Cyrix CPU Detection Guide

Preliminary Revision 1.01

Trademarks

Revision History

REVISION	RELEASE DATE	DESCRIPTION OF CHANGES
1.00	09/30/97	First Release (preliminary).
1.01	10/02/97	Corrected the GXm and MediaGX values for DIR0 lookups

Cyrix CPU Detection

Introduction

This document provides an overview of the three possible methods for detecting a Cyrix CPU. Once the correct method is identified (flowchart below), each detection method is covered in detail. This includes: How to detect the Cyrix CPU; Which CPU is present; What is the standard feature set; What are the Cyrix specific features.

The three CPU detection methods are:

1. **CPUID - Standard Levels**
   This method provides the standard feature set (not vendor specific feature such as Extended MMX) and requires a look-up table.

2. **CPUID - Extended Levels**
   This is the preferred method of detection because it provides the ability to get the CPU name without requiring a look-up table. It also provides information that may be Cyrix specific. The CPUID - Extended Levels are only supported in the most recent CPUs such as GXm.

3. **The 5/2 Method**
   This method is used for older CPUs that do not support CPUID such as the 486DLC, 486SLC, 486DX, 486DX2 etc.

The flowchart below should be used to determine the correct detection method. After identifying which method to use, refer to the correct section for further explanation.

```
                    CPUID Supported?
                           |
           No ------------------------------------------ Yes
            |                                              |
    ( Use 5/2 Method )                                Vendor =
                                                  "CyrixInstead"?
                                                           |
                                        Yes ------------------- No
                                         |                       |
                                      Extended          ( Other Vendor )
                                 CPUID Supported?
                                         |
                     Yes ----------------------------------- No
                      |                                       |
             ( Use Extended Method )              ( Use Standard Method )
```

CPUID - Standard Levels

Overview

The CPUID instruction is an application level (ring 3) instruction that provides information about the system's processor and its feature set. The CPUID instruction provides multiple functions, each containing different information about the processor. The CPUID instruction is used to identify the vendor, family, and type of processor, as well as information about any special features, like MMX™, that the processor may support. The CPUID instruction may be executed at any privilege level.

Testing for CPUID Support

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 sample code will check for the presence of the CPUID instruction. The following code should be executed after support for EFLAGS is verified, or at least a 80386/80486 is known to be present.

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
```

Standard CPUID Levels

Each of the standard CPUID levels (EAX = 0 and EAX = 1) contain the same information for all vendors. The higher CPUID levels, including the extended levels, report information that is specific to the Cyrix family of processors.

Table 1 summarizes the actual CPUID values currently returned by Cyrix processors.

Table 1. Actual CPUID Result Values:

Description	**6x86**	**6x86 (4.x)**	**MediaGX**	**6x86MX**	**GXm**
Standard Levels[1]	**1**	**1**	**1**	**1**	**2**
Stepping	**xx**	**xx**	**xx**	**0**	**0**
Model	**2**	**2**	**4**	**0**	**4**
Family	**5**	**5**	**4**	**6**	**5**
Type	**0**	**0**	**0**	**0**	**0**
Extended Levels[2]	**-**	**-**	**-**	**-**	**8000 0005h**
TLB Info[3]	**-**	**-**	**-**	**-**	**00 00 70 01h**
Cache Info[3]	**-**	**-**	**-**	**-**	**00 00 00 80h**

(EAX = 0h) - Vendor String and Max Standard CPUID Levels Supported

Standard function 0h (EAX = 0) of the CPUID instruction returns the maximum standard CPUID levels supported by the current processor in EAX. EBX through EDX return the vendor string of the processor. Please make note of the order of the registers.

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

[1] EAX Value when CPUID (EAX= 0) Executed
[2] Extended CPUID - EAX = 8000 0000h
[3] EAX = 2; See Table 5 for Value Definitions

(EAX = 01h) - Processor Signature and Standard Feature Flags

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

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

(EAX = 02h) - TLB and L1 Cache Information

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 4)

EAX	TLB Information
EBX	Reserved
ECX	Reserved
EDX	L1 Cache Information

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 2 summarizes the standard feature flags.

Before using any of these features on the processor, the software should check the corresponding feature flag. 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. See the glossary for a definition of each feature.

Table 2 - Standard Feature Flags Values:

Feature Flag	EDX Bit	CR4 Bit	6x86*	6x86* (4.x)	MediaGX*	6x86MX	GXm
FPU	**0**	**-**	**X**	**X**	**X**	**X**	**X**
V86	**1**	**0,1**	**-**	**-**	**-**	**-**	**-**
Debug Extension	**2**	**3**	**-**	**X**	**-**	**X**	**-**
4MB Page Size	**3**	**4**	**-**	**-**	**-**	**X**	**-**
Time Stamp Counter	**4**	**2**	**-**	**-**	**-**	**X**	**X**
RDMSR / WRMSR	**5**	**8**	**-**	**-**	**-**	**X**	**X**
PAE	**6**	**5**	**-**	**-**	**-**	**-**	**-**
MC Exception	**7**	**6**	**-**	**-**	**-**	**-**	**-**
CMPXCHG8B	**8**	**-**	**-**	**X**	**-**	**X**	**X**
APIC on Chip	**9**	**-**	**-**	**-**	**-**	**-**	**-**
Reserved	**10-11**	**-**	**-**	**-**	**-**	**-**	**-**
MTRR	**12**	**-**	**-**	**-**	**-**	**-**	**-**
Global Bit	**13**	**7**	**-**	**-**	**-**	**X**	**-**
Machine Check	**14**	**-**	**-**	**-**	**-**	**-**	**-**
CMOV	**15**	**-**	**-**	**-**	**-**	**X**	**X**
Reserved	**16-22**	**-**	**-**	**-**	**-**	**-**	**-**
MMX	**23**	**-**	**-**	**-**	**-**	**X**	**X**

Extended CPUID Levels

Overview

The extended CPUID levels are provided to simplify the detection routines used by developers as well as provide information on processor specific extensions. Like the CPUID instruction, the extended CPUID levels may be executed at any privilege level.

Testing for Extended CPUID Support

Before executing CPUID at the extended levels, 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.

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
```

