WEITEK

4167 FLOATING-POINT COPROCESSOR

ADVANCED DATA

July 1989

The WEITEK 4167, the newest member of the WEITEK Abacus family, is a high-performance single-chip floating-point coprocessor for Intel's 80486 microprocessor. It is upwardly binary compatible with the WEITEK 3167 coprocessor. Fully supported by a wide selection of application packages and by high-level language compilers, under DOS, UNIX System V.3, and XENIX Release 3.2, the 4167 provides a superior floating-point accelerator for high-end PCs, workstations, graphics, and numeric controllers.

Contents

4167 Floating-Point Coprocessor
July, 1989

WEITEK Corporation
1060 East Arques Avenue
Sunnyvale, California 94086
Telephone (408) 738-8400

Printed in the United States of America
90 89 6 5 4 3 2 1

DOC 8943

WTL 4167 FLOATING-POINT COPROCESSOR

ADVANCE DATA
July 1989

Features

SINGLE-CHIP FLOATING-POINT COPROCESSOR

Designed for use with the Intel 80486

Fits a standard 142-pin PGA socket

Upward object-code-compatible from WEITEK 1167 and 3167 coprocessors for Intel 80386

HIGH PERFORMANCE

17.0 single-precision mega-whetstones and 3.8 single-precision Linpack MFLOPS

HIGH-LEVEL LANGUAGES

Supported by C, FORTRAN, and Pascal compilers under UNIX System V.3, XENIX Release 3.2, and MS-DOS real and protected mode

IEEE FORMAT

Conforms to the IEEE Standard Format for Floating-Point Arithmetic in both single- and double-precision (ANSI/IEEE Standard 754-1985)

FULL FUNCTION

Add, subtract, multiply, divide, and square root

Integer-floating-point conversions

Absolute value

Compare

Transcendental functions supported by run-time libraries

Low power CMOS

Dissipates 2.5 Watts max at 25 MHz

142-pin PGA package

Description

The 4167 is a high-performance single-chip floating-point coprocessor for Intel's 80486 32-bit microprocessor. It delivers 2 to 3 times the system performance of the 80486's on-chip math coprocessor. (Benchmark results are given in figure 1).

The interface signals between the 4167 and the 80486 are provided by a 142-pin socket.

The 4167 is upward object-code-compatible with the 1167 coprocessor daughter board and 3167 coprocessor chip. All of the applications ported to the 3167 will run as is on the 4167. FORTRAN, C, and Pascal compilers fully support the 4167, allowing new applications to be easily recompiled to take advantage of the WEITEK coprocessor.

Benchmark	Performance
Linpack (SP)	3.8 MFLOPS
Linpack (DP)	2.4 MFLOPS
Whetstone (SP)	17.0 MWhetstones
Whetstone (DP)	10.5 MWhetstones
Estimates	

Figure 1. Benchmark results at 25 MHz

Description, continued

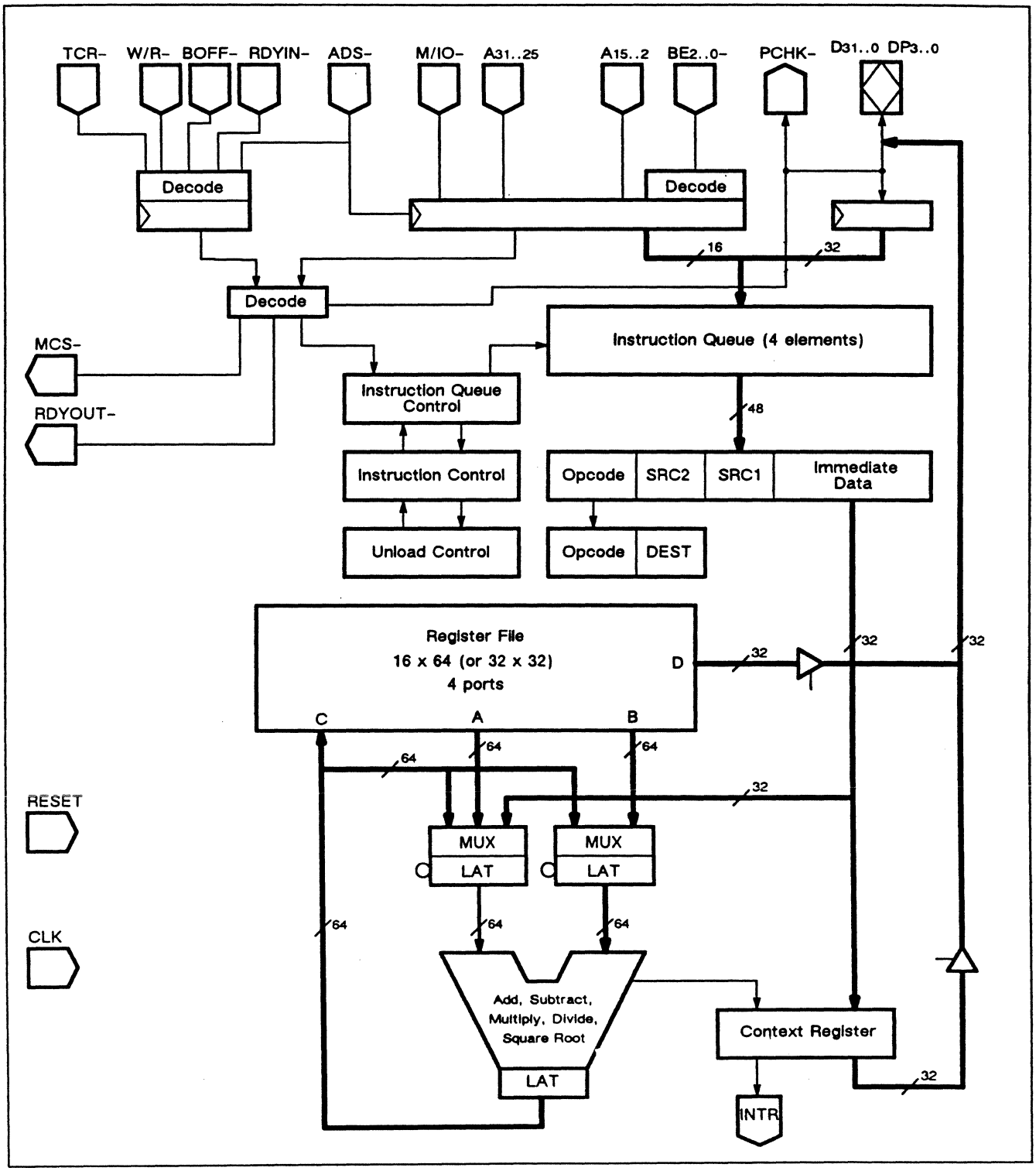


Figure 2. 4167 simplified block diagram

Description, continued

SUPPORTING THE 4167

Several key suppliers are involved in providing complete end-user solutions that support the WEITEK coprocessor (See figure 3).

Manufacturers developing systems based on the Intel 80486 must design the 142-pin 4167 coprocessor socket into their motherboards. They will offer a coprocessor detection mechanism by modifying their ROM BIOS.

Systems software developers, including operating system, DOS extender and Extended Memory Manager developers, need to enable the addressing of the WEITEK coprocessor and, in the case of multitasking environments, they also have to handle coprocessor context switching.

Compiler manufacturers will support coprocessor presence detection, initialization, exception handling, and code generation.

Finally, application vendors will recompile their applications to take advantage of the WEITEK coprocessor.

The present document is addressed to hardware manufacturers. It consists of two sections: The Hardware Designer's Guide and the Programming Interface Overview..

The Hardware Designer's Guide provides all the information necessary to design the 4167 socket into a 80486-based motherboard. Also, it explains how to modify the ROM BIOS to provide coprocessor presence detection.

The Programming Interface Overview offers a brief overview of the coprocessor programming model and of the software tools already supporting the 4167.

Operating systems, compilers and application programmers that intend to port new software tools or applications to the 4167 should refer to the *Abacus Software Programmer's Guide.*

Supplier	Support	Documentation
Application vendor	Recompile application to generate WEITEK coprocessor binaries	1167 Software Designer's Guide
Compiler manufacturer	WEITEK code generation, presence detection, initialization and exception handling	1167 Software Designer's Guide
Operating system developer (E.g. UNIX, DOS protected mode environment, extended memory manager developer)	Coprocessor addressing and context-switch handling	1167 Software Designer's Guide
Hardware manufacturer	142-pin socket and ROM BIOS support for presence detection	4167 Data Sheet

Figure 3. Layers of WEITEK coprocessor support

Hardware Designer's Guide

This section provides the electrical and mechanical information necessary to design the 4167 socket into an 80486 system. It also explains how to modify the ROM BIOS to support coprocessor presence detection.

The 4167 coprocessor is a memory-mapped peripheral. From the system designers standpoint, integrating the 4167 into the system is as simple as adding additional memory at an upper address. To the 80486 and its application software the 4167 appears to be a segment of memory. All the signals necessary to interface the 4167 to the 80486 are provided by a standard 142-pin grid array socket.

Figure 4 shows the 4167 socket pin-out and size. Figure 5 shows the 4167 physical dimensions.

