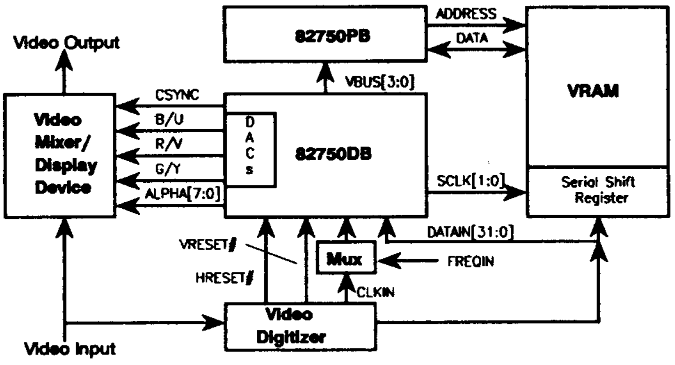
82750PB PIXEL PROCESSOR

- 25 MHz Clock with Single Cycle Execution
- Zero Branch Delay
- Wide Instruction Word Processor
- 512 x 48-Bit Instruction RAM
- 512 x 16-Bit Data RAM
- Two Internal 16-Bit Buses
- ALU with Dual-Add-With-Saturation Mode
- Variable Length Sequence Decoder
- Pixel Interpolator
- High Performance Memory Interface
  - 32-Bit Memory Data Bus
  - 50 MBytes per Second Maximum
  - 25 MBytes per Second with Standard VRAMs or DRAMs
- 16 General-Purpose Registers
- 4 Gbyte Linear Address Space
- 132-Pin PQFP
- Compatible with the 82750PA

Intel's 82750PB is a 25 MHz wide instruction processor that generates and manipulates pixels. When paired with its companion chip, the 82750DB, and used to implement a DVI® Technology video subsystem, the 82750PB provides real time (30 images/sec) pixel processing, real time video compression, interactive motion video playback and real time video effects.

Real time pixel manipulations, including 30 images/sec video compression, are supported by the 25 MHz instruction rate. On-chip instruction RAM provides programmability for execution of a wide range of algorithms that support motion video decompression, text, and 2D and 3D graphics. Inner loops are optimized with the integration of sixteen 16-bit quad ported registers, on-chip DRAM, and two loop counters that provide zero delay two-way branching "free" in any instruction. Two, 16-bit internal buses enable two parallel register transfers on each 82750PB instruction, contributing to the real time performance of the video processing. Another feature that adds to the processing power of the 82750PB is the 16-bit ALU, which includes an 8-bit dual-add-with-saturate operation critical for pixel arithmetic. Other specialized features for pixel processing include a 2D pixel interpolator for image processing functions and a variable length sequence decoder for decoding compressed data.

The 82750PB is implemented using Intel's low-power CHMOS IV Technology and is packaged in a 132-lead space-saving, plastic quad flat pack (PQFP) package.