To verify that the processor supports the extended CPUID levels, software checks for "CyrixInstead" in the vendor string returned by CPUID level 0, and a value greater than or equal to 8000 0000h in the EAX register returned by CPUID level 8000 0000h.

Sample Code:

```
        mov     eax, 8000000    ; try extended cpuid level
        cpuid                   ; execute cpuid instruction
        cmp     eax, 8000000    ; check if extended levels are supported
        jl      no_extended     ; extended cpuid functions not available
```

Among the processors in the Cyrix family, different ones may execute different levels of CPUID. Table 3 summarizes the CPUID levels currently implemented on Cyrix processors.

Table 3. Summary of CPUID Functions:

Standard Functions	Extended Functions	Description	5x86 & Prior	6x86*	Media GX*	6x86MX	GXm
0	-	Standard Levels Vendor String	-	X	X	X	X
1	-	Processor Information Standard Feature Flags	-	X	X	X	X
2	-	TLB & Cache Information	-	-	-	-	X
-	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

Extended CPUID Levels

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

(EAX = 8000 0000h) - Maximum Extended CPUID Levels Supported

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

EAX	Max Extended Levels
EBX	Reserved
ECX	Reserved
EDX	Reserved

(EAX = 8000 0001h) - Processor Signature and Extended Feature Flags

Extended function 8000 0001h (EAX = 8000 0001h) of the CPUID instruction returns the Processor Type, Family, Model, and Stepping information of the current

[*] CPUID is turned off by default on this CPU and most BIOS

processor in EAX. The Extended Feature Flags supported are returned in EDX. The other registers upon return are currently reserved. The breakdown of the EAX register is as follows:

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

(EAX = 8000 0002h - 8000 0004h) - Official CPU Name

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. 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 string. 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).

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

(EAX = 8000 0005h) - TLB and L1 Cache Information

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.

EAX	Reserved
EBX	TLB Information
ECX	L1 Cache Information
EDX	Reserved

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 4 summarizes the extended feature flags. See the glossary for a definition of each feature.