Pin A1 Identifier	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A			VCC	GND	GND	VCC	GND	GND	VCC	GND	GND	VCC	GND	DP1	D7
B	D22	D23	D21	D19	GND	D17	DP2	GND	D14	D12	GND	D10	D8	VCC	GND
C	DP3	GND	VCC	D20	VCC	D18	D16	VCC	D15	D13	VCC	D11	D9	D6	D5
D	D24	D25	D26										VCC	GND	GND
E	D27	D28	D29										D4	D3	VCC
F	D30	GND	VCC										D2	D1	GND
G	D31	GND	VCC										VCC	GND	GND
H	GND	GND	VCC				15×15 142-PIN PGA TOP VIEW						D0	DP0	VCC
J	RESET	PCHK-	CLK										NC	NC	GND
K	BOFF-	NC	VCC										VCC	GND	GND
L	INTR	RDY IN-	GND										NC	NC	NC
M	RDY OUT-	GND	VCC										VCC	GND	NC
N	W/R-	TCR-	MCS-	A28	VCC	A26	A14	VCC	A11	A8	VCC	A5	A2	BE1-	PRES-
P	NC	M/IO-	A30	A27	GND	A25	A13	GND	A10	A7	GND	A4	BE2-	BE0=	NC
R	ADS-	A31	A29	NC	GND	A15	A12	GND	A9	A6	GND	A3	NC	NC	GND

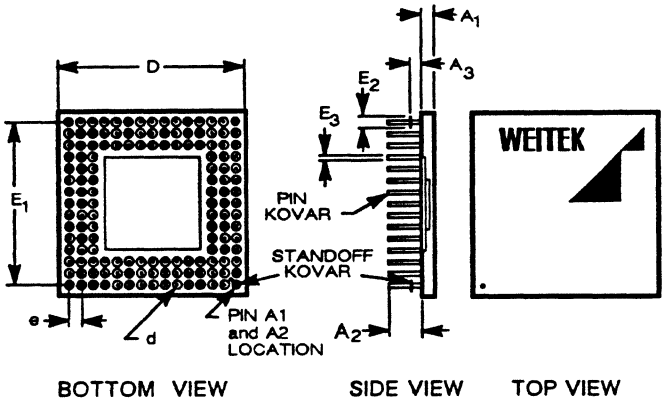
Note: NC = not connected; pins so marked must be left unconnected.
There is no pin at A1 or A2. A1 and A2 are locator holes.

Recommended socket: Augat PPS142-1A1525-L

Figure 4. 4167 socket pinout

Hardware Designer's Guide, continued

142-PIN GRID ARRAY



Symbol	DIMENSIONS	
	INCHES	MM
A1	0.100 ± 0.010	2.54 ± .20
A2	0.180 typ. ± 0.005	4.57 typ. ± .13
A3*	0.050 typ. ± 0.005	1.27 typ. ± .13
D	1.575 sq. ± 0.016	40.0 ± 0.41
E1	1.400 sq. ± 0.012	35.56 ± 0.30
E2	0.050 dia. typ.	1.27 dia. typ.
E3	0.018 ± 0.002	.46 ± 0.05
d	0.065 dia. typ.	1.65 dia. typ.
e	0.100 typ.	2.54 typ.

*A3 is from edge of case to bottom of stand-off.

Figure 5. 4167 physical dimensions

CONNECTING THE COPROCESSOR SOCKET

The following paragraphs describe the connection of each signal of the 4167.

CLK

CLK is the clock input to the 4167. All 4167 timing is relative to CLK. This signal must be the same as CLK of the 80486, but the 4167 should have a dedicated trace.

V_{DD}

Five volt (+5.0 V) power supply for the 4167. All V_{DD} pins must be connected.

GND

Ground for the 4167. All ground pins must be connected.

ADDRESS BUS ($A_{31..25}$, $A_{15..2}$ and $BE_{2..0}$–).

Pins $A_{31..25}$, $A_{15..2}$ and $BE_{2..0}$– should be connected directly to the corresponding 80486 address bus and byte enables respectively.

DATA BUS ($D_{31..0}$ and $DP_{3..0}$).

Pins $D_{31..0}$ and $DP_{3..0}$ should be connected to the 80486 data bus.

ADS–

Address status input signal connects to ADS# on the 80486.

M/IO–

Memory/IO status input signal connects to M/IO# on the 80486.

W/R–

Write/Read status input signal connects to W/R# on the 80486.

BOFF–

Backoff status input signal connects to BOFF# on the 80486.

PCHK–

The parity check output signal reports the results of checking even parity after a write to the coprocessor. The signal is driven during the cycle following the acknowledgement of a write operation. It is driven high (deasserted) during all other cycles. Other than causing the PCHK– signal to be driven low, the parity error has no effect on the 4167 operation.

RESET

System reset input signal connects to RESET on the 80486.

RDYIN–

Bus transfer complete input signal connects to RDY# on the 80486.

RDYOUT–

The RDYOUT– output signal must be "ORed" into the logic generating RDY# for the 80486, see figure 6.

INTR

The INTR output of the 4167 must be connected to the system interrupt controller. In the world of AT-compatible systems, for example, the 4167 INTR should be connected to IRQ13.

PRES–

PRES– signals the presence of a 4167 coprocessor. This signal should be connected to Vcc through a resistor of at least 10KOhm to insure a high level when the 4167 coprocessor is not present.

The basic software method of detecting the presence of a 4167 in an 80486 system is to perform a functional test of the device by attempting to load data into the coprocessor register file and read it back (a coded example is provided in figure 20 on page 17).

The hardware designer can use the PRES– output to make sure that the system acknowledges a bus transfer to an absent 4167, in order to avoid system hangs. The PRES– could also be connected to an I/O port to simplify the presence detection program residing in ROM BIOS (Refer to *Modifying The Rom Bios* paragraph on page 16 for more details).

MCS–

The MCS– signal is asserted whenever a transfer to or from the 4167 is pending on the bus. It can be used by the board designer to turn off other bus drivers during read operations from the 4167. A 20KOhm pull-up resistor can be used to keep MCS– deasserted when the coprocessor is not present.

Hardware Designer's Guide, continued

MCS- should also be combined into the KEN# generation logic on the system motherboard to drive KEN# deasserted (refer to the following TCR- paragraph for details).

TCR-

When asserted, the TCR-, *Three Cycle Read* signal, forces any read operation from the 4167 to require a minimum of one-wait state. This signal is sampled in the same cycle in which ADS- is asserted.

The hardware designer can choose either of two schemes to connect the TCR- signal. Assuming the data being read is available, the first scheme (Alternative A) will lead to zero wait state reads from the 4167 on new, optimized code, and to one wait state reads on existing application code. The second scheme (Alternative B) will always lead to zero wait state reads (assuming that data being read is available) independent of the code being executed. Alternative B will lead to a 5-10% performance improvement when running existing numerically intensive code.

Alternative A: The TCR- input signal of the 4167 is connected to the 80486 PCD output signal. The 4167 MCS- output signal is combined into the KEN# generation logic on the system motherboard to drive KEN# deasserted when a 4167 read is performed. Alternative A implementation is shown in figure 6.

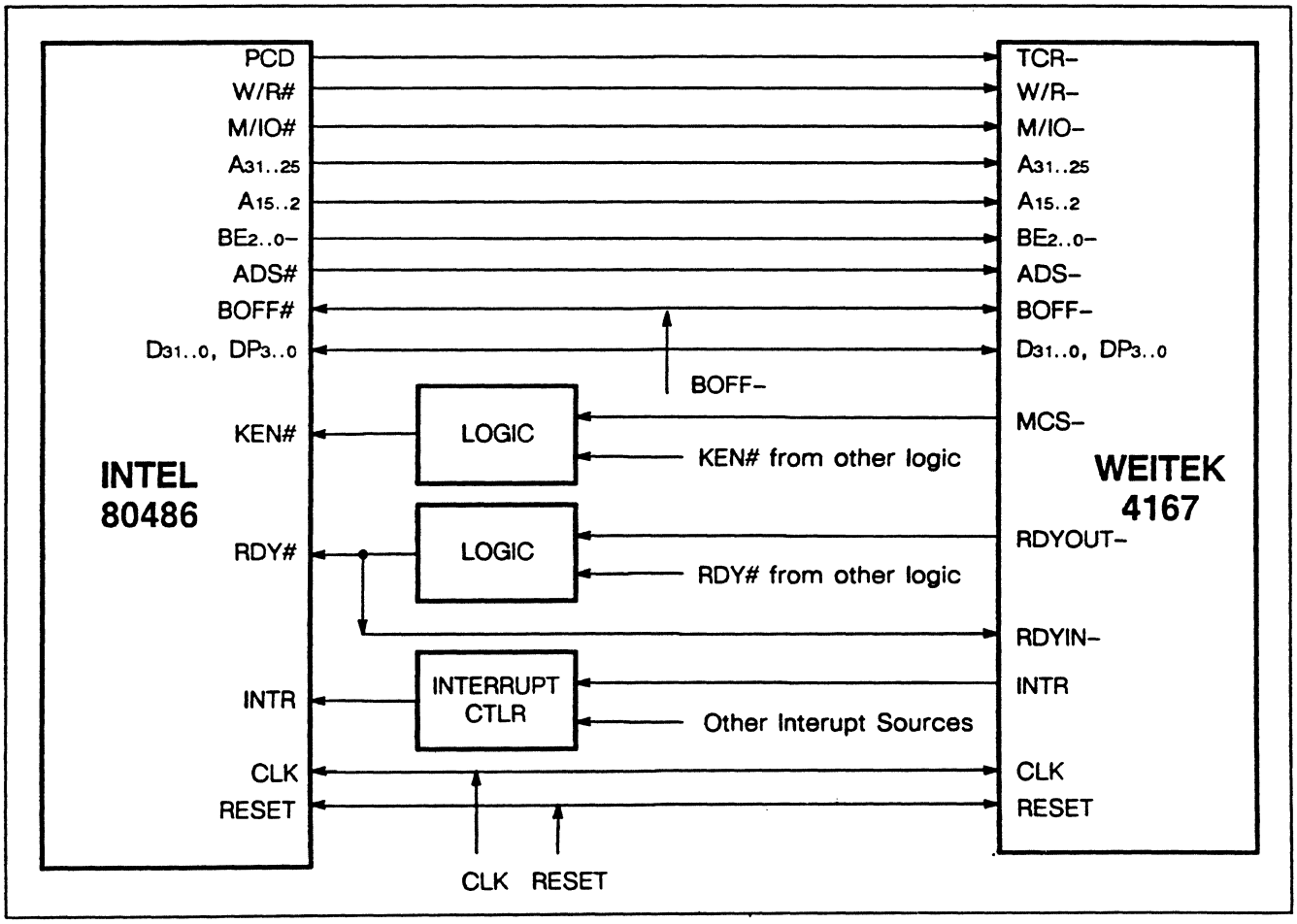


Figure 6. System connection diagram (Alternative A)

Alternative B: The TCR- input signal of the 4167 is connected through a 20KOhm pull-up resistor to the 5V power supply. Fast external logic must then be used to decode the $A_{31..25}$, ADS-, M/IO-, and W/R- signals in order to deassert KEN# on the first cycle of a 4167 bus operation. MCS- is used to keep KEN# deasserted on subsequent cycles. Figure 7 shows the implementation of Alternative B.

