™ Instructions	31	-	-	-

3.3.3.3 CPUID Instruction with EAX = 8000 0002h - 8000 0004h

Extended functions 8000 0002h through 8000 0004h (EAX = 8000 0002h through EAX = 8000 0004h) of the CPUID instruction returns an ASCII string containing the name of the current processor (Table 11). These functions eliminate the need to look up the processor name in a lookup table. Software can simply call these functions to obtain the name of the processor. The string may be 48 ASCII characters long, and is returned in little endian format. If the name is shorter than 48 characters long, the remaining bytes will be filled with ASCII NUL character (00h).

TABLE 11. OFFICIAL CPU NAME

8000 0002H		8000 0003H		8000 0004H	
EAX	CPU Name 1	EAX	CPU Name 5	EAX	CPU Name 9
EBX	CPU Name 2	EBX	CPU Name 6	EBX	CPU Name 10
ECX	CPU Name 3	ECX	CPU Name 7	ECX	CPU Name 11
EDX	CPU Name 4	EDX	CPU Name 8	EDX	CPU Name 12

3.3.3.4 CPUID Instruction with EAX = 8000 0005h

Extended function 8000 0005h (EAX = 8000 0005h) of the CPUID instruction returns information about the TLB and L1 Cache to be looked up in a lookup table. Refer to Tables Figure 12 and Figure 13 shown below.

TABLE 12. CACHE AND TLB INFORMATION

REGISTER	CONTENTS
EAX	Reserved
EBX	TLB Information
ECX	L1 Cache Information
EDX	Reserved

TABLE 13. CACHE AND DESCRIPTOR LOOKUP TABLE

CPUID LEVEL	REGISTER	VALUE	COMMENTS
Standard	EAX	xx xx xx 01h	The CPUID instruction needs to be executed only once with an input value of 02h to retrieve complete information about the cache and TLB.
Extended	EBX		
Standard	EAX	xx xx 70 xxh	TLB is 32 Entry, 4-way set associative, and has 4 KByte Pages
Extended	EBX		
Standard	EDX	xx xx xx 80h	L1 cache is 16 KBytes, 4-way set associative, and has 16 bytes per line.
Extended	ECX		

3.4. Non-CPUID Testing and Inquiry

If the processor does not support CPUID, then the 5/2 test is performed to check if the CPU is a Cyrix processor. If it is a Cyrix processor, inquires must be made to the Device Identification Registers (DIR0 and DIR1) to determine which Cyrix processor is present.

3.4.1 5/2 Test

After making sure that the CPU does not support CPUID, the 5/2 Test is performed. This test will determine if the processor is a Cyrix device. The test consists of a simple division and status check of the Flags register. Software that does not first check for CPUID may report a Cyrix CPU when one is not present.

The 5/2 test checks the state of the undefined flags following execution of the divide instruction that performs 5 by 2 division. The undefined flags in a Cyrix processor remain unchanged following this particular divide operation. Other vendor's processors will modify some of the undefined flags.

5/2 Test Sample Code:

```
xor   ax, ax            ; clear ax
sahf                    ; clear flags, bit 1 is always 1 in flags
mov   ax, 5             ; move 5 into the dividend
mov   bx, 2             ; move 2 into the divisor
div   bl                ; do an operation that does not change flags
lahf                    ; get flags
cmp   ah, 2             ; check for change in flags
jne   not_cyrix         ; flags changed, not a Cyrix CPU
```

Once a Cyrix CPU is determined to be present, the software must use the Cyrix Device ID Registers (DIR0 and DIR1) to determine which CPU is present.

3.4.2 DIR0, DIR1 Inquiry

After determining that a Cyrix processor without CPUID exists, its Device ID Registers (DIR) can be read to identify the Cyrix processor type. The Device ID Registers are located using register indexes FEh and FFh. Access to these registers is achieved by writing the index of the register to I/O port 22h. I/O port 23h is then used for data transfer. Each port 23h data transfer must be preceded by a port 22h-register index selection; otherwise the second and later port 23h operations are directed off-chip and produce external I/O cycles.

The Tables 14 and 15 describe the bit definitions for the DIR0 and DIR1 Registers.

TABLE 14. DIR0 BIT DEFINITIONS

BIT POSITION	DESCRIPTION
7 - 4	CPU Device Identification Number (read only)
3 - 0	CPU Clock Multiplier (read only)

TABLE 15. DIR1 BIT DEFINITIONS

BIT POSITION	DESCRIPTION
7 - 4	CPU Step Identification Number (read only)
3 - 0	CPU Revision Identification (read only)

Table 16 describes the range of DIR0 values for the different generations of Cyrix CPUs.