Table 4 - Extended Feature Flags Values:

Feature Flag	EDX Bit	CR4 Bit	CPUs Prior to GXm	GXm
FPU	**0**	**-**	**-**	**X**
V86	**1**	**0,1**	**-**	**-**
Debug Extension	**2**	**3**	**-**	**-**
Page Size Extensions	**3**	**4**	**-**	**-**
Time Stamp Counter	**4**	**2**	**-**	**X**
Cyrix MSR	**5**	**8**	**-**	**X**
PAE	**6**	**5**	**-**	**-**
MC Exception	**7**	**6**	**-**	**-**
CMPXCHG8B	**8**	**-**	**-**	**X**
APIC on Chip	**9**	**-**	**-**	**-**
SYSCALL / SYSRET	**10**	**-**	**-**	**-**
Reserved	**11**	**-**	**-**	**-**
MTRR	**12**	**-**	**-**	**-**
Global Bit	**13**	**7**	**-**	**-**
Machine Check	**14**	**-**	**-**	**-**
CMOV	**15**	**-**	**-**	**X**
FPU CMOV	**16**	**-**	**-**	**X**
Reserved	**17-22**			**-**
MMX	**23**			**X**
Extended MMX	**24**			**X**

Table 5 - Cache and TLB Descriptor Lookup Table

Value	Name	Size	Associative	Comments
01h	If the least-significant byte (byte 0) is set to 01h, this indicates that the CPUID instruction needs to be executed only once with an input value of 2 to retrieve complete information about the processor's caches and TLBs.			
70h	TLB	32 Entry	4 Way	4K-Byte Pages
80h	Level 1 Cache	16K	4 Way	16 Bytes/Line

Cyrix CPU Detection - "The 5/2 Method"

Overview

Each of the CPU vendors has created a unique way of determining which CPU is in a user's machine. A developer may detect if a Cyrix CPU is present by using a simple division routine then checking the status of the Flags register. This method is only valid after determining that the CPU does not support CPUID. Software that does not first check for CPUID may report a Cyrix CPU when one is not present.

Once a Cyrix CPU is determined to be present, the software may use the "Cyrix Device ID Registers" and a lookup table to determine which CPU is present.

Detecting a Cyrix CPU

A software check is required to determine if the CPU is an 80486 or above class processor, since all Cyrix CPUs are 80486 and above class processors. If this check is not made, the software may report a Cyrix CPU when one is not present. The following code is an example of how to detect an 80486 and above class processor.

Sample Code:

```
        pushfd                  ; save EFLAGS
        pop     eax             ; get EFLAGS
        mov     ecx, eax        ; temp storage EFLAGS
        xor     eax, 40000h     ; change AC bit in EFLAGS
        push    eax             ; put new EFLAGS value on stack
        popfd                   ; replace current EFLAGS value
        pushfd                  ; get EFLAGS
        pop     eax             ; save new EFLAGS in EAX
        cmp     eax, ecx        ; compare temp and new EFLAGS
        jz      is_not_80486    ; not a 486 or above class processor
```

The division routine is used after it has been determined that the processor is a 80486 or above class processor. Detection of a Cyrix CPU is accomplished by checking the state of the undefined flags following execution of the divide instruction that divides 5 by 2 (5÷2). The undefined flags in a Cyrix processor remain unchanged following the divide operation. Other vendor's processors will modify some of the undefined flags.