SYSTEM-LEVEL CONSIDERATIONS

The 4167 coprocessor is a memory-mapped peripheral that communicates with the 80486 over the same address bus that connects the main memory to the CPU. Instructions are defined by the 14 least-significant address bits ($A_{15..2}$) as well as three of the four byte enables ($BE_{2..0}$-).

The seven most significant bits of the 80486 address bus ($A_{31..25}$), together with the Memory I/O control Signal (M/IO-), select the 4167 coprocessor. Only the upper seven address bits are decoded to determine when a coprocessor operation is being requested.

The coprocessor will respond to memory addresses C0000000 through C1FFFFFFH. Although by convention only addresses C0000000 to C000FFFFH are used, it is important to be sure that other components in the system do not conflict with the address space decoded by the coprocessor. Writing to this address space will cause the 4167 to execute instructions and reading from it will cause the coprocessor to drive the data bus.

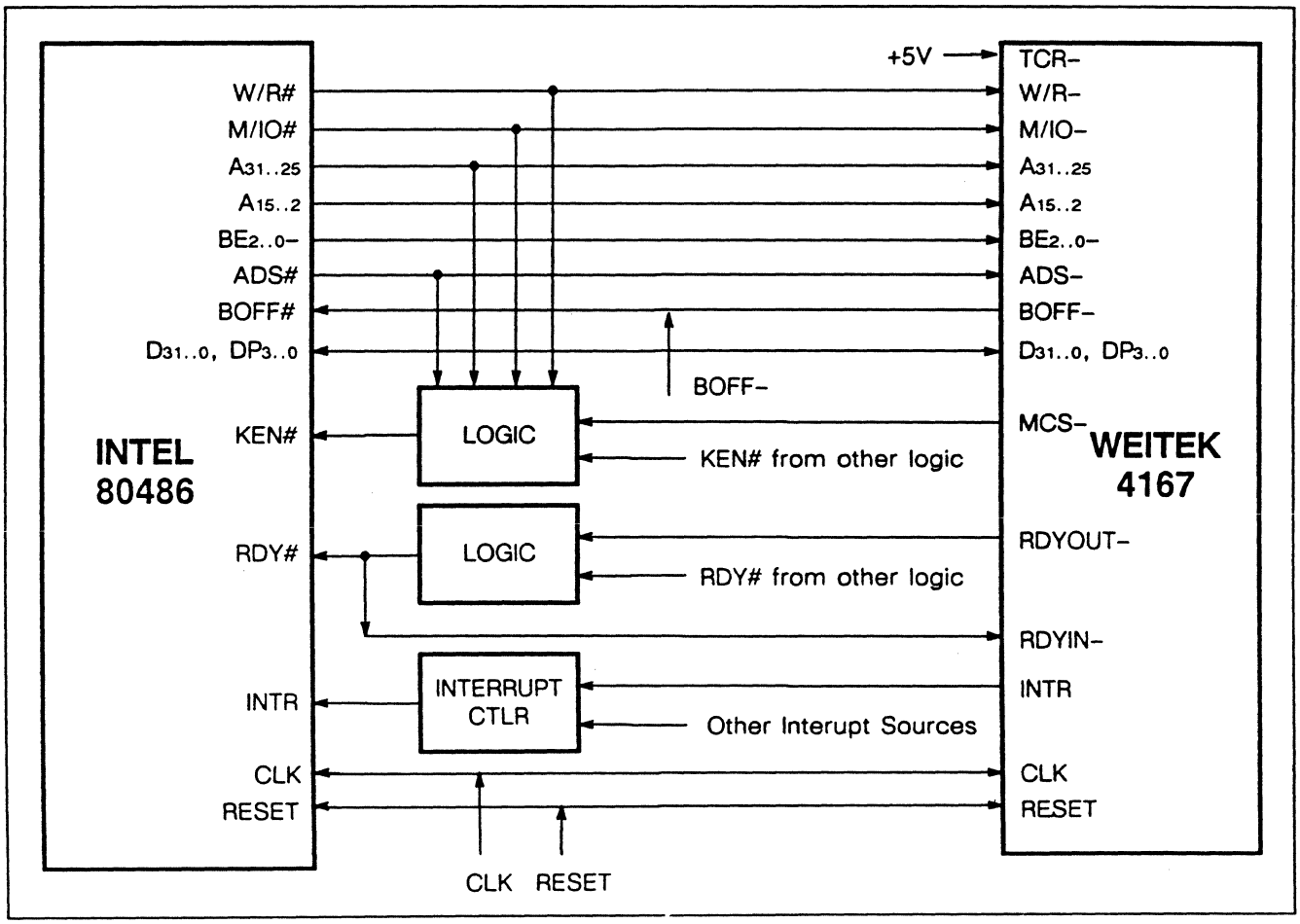


Figure 7. System connection diagram (Alternative B)

Hardware Designer's Guide, continued

SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

Supply voltage ..... -0.5 to 7.0 V
Input voltage ..... -0.5 to VDD
Output voltage ..... -0.5 to VDD

Storage Temperature Range ..... -65°C to 150°C
Operating Temperature Range ..... 0°C to 85°C

RECOMMENDED OPERATING CONDITIONS

Parameter	Test Conditions	Commercial		Unit
		Min	Max	
V_{DD} Supply Voltage		4.75	5.25	V
Tcase Operating Temperature		0	85	°C

Figure 8.

DC ELECTRICAL CHARACTERISTICS

Parameter		Test Conditions	Commercial		Unit
			Min	Max	
V_{IH}	High-level input voltage	V_{DD} = MAX	2.0		V
V_{IL}	Low-level input voltage	V_{DD} = MIN		0.8	V
V_{OH}	High-level output voltage	V_{DD} = MIN, I_{OH} = -1.0 mA	2.4		V
V_{OL}	Low-level output voltage	V_{DD} = MIN, I_{OL} = 4.0 mA		0.4	V
I_{IH}	High-level input current	V_{DD} = MAX, V_{IN} = V_{DD}		±10	µA
I_{IL}	Low-level input current	V_{DD} = MAX, V_{IN} = 0V		±10	µA
I_{CC}	Supply current	V_{DD} = MAX, f = 25 MHz		500	mA
C_{INC}	Clock input capacitance	f = 1 MHz, V_{DD} = MAX, Temp. = 25°C		30	pf
C_{IN}	Input capacitance	f = 1 MHz, V_{DD} = MAX, Temp. = 25°C		15	pf
WARNING! Remove power before insertion or removal.					

Figure 9. DC electrical characteristics over recommended temperature range

Hardware Designer's Guide, continued

AC SWITCHING CHARACTERISTICS

Symbol	Parameter	4167-025		4167-033		Unit	Ref Figure	Notes
		Min	Max	Min	Max			
T_{CY}	Clock Cycle Time	40		30		ns	12	3
T_{CH}	CLK High Time	14		11		ns	12	3
T_{CL}	CLK Low Time	14		11		ns	12	3
T_R	Clock Rise Time		4		3	ns	12	3
T_F	Clock Fall Time		4		3	ns	12	3
T_1	ADS- Setup Time	15		8		ns	15, 16, 17	
T_2	ADS- Hold Time	2		2		ns	15, 16, 17	
T_3	$A_{15..2}$, $BE_{2..0}$- Setup Time	13		6		ns	15, 16, 17	
T_4	$A_{15..2}$, $BE_{2..0}$- Hold Time	2		2		ns	15, 16, 17	
T_5	M/IO-, $A_{31..25}$ Setup Time	13		6		ns	15, 16, 17	
T_6	M/IO-, $A_{31..25}$ Hold Time	2		2		ns	15, 16, 17	
T_7	$D_{31..0}$ Setup Time	13		6		ns	15	
T_8	$D_{31..0}$ Hold Time	2		2		ns	15	
T_9	RDYIN- Setup Time	7		5		ns	15, 16, 17	
T_{10}	RDYIN- Hold Time	2		2		ns	15, 16, 17	
T_{11}	$D_{31..0}$, $DP_{3..0}$ Output Delay		30		20	ns	16	
T_{12}	$D_{31..0}$, $DP_{3..0}$ Valid Output	4		4		ns	16	
T_{13}	$D_{31..0}$, $DP_{3..0}$ Float Delay		30		20	ns	16	1
T_{14}	RESET Setup Time	10		8		ns	18	2
T_{15}	RESET Hold Time	3		3		ns	18	2
T_{16}	INTR Output Delay		31		23	ns	19	
T_{17}	INTR Valid Output	3		3		ns	19	
T_{18}	MCS- Output Delay		21		15	ns	15, 16, 17	
T_{19}	MCS- Valid Output	3		3		ns	15, 16, 17	
T_{20}	RDYOUT- Output Delay		21		15	ns	15, 16, 17	
T_{21}	RDYOUT- Valid Output	3		3		ns	15, 16, 17	
T_{22}	BOFF- Setup Time	9		7		ns	15, 16, 17	
T_{23}	BOFF- Hold Time	2		2		ns	15, 16, 17	
T_{24}	W/R- Setup Time	15		8		ns	15, 16, 17	
T_{25}	W/R- Hold Time	2		2		ns	15, 16, 17	
T_{26}	PCHK- Output Delay		25		20	ns	15	
T_{27}	PCHK- Valid Output	3		3		ns	15	
T_{28}	TCR- Setup Time	15		8		ns	16, 17	
T_{29}	TCR- Hold Time	2		2		ns	16, 17	

Functional Operating Range: V_{DD} = 5V ±5%; T_{case} = 0°C to 85°C
All parameters are specified at 1.5V unless otherwise noted
All outputs are specified with 50pf of capacitive loading
1. Tri-State timing is guaranteed, but not tested
2. Setup and Hold times specified only to guarantee recognition within a specific clock cycle
3. Parameters are at the voltage specified in the referenced figure