TABLE 16. CPU GENERATION VALUES IN DIR0

DIR0 VALUES	DESCRIPTION
00h - 0h7	Cx486SLC Cx486DLC Cx486SRx2 Cx486DRx2
10h - 13h	Cx486S
1Ah - 1Fh	Cx486DX Cx486DX2
28h - 2Fh	5x86
30h - 35h	6x86 6x86L
40h - 4Xh	MediaGX
42h	GXm
50h - 5Fh	6x86MX
50h - 5Fh	MII

Tables 17 through 26 list individual Cyrix Processor DIR values.

TABLE 17. CX486SLC/DLC/SRX/DRX (M0.5)

DIR0	DIR1	DESCRIPTION
00h	Stepping	Cx486SLC
0h1		Cx486DLC
02h		Cx486SLC2
03h		Cx486DLC2
06h		Cx486SRx2 (Retail Upgrade CPU) 2x
07h		Cx486DRx2 (Retail Upgrade CPU) 2x

TABLE 18. CX486S (M0.6)

DIR0	DIR1	DESCRIPTION
10h	Stepping	Cx486S (B step)
11h		Cx486S2 (B step)
12h		Cx486Se (B step)
13h		Cx486S2e (B step)

TABLE 19. CX486DX/DX2 (M0.7)

DIR0	DIR1	DESCRIPTION
1Ah	Stepping	Cx486DX
1Bh		Cx486DX2
1Fh		Cx486DX4

TABLE 20. 5X86 (M0.9)

DIR0	DIR1	CORE CLOCK TO BUS CLOCK RATIO
28h	Stepping	1x
29h		2x
2Dh		3x
2Ch		4x

TABLE 21. 6X86 (M1)

DIR0	DIR1	CORE CLOCK TO BUS CLOCK RATIO
30h	Stepping	1x
31h		2x
35h		3x
34h		4x

TABLE 22. 6X86L (M1- LOW POWER)

DIR0	DIR1*	CORE CLOCK TO BUS CLOCK RATIO	DESCRIPTION
30h	22h - FFh	1 x	Supports CMPXCHG8B instruction and Debug Extension.
31h		2 x	
35h		3 x	
34h		4 x	

*Note: Lower nibble of DIR1 identifies CPU stepping.

TABLE 23. 6X86MX (M2)

DIR0	DIR1*	CORE CLOCK TO BUS CLOCK RATIO
50h	00 - 07h or 80h - 8Fh	1.0 x
51h		2.0 x
52h		2.5 x
53h		3.0 x
54h		3.5 x
55h		4.0 x
56h		4.5 x
57h		5.0 x
58h		1.0 x
59h		2.0 x
5Ah		2.5 x
5Bh		3.0 x
5Ch		3.5 x
5Dh		4.0 x
5Eh		4.5 x
5Fh		5.0 x

*Note: Lower nibble of DIR1 identifies CPU stepping.

TABLE 24. MII (M2)

DIR0	DIR1*	CORE CLOCK TO BUS CLOCK RATIO
52h	8h - 7Fh	2.5 x
53h		3.0 x
54h		3.5 x
55h		4.0 x
56h		4.5 x
57h		5.0 x
5Ah		2.5 x
5Bh		3.0 x
5Ch		3.5 x
5Dh		4.0 x
5Eh		4.5 x
5Fh		5.0 x

*Note: Lower nibble of DIR1 identifies CPU stepping.

TABLE 25. MEDIAGX (GX86)

DIR0	DIR1*	CORE CLOCK TO BUS CLOCK RATIO
41h	00h - 2Fh	3x
44h		4x
45h		3x
46h		4x
47h		3x

*Note: Lower nibble of DIR1 identifies CPU stepping.

TABLE 26 -- (GXM)

DIR0	DIR1*	CORE CLOCK TO PCI BUS CLOCK RATIO
40h	-	4x
41h	50h - FFh	10x
	30h - 3Fh or 40h - 4Fh	6x
42h	-	4x
43h	-	6x
44h	50h - FFh	9x
	30h - 3Fh or 40h - 4Fh	7x
45h 45h	30h - 3Fh or 40h - 4Fh	8x
	50h - FFh	5x
46h	-	7x
47h 47h	30h - 3Fh or 40h - 4Fh	5x
	50h - FFh	8x

*Note: Lower nibble of DIR1 identifies CPU stepping.

Printed in the United States of America

Order Number: none (web document)

Cyrix Corporation

2703 North Central Expressway

Richardson, Texas 75080-2010

United States of America

July 21, 1998 5:44 pm
C:\!!!devices\appnotes\112ap.fm5

Rev 1.9 Corrected Table 4, Table 14
Rev 1.8 Corrected code example on page 6 and updated Tables 3 and10
Rev 1.7 Revised DIR1 information on pages 21 and 22
Rev 1.6 Added MII, Updated GXm