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
```

Determining which Cyrix CPU is present

After determining that a Cyrix processor exists, its Device ID Registers can be read to identify its type. The Device ID Registers exist at 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 22hregister index selection, otherwise the second and later port 23h operations are directed off-chip and produce external I/O cycles.

The following is a table describing the bit definitions of each Device ID Register:

DIR0 Bit Definitions

Bit Position	Description
7-0	CPU Device Identification Number (read only)

DIR1 Bit Definitions

Bit Position	Description
7-4	CPU Step Identification Number (read only)
3-0	CPU Revision Identification (read only)

The following is a table describing the base level DIR0 values for the different generations of Cyrix CPUs. A more detailed table of each generation of Cyrix CPU is located in Appendix A.

DIR1 Values	Description
00h - 07h	Cx486SLC/DLC/SRx/DRx
10h - 13h	Cx486S
1Ah - 1Fh	Cx486DX/DX2
28h - 2Fh	5x86
30h - 35h	6x86 / 6x86L
50h - 5Fh	6x86MX
40h - 4xh	MediaGX
42h	GXm

GLOSSARY:

Feature Flags Descriptions:

FPU	A Floating-point unit is onboard the CPU.
V86	Virtual mode extensions are available.
Debug	I/O Breakpoint debug extensions are supported.
Page Size	4-Mbyte pages are supported.
Time Stamp	A time stamp counter is available, and the RDTSC instruction is available.
MSRs	Cyrix model-specific registers are available, and the RDMSR and WRMSR instructions are supported.
PAE	Physical Address Extensions. (Need more info...)
MCExt	Machine Check Exception is supported.
CMPXCHG8B	Compare Exchange Eight Byte instruction is supported.
APIC	A local APIC unit is available.
MTRR	Memory Type Range Register (Need more info...)
Global Paging	Global paging extensions are available.
Machine Check	Machine Check Arch. (Need more info...)
Cond. Move	The conditional move instructions CMOV, FCMOV, and FCOMI are supported.
MMX	MMX™ instruction set is supported.
SYSCALL/RET	SYSCALL and SYSRET instructions and associated extensions are supported.
FPU CMOVs	Floating-point conditional move instructions FCMOV and FCOMI are supported.

Appendix A:

Tables of DIR values for each Cyrix Processor

Cx486SLC/DLC/SRx/DRx (M0.5)

DIR0	DIR1	Description
00h	Stepping	Cx486_SLC
01h	Stepping	Cx486_DLC
02h	Stepping	Cx486_SLC2
03h	Stepping	Cx486_DLC2
04h	Stepping	Cx486SRx (Retail Upgrade CPU)
05h	Stepping	Cx486DRx (Retail Upgrade CPU)
06h	Stepping	2x Cx486SRx2 (Retail Upgrade CPU)
07h	Stepping	2x Cx486DRx2 (Retail Upgrade CPU)

Cx486S (M0.6)

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

Cx486DX/DX2 (M0.7)

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

5x86 (M0.9)

DIR0	DIR1	Description
28h	Stepping	1x Clock (Core/Bus)
2Ah	Stepping	1x Clock (Core/Bus)
29h	Stepping	2x Clock (Core/Bus)
2Bh	Stepping	2x Clock (Core/Bus)
2Dh	Stepping	3x Clock (Core/Bus)
2Fh	Stepping	3x Clock (Core/Bus)
2Ch	Stepping	4x Clock (Core/Bus)
2Eh	Stepping	4x Clock (Core/Bus)

6x86 (M1)

DIR0	DIR1	Description
30h	Stepping	1x Clock (Core/Bus)
31h	Stepping	2x Clock (Core/Bus)
35h	Stepping	3x Clock (Core/Bus)
34h	Stepping	4x Clock (Core/Bus)

6x86L (M1)

DIR0	DIR1	Description
30h	> 21h	1x Clock (Core/Bus) - Supports CMPEX8B, Debug Ext.
31h	> 21h	2x Clock (Core/Bus) - Supports CMPEX8B, Debug Ext.
35h	> 21h	3x Clock (Core/Bus) - Supports CMPEX8B, Debug Ext.
34h	> 21h	4x Clock (Core/Bus) - Supports CMPEX8B, Debug Ext.

6x86MX (M2)

DIR0	DIR1	Description
50h	Stepping	1x Clock (Core/Bus)
58h	Stepping	1x Clock (Core/Bus)
51h	Stepping	2x Clock (Core/Bus)
59h	Stepping	2x Clock (Core/Bus)
52h	Stepping	2.5x Clock (Core/Bus)
5Ah	Stepping	2.5x Clock (Core/Bus)
53h	Stepping	3x Clock (Core/Bus)
5Bh	Stepping	3x Clock (Core/Bus)
54h	Stepping	3.5x Clock (Core/Bus)

5Ch	Stepping	3.5x Clock (Core/Bus)
55h	Stepping	4x Clock (Core/Bus)
5Dh	Stepping	4x Clock (Core/Bus)
56h	Stepping	4.5x Clock (Core/Bus)
5Eh	Stepping	4.5x Clock (Core/Bus)
57h	Stepping	5x Clock (Core/Bus)
5Fh	Stepping	5x Clock (Core/Bus)

MediaGX (Gx86)

DIR0	DIR1	Description
41h	Stepping	3x Clock (Core/Bus)
45h	Stepping	3x Clock (Core/Bus)
47h	Stepping	3x Clock (Core/Bus)
44h	Stepping	4x Clock (Core/Bus)
46h	Stepping	4x Clock (Core/Bus)

(GXm)

DIR0	DIR1	Description
40h	Stepping	4x Clock (Core/Bus)
42h	Stepping	4x Clock (Core/Bus)
47h	Stepping	5x Clock (Core/Bus)
41h	Stepping	6x Clock (Core/Bus)
43h	Stepping	6x Clock (Core/Bus)
44h	Stepping	7x Clock (Core/Bus)
46h	Stepping	7x Clock (Core/Bus)
45h	Stepping	8x Clock (Core/Bus)