Figure 10. AC Characteristics

Hardware Designer's Guide, continued

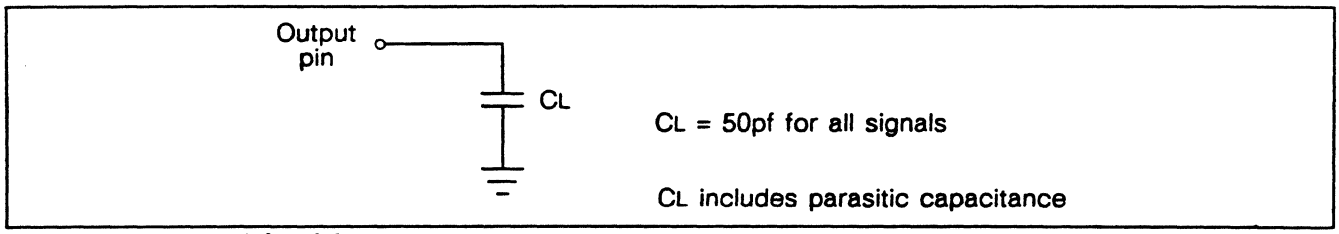


Figure 11. Test load for delay measurement

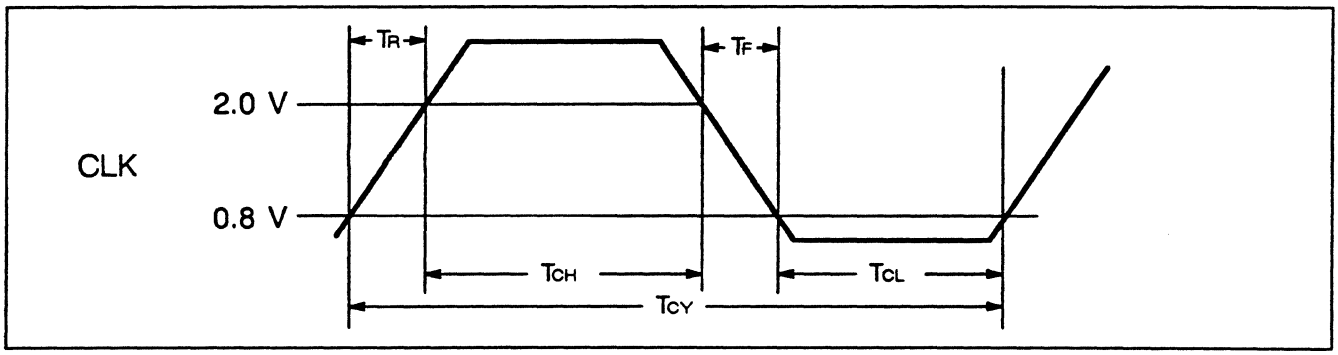


Figure 12. CLK Waveform diagram

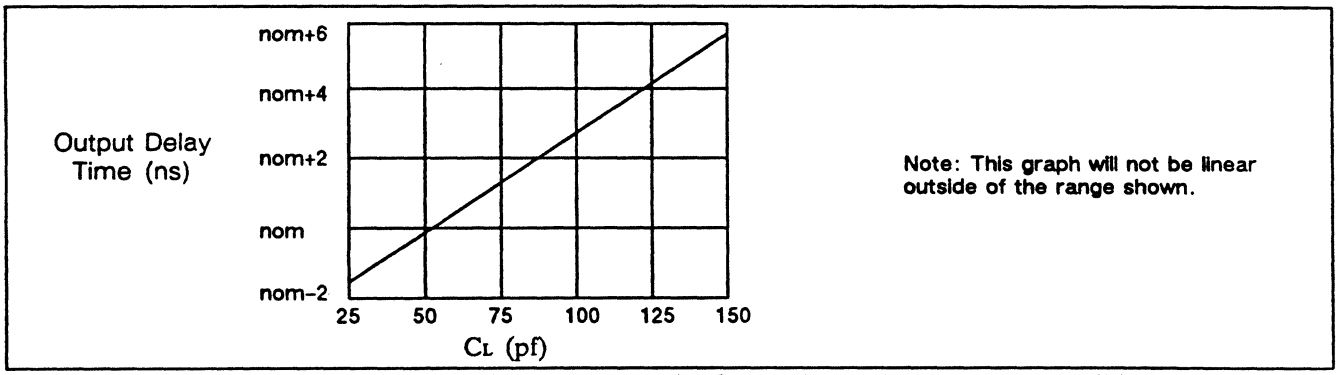


Figure 13. Typical increase in output delay time versus load capacitance in worst-case conditions

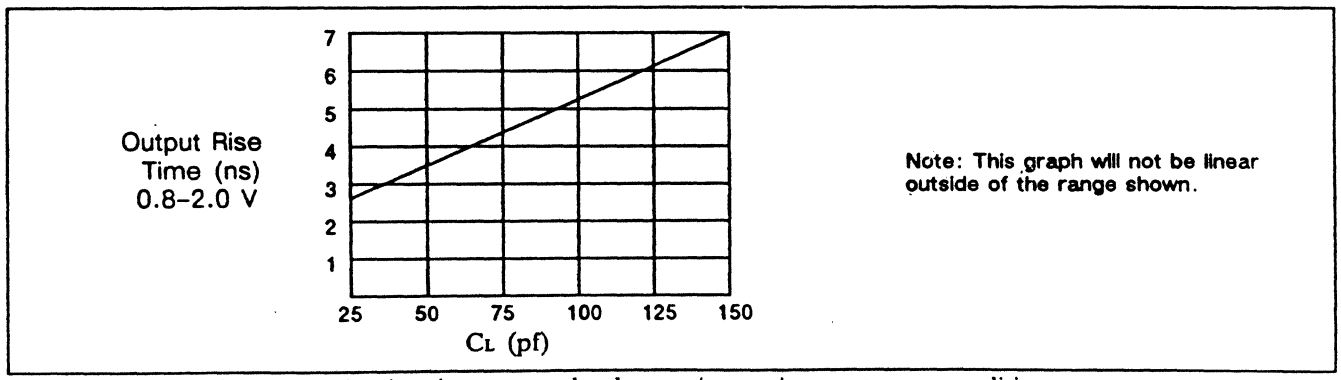


Figure 14. Typical increase in rise time versus load capacitance in worst-case conditions

Hardware Designer's Guide, continued

BUS CYCLES

Figure 15 shows two 4167 write cycles. Write cycles are performed every time the 80486 broadcasts instructions to the coprocessor. The RDYOUT- output of the 4167 handles the handshaking between the 4167 and the 80486. To acknowledge the current bus cycle, the 4167 asserts RDYOUT- and the 80486 terminates the bus cycle. The first bus write operation does not have the RDYIN- input delayed while the second does. In the delayed RDYIN- write operation, even though the bus does not advance and $D_{31..0}$ is held constant, it is latched in the same cycle it would be if RDYIN- were not delayed. Thus, if the data changes in the time slots indicated in figure 15 with crosshatching, the new data is not used by the 4167.

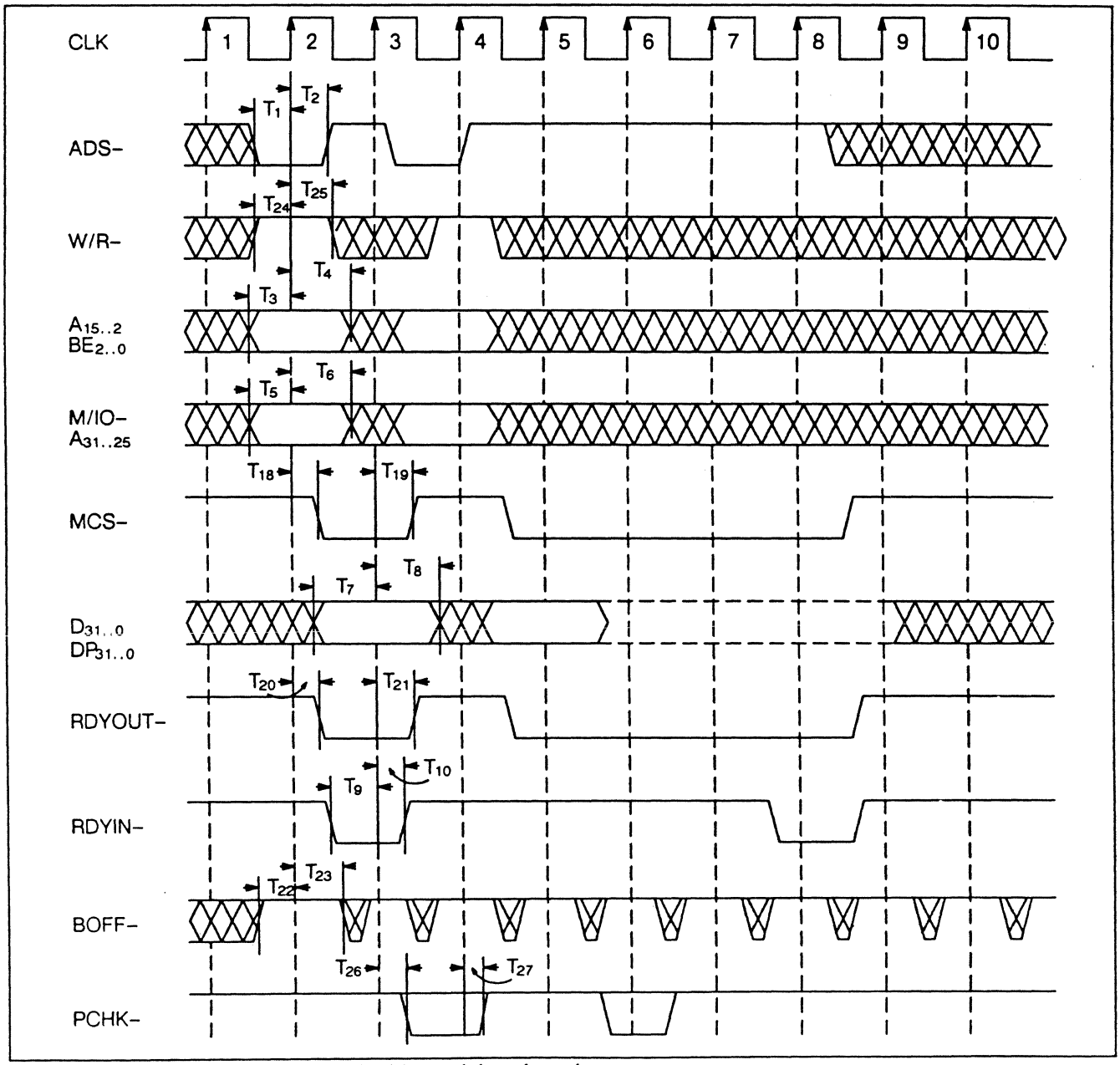


Figure 15. Bus write cycle with and without delayed ready

Hardware Designer's Guide, continued

Read cycles are performed every time data must be read from the 4167 into the 80486. Figure 16 shows a zero wait state.