240854-1

82750PB Subsystem Diagram

82750PB Pixel Processor

CONTENTS

82750PB Pixel Processor

CONTENTS PAGE

82750PB Pixel Processor

1.0 82750PB PIN DESCRIPTION

Pinout

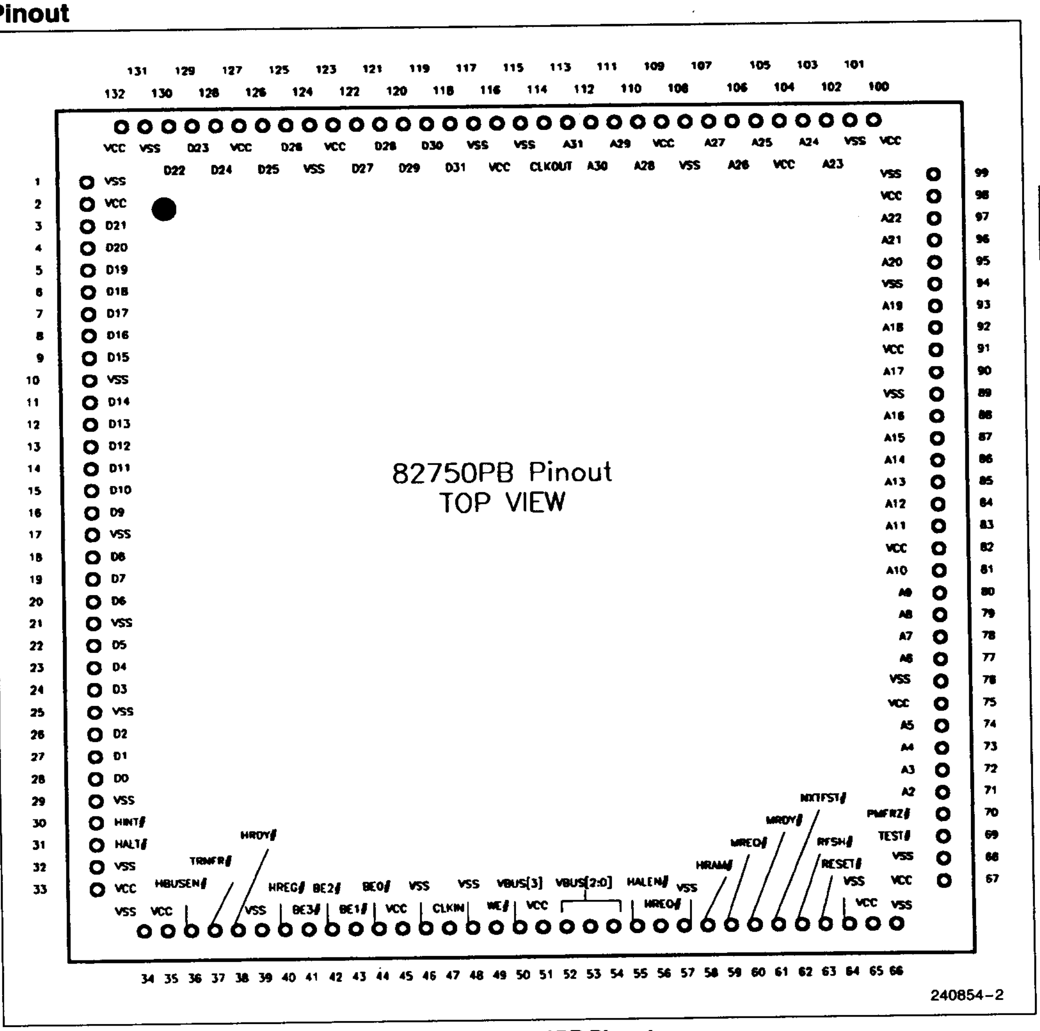


Figure 1-1. 82750PB Pinout

Table 1-1. Pin Cross Reference by Pin Name

Pin Name	Location
A2	71
A3	72
A4	73
A5	74
A6	77
A7	78
A8	79
A9	80
A10	81
A11	83
A12	84
A13	85
A14	86
A15	87
A16	88
A17	90
A18	92
A19	93
A20	95
A21	96
A22	97
A23	102
A24	103
A25	105
A26	106
A27	107
A28	110
A29	111
A30	112
A31	113
BE0#	44
BE1#	43
BE2#	42
BE3#	41
CLKIN	47
CLKOUT	114
D0	28
D1	27
D2	26
D3	24
D4	23
D5	22
D6	20
D7	19
D8	18
D9	16
D10	15
D11	14
D12	13
D13	12
D14	11
D15	9
D16	8
D17	7
D18	6
D19	5
D20	4
D21	3
D22	130
D23	129
D24	128
D25	126
D26	125
D27	122
D28	121
D29	120
D30	119
D31	118
HALEN#	55
HALT#	31
HBUSEN#	36
HINT#	30
HRAM#	58
HRDY#	38
HREG#	40
HREQ#	56
MRDY#	60
MREQ#	59
NXTFST#	61
PMFRZ#	70
RESET#	63
RFSH#	62
TEST#	69
TRNFR#	37
VBUS[0]	54
VBUS[1]	53
VBUS[2]	52
VBUS[3]	50
V_{CC}	2
V_{CC}	33
V_{CC}	35
V_{CC}	45
V_{CC}	51
V_{CC}	65
V_{CC}	67
V_{CC}	75
V_{CC}	82
V_{CC}	91
V_{CC}	98
V_{CC}	100
V_{CC}	104
V_{CC}	109
V_{CC}	116
V_{CC}	123
V_{CC}	127
V_{CC}	132
V_{SS}	1
V_{SS}	32
V_{SS}	34
V_{SS}	39
V_{SS}	48
V_{SS}	57
V_{SS}	66
V_{SS}	68
V_{SS}	76
V_{SS}	89
V_{SS}	94
V_{SS}	99
V_{SS}	101
V_{SS}	108
V_{SS}	115
V_{SS}	117
V_{SS}	124
V_{SS}	131
V_{SS}	10
V_{SS}	17
V_{SS}	21
V_{SS}	25
V_{SS}	29
V_{SS}	46
V_{SS}	64
WE#	49