If RDYIN– is delayed, the data will continue to be driven until RDYIN– is asserted. Valid data and data parity check (PCHK–) are only present when RDYOUT– is asserted.

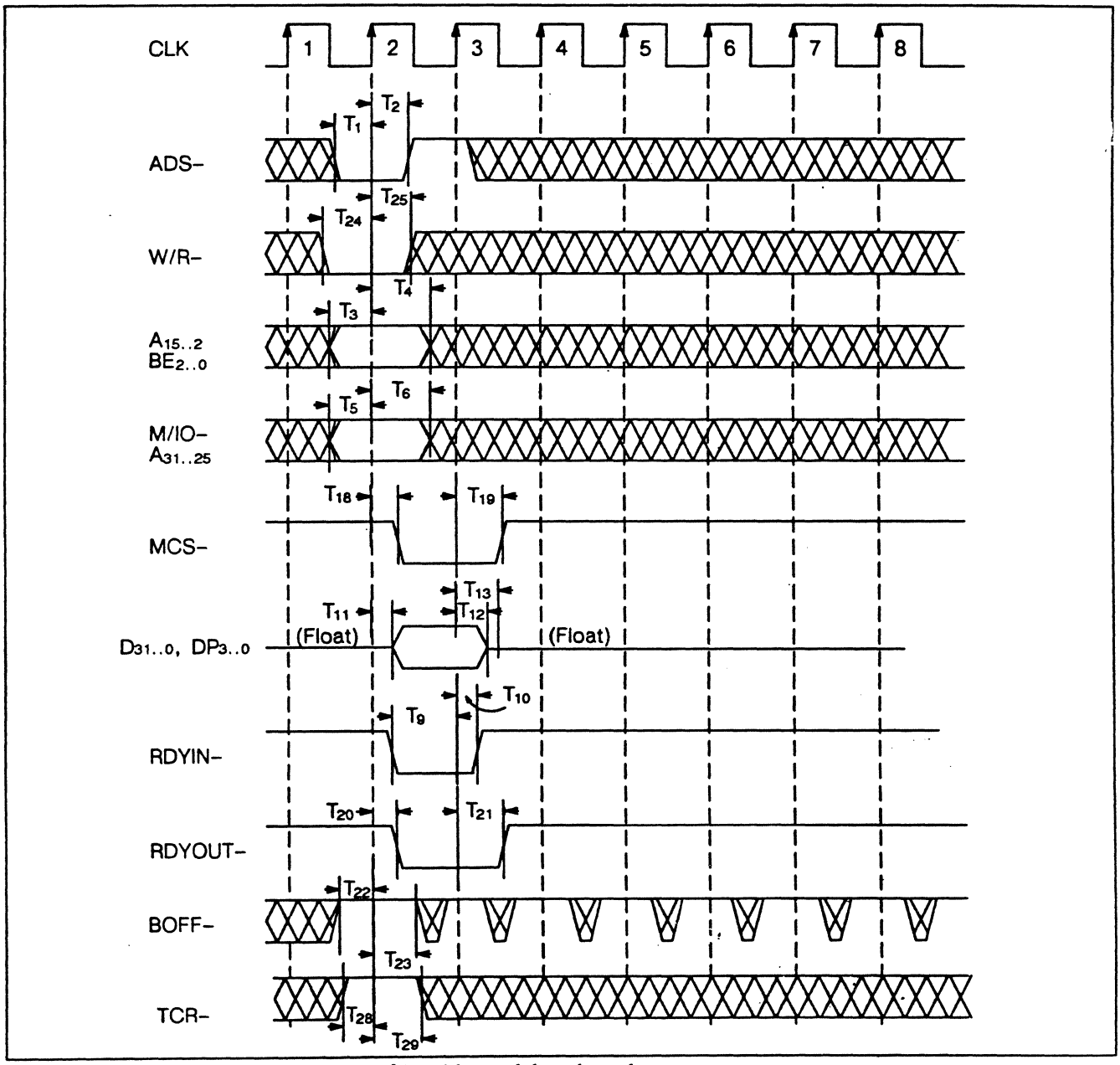


Figure 16. Zero wait state bus read cycle without delayed ready

Figure 17 shows a three wait state read cycle. When TCR- is asserted at least one wait state is always inserted during a read cycle to allow KEN# to be deasserted. If the data being read is not available, the 4167 inserts additional wait states. Wait states are fully transparent to the programmer. The maximum number of wait states is 162. Such an event only occurs when the 80486 requests a store after broadcasting five double-precision square-root or divide instructions in a row to the coprocessor. When the 4167 receives a bus read operation, it turns on its bus drivers even before the data is ready. The dotted lines in figure 17 shows the time slots during which the 4167 is driving the bus with invalid data.

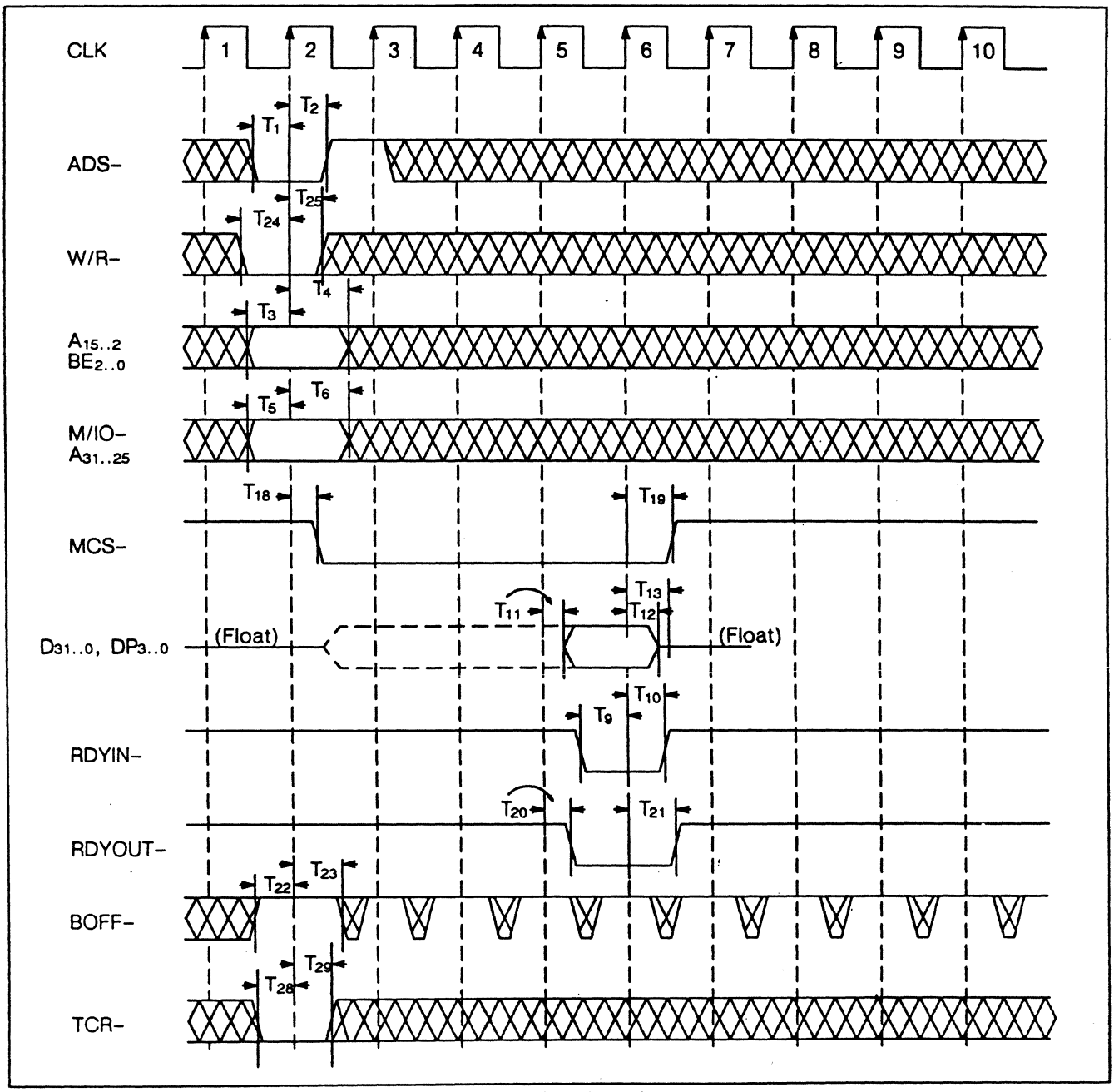


Figure 17. Three wait state bus read cycle without delayed ready

Hardware Designer's Guide, continued

RESET AND INTERRUPT TIMING

RESET set-up and hold time and interrupt valid delays are shown in figures 18 and 19. RESET need not be synchronous with the 80486's clock to guarantee proper operation, setup and hold times are shown only to guarantee recognition within a specific clock cycle.

ABORTED BUS TRANSFERS (BOFF-)

The 80486 supports the ability to abort a local bus transfer by asserting the BOFF- signal. For a complete discussion of the BOFF- signal and its effects on the system refer to the 80486 documentation.

The 4167 samples BOFF- on each clock cycle. If no 4167 bus operation is in progress then this signal is ignored. If a 4167 bus operation is in progress then the operation is aborted and MCS- is deasserted in the following cycle.

CACHE INVALIDATION CYCLES

80486 Cache invalidation cycles have no effect on the 4167. When ADS- is asserted the 4167 latches the address internally and thereafter uses the internal copy. Therefore the 4167 only samples the address bus on the cycles when ADS- is asserted. It is assumed that the assertion of AHOLD in the 80486 will suppress an ADS- in the following cycle.

80486 CACHE CONTROL

Data transferred to or from the 4167 may not be placed into the 80486 internal cache (or an external cache either). In order to prevent the caching of the data associated with the 4167 the KEN# signal must be deasserted during 4167 read operations to prevent a cache fill operation from occurring. This signal must be deasserted in the cycle prior to the assertion of RDY#.

Alternative A: When the 80486 asserts PCD, the data being read into the microprocessor will not be cached, independent of the status of the KEN# input signal.

As PCD is asserted, TCR- is deasserted, and if the data being requested from the 4167 is available, a zero wait state read is performed.

When the 80486 deasserts PCD, TCR- will be asserted and any read operation from the 4167 will require a minimum of one wait state. The MCS- signal will then be effective in deasserting KEN#.

PCD is controlled via software, by setting the PCD bit of the page table entries that map the 4167. As the default value for PCD is zero, existing application code will always see a minimum of one wait state when reading from 4167. New optimized code will see zero wait state reads leading to performance improvement.

Alternative B: As TCR- is always asserted, if the data being read is available, the 80486 will perform a zero wait state read from the 4167, independent of the code being executed. As the timing of MCS- is too slow, by one entire cycle (See figure 16), fast external logic is needed to deassert KEN# in the first cycle of the transfer.

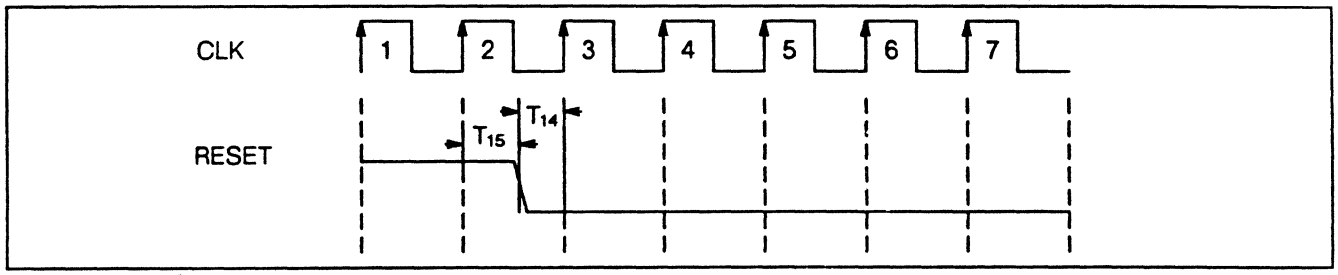


Figure 18. RESET timing

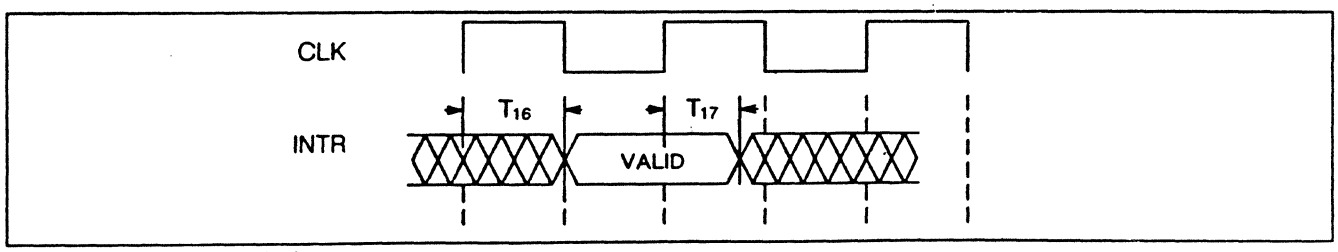


Figure 19. Interrupt timing

ROM BIOS SUPPORT

Designing the 4167 coprocessor socket on the system motherboard is necessary, but not sufficient to offer complete support for the WEITEK processor.

PC manufacturers must also provide a presence detection program, residing in the system ROM BIOS. The program will detect the presence of the WEITEK coprocessor and modify the interrupt 11H service routine so that bit 24 of the value returned in register EAX by the interrupt 11H routine is set if the 4167 coprocessor is present. Existing MS-DOS applications use such mechanism to determine whether the WEITEK coprocessor is present in the system.

MODIFYING THE ROM BIOS

ROM BIOS programmers can use several detection routines. If the hardware designer has connected the PRES- signal to an I/O port , presence detection can be accomplished by simply reading from such I/O port. Otherwise the ROM BIOS programmer can use a software sequence that loads a WEITEK coprocessor register with a specific data pattern and then reads it back. To access the WEITEK coprocessor the ROM BIOS programmer must first go into protected mode and set up page tables to address the 4167.

A code fragment that implements the detection routine is presented in figure 20. It assumes that the system is in protected mode and page tables have been set up to address the WEITEK coprocessor.

Figure 20 also shows a code fragment that modifies the interrupt 11H routine so that bit 24 of the value returned in register EAX by the interrupt 11H service routine is set if the coprocessor is present.

Hardware Designer's Guide, continued

```
; see if WEITEK coprocessor is present
; (this code assumes that the base logical address of the ABACUS is loaded into the FS register)
; first save contents of memory which may change if WEITEK coprocessor is not present
WFST      ECX, ws1
WFST      EDX, ws2
; read register ws1 into EAX
WFST      EAX, ws1
; write the data now in EAX into WEITEK coprocessor register ws2
WFLD      ws2, EAX
; complement data in EAX, save it in EBX, and write it back into register ws1
NOT       EAX
MOV       EBX, EAX
WFLD      ws1, EAX
; read the two WEITEK coprocessor registers ws1 and ws2, and compare them to EBX
WFST      EAX, ws1
CMP       EAX, EBX
WFST      EAX, ws2
; restore memory which may have changed
WFLD      ws1, ECX
WFLD      ws2, EDX
; restore Interrupts
STI
; branch if either register does not compare
JNZ       short i0init
NOT       EAX
CMP       EAX, EBX
JNZ       short i0init
; if the WEITEK coprocessor is present the system must modify the interrupt 11H routine so that
bits; 24 of the value returned by interrupt 11H in EAX is set (See Note). Application software will
then  ; use this mechanism to determine whether the WEITEK coprocessor is present.
MOV       di, offset Handlerjump
MOV       dwork ptr [di-4], 11 shl 24
i0init:                          ; WEITEK coprocessor is not present
```

Note: the code that modifies interrupt 11H assumes that the interrupt handler has been previously loaded as follows:

```
Handler:
MOV       EAX,0
Handlerjump:
JMP       far ptr original      ; Jump to original interrupt 11H handler routine
```

Figure 20. Test for presence of WEITEK coprocessor (4167 or 1167)

Hardware Designer's Guide, continued

DEBUGGING THE SYSTEM

Once the coprocessor has been designed into the motherboard and the ROM BIOS have been modified the system is ready to be debugged.

WEITEK supplies diagnostics and demo software both in the UNIX and DOS environments.

The diagnostics software tests for coprocessor presence by calling interrupt 11H, then initializes and exercises the coprocessor. Code fragments describing the presence detection and the initialization routines used by the diagnostics software are provided in figures 22, 23, and 24.

Product	Part Number
UNIX Diagnostics	4800-1167-02
DOS Diagnostics and Macros	4800-1167-03
DOS Demos	4800-1167-04

Figure 21. WEITEK-supplied support software

```
; see if WEITEK coprocessor is present
XOR         EAX, EAX
INT         11H
AND         EAX, 11 shl 24
JNZ         short J4167
K4167:      ; WEITEK coprocessor not present
J4167:      ; WEITEK coprocessor is present
```

Figure 22. Test for presence of 4167

```
WFLDCTX  B8000000h        ; load B8000000h int PCR
WFSTRL   EAX              ; store revision level
CMP      ah, 00h
JNE      short j1init
k1init:
WFLDCTX  016000000h       ; initialize Multiplier and ALU units flowthrough timers in
                          ; 1167
JMP      short i1init
j1init:
WFLDCTX  056000000h       ; initialize Multiplier flowthrough timer in 1167 type A
WFLDCTX  098000000h       ; initialize ALU flowthrough timer in 1167 type A
i1init:
; regardless of the coprocessor type load the following remaining power-up sequence
WFLDCTX  064000000h
WFLDCTX  0A0000000h
WFLDCTX  030000000h
```

Figure 23. Initializing the WEITEK coprocessor

Hardware Designer's Guide, continued

The rounding mode, the exception mask field and the accumulated exception field of the coprocessor Process Context Register are initialized as well. The instruction in figure 24 will set round to nearest rounding mode and will mask and clear all exceptions.

```
; initialize exception masks and rounding mode
WFLDCTX  003FF0000h
```

Figure 24. Exception mask and rounding mode initialization

Programmer's Interface Overview

This section provides an overview of the software tools currently available for the WEITEK coprocessors as well as a brief description of the 4167 registers, instruction set, data types, and exception handling.

Operating system developers, compiler manufacturers and application programmers that intend to provide new tools or port new applications to the 4167 should refer to the *1167 Software Designer's Guide*.

SOFTWARE TOOLS OVERVIEW

Once the 4167 has been designed into the motherboard and the ROM BIOS has been modified, the new system can take advantage of the wide selection of software tools and applications supporting the WEITEK coprocessors.

The WEITEK coprocessor is supported by the UNIX operating system (System V release 3.0). Operating system support includes coprocessor addressing, presence detection at power-up, and context-switch handling. For UNIX operating systems information contact your UNIX supplier. XENIX 386 support is also available.

The 4167 is also supported by Phar Lap, IGC, and AI Architects MS-DOS protected mode environments. MS-DOS protected mode environment support for the 4167 involves coprocessor addressing.

The WEITEK coprocessor can be supported under real mode MS-DOS as well. OEMs that intend to provide MS-DOS real mode support for the WEITEK coprocessor must offer an Extended Memory Manager that supports WEITEK coprocessor addressing. (Refer to *1167 Software Designer's Guide* for details).

C, FORTRAN and Pascal Compilers for the 80486 and 4167 under UNIX V.3 and MS-DOS protected mode are provided by Green Hills, Metaware, Microway, and Silicon Valley Software. Lahey Computer Systems offers an MS-DOS real mode FORTRAN compiler. Metaware also provides MS-DOS real mode C and Pascal compilers. Contact vendors for details.

The WEITEK Coprocessor is fully transparent to the programmer using these compilers, as the floating-point operations are specified with familiar high-level language commands. The compilers include a run-time library for transcendental operations.

TRANSCENDENTAL ROUTINES LIBRARY

WEITEK provides a library of transcendental routines to compiler developers. Routines are available through a simple license agreement.