Table 1-2. Pin Cross Reference by Location

Location	Pin Name
1	V_{SS}
2	V_{CC}
3	D21
4	D20
5	D19
6	D18
7	D17
8	D16
9	D15
10	V_{SS}
11	D14
12	D13
13	D12
14	D11
15	D10
16	D9
17	V_{SS}
18	D8
19	D7
20	D6
21	V_{SS}
22	D5
23	D4
24	D3
25	V_{SS}
26	D2
27	D1
28	D0
29	V_{SS}
30	HINT#
31	HALT#
32	V_{SS}
33	V_{CC}
34	V_{SS}
35	V_{CC}
36	HBUSEN#
37	TRNFR#
38	HRDY#
39	V_{SS}
40	HREG#
41	BE3#
42	BE2#
43	BE1#
44	BE0#
45	V_{CC}
46	V_{SS}
47	CLKIN
48	V_{SS}
49	WE#
50	VBUS[3]
51	V_{CC}
52	VBUS[2]
53	VBUS[1]
54	VBUS[0]
55	HALEN#
56	HREQ#
57	V_{SS}
58	HRAM#
59	MREQ#
60	MRDY#
61	NXTFST#
62	RFSH#
63	RESET#
64	V_{SS}
65	V_{CC}
66	V_{SS}
67	V_{CC}
68	V_{SS}
69	TEST#
70	PMFRZ#
71	A2
72	A3
73	A4
74	A5
75	V_{CC}
76	V_{SS}
77	A6
78	A7
79	A8
80	A9
81	A10
82	V_{CC}
83	A11
84	A12
85	A13
86	A14
87	A15
88	A16
89	V_{SS}
90	A17
91	V_{CC}
92	A18
93	A19
94	V_{SS}
95	A20
96	A21
97	A22
98	V_{CC}
99	V_{SS}
100	V_{CC}
101	V_{SS}
102	A23
103	A24
104	V_{CC}
105	A25
106	A26
107	A27
108	V_{SS}
109	V_{CC}
110	A28
111	A29
112	A30
113	A31
114	CLKOUT
115	V_{SS}
116	V_{CC}
117	V_{SS}
118	D31
119	D30
120	D29
121	D28
122	D27
123	V_{CC}
124	V_{SS}
125	D26
126	D25
127	V_{CC}
128	D24
129	D23
130	D22
131	V_{SS}
132	V_{CC}