Vendor	Product	Phone
AI Architects	OS 386 (MS-DOS protected mode environment)	(617) 577-8052
Green Hills Software	C, F, P Compilers (UNIX and MS-DOS protected mode)	(818) 246-5555
IGC	X-AM (MS-DOS protected mode environment)	(408) 986-8373
Lahey Computer Systems	F Compiler (MS-DOS real mode)	(702) 831-2500
Metaware	C, P Compilers (UNIX, MS-DOS real and protected mode)	(408) 429-6382
Microway	C, F, P Compilers (UNIX and MS-DOS protected mode)	(617) 746-7341
Phar-Lap Software	RUN386 (MS-DOS protected mode environment)	(617) 661-1510
Silicon Valley Software	C, F, P Compilers (UNIX and MS-DOS protected mode)	(408) 725-8890
SAIC	C, F, P Compilers (UNIX and MS-DOS protected mode)	(415) 960-5931
Note: F = Fortran, P = Pascal		

Figure 25. Software tools information

Programmer's Interface Overview, continued

REGISTERS OVERVIEW

The 4167 provides a register set of 32 single-precision registers, named ws0 through ws31. Pairs of 4167 registers can be used for double-precision operations, allowing up to 16 double-precision registers, numbered wd0, wd2, wd4, ..., wd30. The MSW is stored in the even register and the LSW is stored in the next contiguous odd register (that is, MSW in wsN, LSW in wsN+1). In addition, any 80486 doubleword register can be used to move data, or as the source operand to an arithmetic instruction.

The 4167 also provides a 32-bit process context register (PCR), which can be written to control rounding modes and exception handling. The context register can also be read to save control settings and read various status flags.

	MSW	LSW
wd0 →	ws0 (Restricted)	ws1
wd2 →	ws2	ws3
wd4 →	ws4	ws5
⋮	⋮	⋮
wd30 →	ws30	ws31

Figure 26. 4167 register file

INSTRUCTION SET OVERVIEW

4167 instructions can be divided into:

1. Data movement instructions
2. Format conversion instructions
3. Arithmetic instructions
4. Compare and test instructions
5. Sign manipulation instructions

Most 4167 instructions operate on either two 4167 registers or on one 4167 register and the contents of the 80486 data bus. WEITEK coprocessor macro instructions have the format:

OPCODE Source2/Destination, Source1

Source1 and Source2/Destination specify the operand addresses. The operation result is always stored in the same location as Source2. While Source2/Destination always specifies one of the thirty-two 4167 internal registers, Source1 can either specify an internal register (for register-to-register operations), an immediate constant or the content of a 80486 register (for memory-to-register operations).

Programmer's Interface Overview, continued

INSTRUCTION SUMMARY

Figure 27 summarizes the 4167 instruction set macros. Macros are available from WEITEK (P/N 4800-1167-03). All 4167 register names begin with "w". We follow the "w" with either "s" for single, "d" for double, or "x" meaning either "s" or "d". The register name ends with the letter "t" or "f". "t" stands for "to" and "f" stands for "from". For most instructions, wxt is the destination register and wxf is the source register.

```
Data Movement
WFLD      wst, wsf            ; load: wst = wsf
WFLD      wst, data           ; load: wst = 486 data
WFLD      wdt, wdf            ; load: wdt = wdf
WFLDCTX   ereg                ; load: CTX = 486 E-register
WFPOP     wst                 ; pop wst from the 486 stack
WFPOP     wdt                 ; pop two doublewords from the 486 stack to wdt
WFLDSD    wst, addr, count    ; block move: wst array = 486 memory

WFST      ereg, wst           ; store: 486 E-register = wst
WFST      ereg, wst, opcode   ; store: 486 ereg = ereg <opcode> wst
WFSTCTX   ereg                ; store: 486 E-register = CTX
WFSTCTX   ereg, opcode        ; store: 486 ereg = ereg <opcode> CTX
WFPUSH    wst                 ; push wst onto the 486 stack
WFPUSH    wdt                 ; push wdt (two doublewords) onto the 486 stack
WFSTSD    wst, addr, count    ; block move: 486 memory = wst array
WFSTRL    EAX                 ; store revision level to EAX
Format Conversion
WFLOAT    wxt, wsf            ; convert integer wsf to floating wxt
WFLOAT    wxt, data           ; convert integer 486 data to floating wxt
WFIX      wst, wxf            ; convert floating wxf to integer wst
WFIX      wst, data           ; convert floating (486 data) to integer wst

WFCVT     wst, wdf            ; convert wdf to wst
WFCVT     wst, data           ; convert double-precision (486 data and ws1) to wst
WFCVT     wdt, wsf            ; convert wsf to wdt
WFCVT     wdt, data           ; convert single-precision 486 data to wdt
```

(continued next page)

Figure 27. The 4167 instruction set macros

Programmer's Interface Overview, continued

```
Four-Function Arithmetic
WFADD     wxt, wxf          ; add: wxt = wxt + wxf
WFADD     wxt, data         ; add: wxt = wxt + (486 data)

WFSUBR    wxt, wxf          ; reversed subtract: wxt = wxf - wxt
WFSUBR    wxt, data         ; reversed subtract: wxt = (486 data) - wxt
WFSUB     wxt, wxf          ; subtract: wxt = wxt - wxf (1)
WFSUB     wxt, data         ; subtract: wxt = wxt - (486 data) (1)

WFMUL     wxt, wxf          ; multiply: wxt = wxt × wxf
WFMUL     wxt, data         ; multiply: wxt = wxt × (486 data)
WFMULN    wxt, wxf          ; negative multiply: wxt = -wxt × wxf
WFMULN    wxt, data         ; negative multiply: wxt = -wxt × (486 data)
WFAMUL    wxt, wxf          ; absolute multiply: wxt = |wxt × wxf|
WFAMUL    wxt, data         ; absolute multiply: wxt = |wxt × (486 data)|

WFMAC     wst, wsf          ; multiply and accumulate: ws2 = ws2 + wst × wsf
WFMAC     wst, data         ; multiply and accumulate: ws2 = ws2 + wst × (486 data)
WFMACD    wst, wsf          ; multiply and accumulate: wd2 = wd2 + wst × wsf (1)
WFMACD    wst, data         ; multiply and accumulate: wd2 = wd2 + wst × (486 data) (1)
WFMACD    wdt, wdf          ; multiply and accumulate: wd2 = wd2 × wdf (1)

WFDIVR    wxt, wxf          ; reversed divide: wxt = wxf ÷ wxt
WFDIVR    wxt, data         ; reversed divide: wxt = (486 data)/wxt
WFSQRT    wxt, wxf          ; square root: wxt = sqrt(wxf) (1)
WFSQRT    wxt, data         ; square root: wxt = sqrt(data) (1)

Compare and Test
WFCMPR    wxt, wxf;         ; reversed compare: set CTX flags for (wxf - wxt)
WFCMPR    wxt, data         ; reversed compare: set CTX for (486 data) - wxt
WFCMPRT   wxt, wxf          ; reversed compare with trap: set CTX flags for (wxf - wxt)
WFCMPRT   wxt, data         ; reversed compare with trap: set CTX for (486 data) - wxt

WFTST     wxf               ; test: set CTX flags for (wxf - 0)
WFTST     data              ; test: set CTX flags for (486 data) - 0
WFTST     ata, ws1          ; test: set CTX flags for double-precision (486 data, ws1) - 0
WFTSTT    wxf               ; test with trap: set CTX flags for (wxf - 0)
WFTSTT    data              ; test with trap: set CTX flags for (486 data) - 0
WFTSTT    data, ws1         ; test with trap: set CTX flags for (486 data, ws1) - 0

Sign Manipulation
WFNEG     wxt, wxf          ; negate: wxt = -wxf
WFNEG     wxt, data         ; negate: wxt = -(486 data)
WFABS     wxt, wxf          ; absolute value: wxt = |wxf|
WFABS     wxt, data         ; absolute value: wxt = |486 data|

Paging Directives
WFSPAGE                     ; force next wfld/wfst to single-precision page
WFDPAGE                     ; force next wfld/wfst to double-precision page
```

(1) These instructions are available on the 4167 and 3167, but not on the 1167

Figure 27. The 4167 instruction set macros, continued

EXECUTION TIMES FOR INDIVIDUAL INSTRUCTIONS

To estimate 4167 performance, the table in figure 28 may be used.

INSTRUCTION SET—MACHINE'S POINT OF VIEW

The 4167 is a memory-mapped device. The coprocessor is mapped in the physical memory area ranging from C0000000 H to C000FFFF H. A given address in this memory area selects the coprocessor, indicates the instruction which the 4167 has to perform, and specifies the location of Source1 and Source2/Destination. Figure 29 shows how the 4167 views a 32-bit address word.

COPROCESSOR SELECT

The most-significant 16 bits of the physical address identify a coprocessor instruction. If the upper bits do not fall in the C000–C1FF range, the address does not specify a WEITEK command and is then ignored by the 4167. To ensure compatibility with future devices, we recommend that you set the coprocessor select field to C000 when specifying a 4167 instruction.

OPCODE FIELD

The next six bits specify the coprocessor instruction to be executed. Figure 29 provides the binary and Hexadecimal offset, the hexadecimal number obtained by placing the six opcode bits into the opcode field of the address, for the 4167 instructions.

OPERAND FIELDS

The five bits of the Source1 and Source2/Destination fields identify the registers that will provide sources and destination for the instruction. If Source1 is set to zero, the Source1 data is moved over the system data bus. In order to take advantage of the 80486 block-move instruction the Source1 field is split into a three-bit and a two-bit field. The two-bit field occupies the two least-significant bits of the address. For details on Opcode and Operands (Source1 and Source2/Destination) encoding the reader should refer to the *1167 Software Designer's Guide*.

Instruction Type	Single-Precision Register-to-Register	Double-Precision Register-to-Register
LOAD, Compare, ABS	2 cycles	2 cycles
ADD, SUB, NEG, Conversion	2 cycles	2 cycles
MUL	2 cycles	3 cycles
AMUL	2 cycles	3 cycles
MULN	2 cycles	3 cycles
DIV	17 cycles	31 cycles
SQRT	17 cycles	31 cycles
MAC	4 cycles	5 cycles
MACD.S	4 cycles	
STORE*	2 cycles	

*Store operations require a variable number of cycles because they cannot be performed if the data is not available.

Figure 28. Latency

Programmer's Interface Overview, continued

GENERATING 4167 INSTRUCTIONS WITH 80486 MEMORY MOVES

Suppose that two single-precision numbers, stored in the 4167 registers WS1 and WS2, need to be added and the result stored in WS2. Since the coprocessor is mapped in the memory range C0000000-C000EFFF H, the instruction will be specified by the following coprocessor select, opcode, and operand address fields:

COPROCESSOR SELECT = C000 0000 H
OPCODE = ADD.S = 0000 H
Source1 = WS1 = 01 H
Source2/Destination = WS2 = 08 H
$A_{31}..A_0$ = C0000009 H

An 80486 move instruction which generates a physical address of C0000009 H causes the 4167 to execute the floating-point addition.

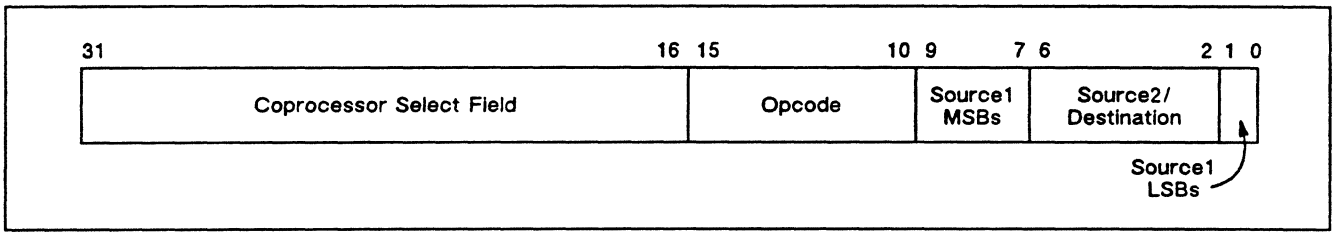


Figure 29. 4167 view of 80486 address word

Programmer's Interface Overview, continued

DATA TYPES

The 4167 floating-point coprocessor provides compatibility with the formats specified in IEEE Standard 754, Version 10.0. Several number types are required to implement the standard. The types supported by the 4167 are described below.

NORMALIZED NUMBERS (NRM)

Most calculations are performed on normalized numbers. Single-precision normalized numbers have an exponent that ranges from binary 00000001 to binary 11111110 (1 to 254) and a normalized fraction field (the leftmost or hidden bit is a one). In decimal notation, this allows one to represent a range of both positive and negative numbers from roughly 10^{+38} to 10^{-38} with accuracy to seven decimal places. Double-precision numbers have an exponent ranging from one to 2,046 and a normalized fraction field.

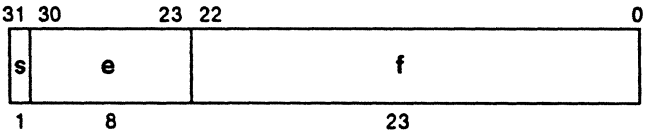
INFINITY (INF)

Infinity has an exponent of all ones and a fraction field equal to zero. Both positive and negative infinity are allowed.

ZERO

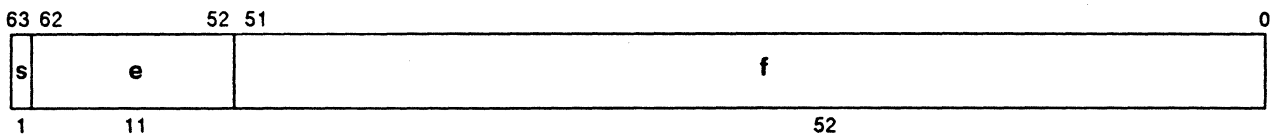
ZERO has an exponent of zero, a hidden bit equal to zero, and a value of zero in the fraction field. Both +0 and −0 are supported.

Single-Precision



e	f	Value	Name
255	not 0	none	NaN (Not A Number)
255	0	$(-1)^S \times$ infinity	Infinity
1 .. 254	any	$(-1)^S \times 2^{e-127} \times (1.f)$	Normalized number
0	0	$(-1)^S \times 0$	Zero

Double-Precision



e	f	Value	Name
2047	not 0	none	NaN (Not A Number)
2047	0	$(-1)^S \times$ infinity	Infinity
1 .. 2046	any	$(-1)^S \times 2^{e-1023} \times (1.f)$	Normalized number
0	0	$(-1)^S \times 0$	Zero

Figure 30. IEEE data types

Programmer's Interface Overview, continued

NOT A NUMBER (NaN)

NaN is a special data format usually used as a flag for data flow control, for uninitialized variables, or to signify an invalid operation such as 0 times infinity. The format for a NaN is an exponent of all ones and a non-zero fraction.

DENORMALIZED NUMBERS (DNRM)

Denormalized numbers have a zero exponent and a denormalized (hidden bit equal to zero) non-zero fraction field. They represent numbers smaller than 2^{-127} (single-precision) or 2^{-1023} (double-precision).

ROUNDING OPTIONS

The 4167 supports all four rounding modes of the IEEE standard: round to nearest, round toward zero, round toward plus infinity, and round toward minus infinity. Rounding may be biased or unbiased. Biased rounding introduces a small offset in the direction of the bias. Positive bias, negative bias, or a bias toward zero are specified in the IEEE format. Unbiased rounding rounds the result to the nearest representable number. In the case of a number exactly halfway between two representable numbers, the number is rounded toward the closest even number, resulting in half of the numbers rounding up and half rounding down, on average.

ROUND TO NEAREST (RN)

Rounds the result to the nearest representable value. If two numbers are equally near the result, the even number is chosen.

ROUND TOWARD ZERO (RZ)

Rounds the result to the value closest to but not greater than the magnitude of the result.

ROUND TOWARD PLUS INFINITY (RP)

Rounds the result to the value closest to but not less than the result.

ROUND TOWARD MINUS INFINITY (RM)

Rounds the result to the value closest to but not greater than the result.

IEEE CONSIDERATIONS

While the IEEE floating-point formats are supported by the 4167, some features of the IEEE standard are not provided due to the design focus on high speed.

EXCEPTION HANDLING

The occurrence of an enabled exception causes an interrupt. Due to extensive instruction overlapping, the exact location of an exception is not maintained. In the debugging stage of a program it is possible to identify the instruction which caused the exception by performing a store context after every floating-point instruction and then testing the enabled exception bit.

The following exceptions are flagged by the 4167:

Undefined Opcode Exception (UOE)

Whenever an illegal opcode is detected, the undefined opcode exception is set. On a read bus operation, for example, only store-type opcodes are allowed. If a read bus operation specifies any other instruction, such as multiply, then the undefined opcode exception bit is set.

Precision Exception (PE)

The precision exception (PE) flag of the accumulated exception field is set whenever there is a loss of accuracy. The coprocessor data paths compute results to higher precision than the number of mantissa bits that appear in the result. If any of the fraction bits less than the LSB was equal to one prior to rounding, then the PE bit will be set high. The precision exception will also be signaled if there is a partial or complete loss of significance in a float-to-fixed operation.

Overflow Exception (OE)

An overflow exception (OE) is generated when the result of a floating-point operation overflows the largest representable number. The result produced at the output is either infinity or the largest representable positive or negative number, depending upon the rounding mode as follows:

Largest positive normalized number	if ((RM or RZ) and the result is positive)
Largest negative normalized number	if ((RP or RZ) and the result is negative)
+Infinity	if ((RN or RP) and the result is positive)
-Infinity	if ((RN or RM) and the result is negative)

Programmer's Interface Overview, continued

Overflow is also generated when converting floating-point-to-fixed point and the result overflows the 32-bit format.

Underflow Exception (UE)

When the result of an operation after rounding is less than the minimum normalized number in the destination format, UE is asserted and the result is flushed to zero. A result of exactly zero does not underflow.

Zero Divide Exception (ZE)

The 4167 will assert a ZE exception when performing division on a normalized dividend and a zero divisor. The result is a properly signed infinity.

Invalid Operation Exception (IE)

IE is asserted if a NaN input or if an invalid operation occurs. The invalid 4167 operations are $\infty \times 0$, 0/0, ∞/∞, subtraction of like infinities ($\infty - \infty$) and addition of opposite infinities $\infty + (-\infty)$. The result of any invalid operation is a NaN with the fraction and exponent of all ones. The sign bit is zero.

FAST MODE

The 4167 always operates in Fast Mode: denormalized inputs to either the multiplier or ALU are flushed to zero as well as unnormalized outputs. The minimum normalized number has an exponent of one and a fraction field of zero. Zero has an exponent of zero and a fraction field of all zeros. This allows to represent numbers between the smallest normalized number and zero. These numbers are known as denormals (DNRM). Since denormals are very close to zero, most applications can substitute zero for a denormal without a significant loss of accuracy.

3167 and 4167 Compatibility

This section describes the hardware and software differences between the 4167 and the 3167.

HARDWARE COMPATIBILITY

The 4167 is designed to efficiently interface with the Intel 80486 microprocessor. It fits into a 142-pin socket. The 3167 is a coprocessor for the 80386 that fits into a 121-pin socket.

APPLICATION SOFTWARE COMPATIBILITY

The 4167 is upward object-code-compatible from the 3167. The application programs and all of the software tools available for the 3167 will also run on the 4167.

SYSTEM SOFTWARE COMPATIBILITY

Addressing, initialization, presence detection, exception handling, context switching, and coprocessor emulation for the 4167 are the same as they are for the 3167. Therefore, the 4167 works in all of the operating system environments that support the 3167.

Ordering Information

Part Description	Temperature Range	Order Number
25 MHz 4167 Coprocessor	T_{CASE} = 0 to 85° C	4167-025-GCU

Figure 31. 4167 Coprocessor ordering information

WEITEK

Headquarters

WEITEK Corporation
1060 East Arques
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

WEITEK U.S.A.

WEITEK Corporation
1060 East Arques
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

Corporate Place IV
111 South Bedford St.
Suite 200
Burlington, MA 01803
TEL (617) 229-8080
FAX (617) 229-4902

WEITEK Europe

Greyhound House
23/24 George St.
Richmond, Surrey
England TW9 1JY
TEL (011) 441-948-8608
TELEX 928940 RICHBI G
FAX (011) 441-940-6208

WEITEK Japan

4-8-1 Tsuchihashi
Miyamae-Ku
Kawasaki, Kanagawa-Pre
213 Japan
TEL 044-852-1135
FAX 044-877-4268