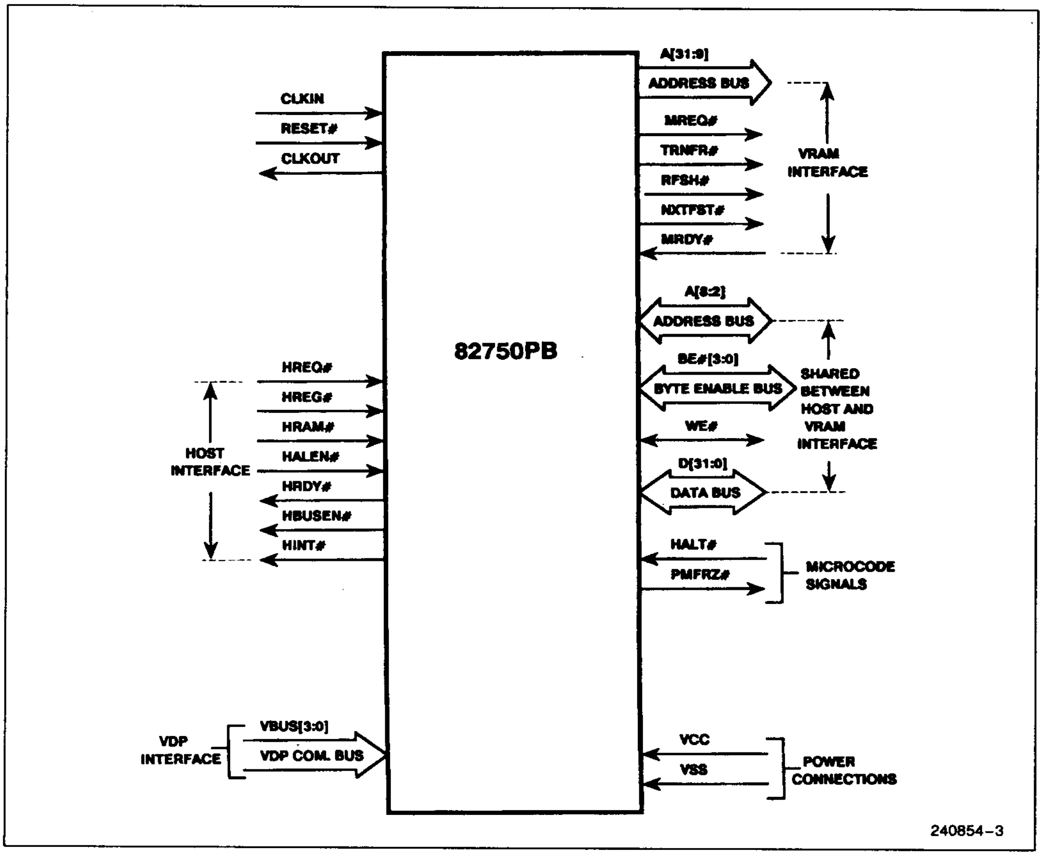


Figure 1-2. 82750PB Functional Signal Groupings

Quick Pin Reference

Table 1-3 provides descriptions of 82750PB pins.

Table 1-3. Pin Descriptions

Symbol	Type	Name and Function
CLKIN	I	CLKIN is a **1X CLOCK INPUT** that provides the fundamental timing for the 82750PB. One cycle of CLKIN is denoted as one T-cycle.
RESET#	I	The 82750PB is reset and initialized by holding this signal active for at least ten T-cycles. Refer to Initializing the 82750PB Section in Chapter 3.
HREQ#	I	The **HOST REQUEST** signal is a request from the host CPU to perform a read or write access to either registers on the 82750PB, an external device, or to VRAM shared by the 82750PB and the host. The type of access that is requested is determined by the host access definition signals: HREG#, HRAM#, and WE#.
HREG,# HRAM#	I	The **HOST REGISTER** and **HOST RAM** signals, when validated by HREQ#, are used to define three host access cycles. HRAM# active indicates the host is requesting a VRAM read or write cycle. HREG# active indicates that the host is requesting a 82750PB register read or register write cycle. When both signals are inactive, a host external cycle is requested.
HBUSEN#	O	**HOST BUS ENABLE** is asserted by the 82750PB at the start of a host access to indicate that the 82750PB Address and Data buses (A[31:2], BE# [3:0], and D[31:0]) have been tri-stated. This allows the host to drive the same buses either for accessing shared VRAM or the 82750PB internal registers.
HALEN#	I	The **HOST ADDRESS LATCH ENABLE** signal is used to indicate to the 82750PB that the host has asserted a valid address (A[31:2], BE# [3:0]) and write enable (WE#).
HRDY#	O	**HOST READY** is asserted by the 82750PB at the end of a host access to indicate that the access cycle is ready for data transfer. For a host write cycle, HRDY# indicates that the 82750PB is ready to accept data from the host. For a host VRAM write cycle, HRDY# indicates that the VRAM has latched the data from the host. For a host read cycle, HRDY# indicates that output data from the 82750PB or VRAM is ready to be latched by the host.
HINT#	O	**HOST INTERRUPT:** This output is asserted when an interrupt condition is detected by the 82750PB, and the enable bit in the PROCESSOR CONTROL register corresponding to that interrupt condition is set to a ONE. HINT# stays active until the host CPU reads the INTERRUPT STATUS register. If an interrupt condition that is enabled occurs during the same cycle that the INTERRUPT STATUS register is being read, HINT# remains active.
D[31:0]	I/O	The **DATA BUS** is used to transfer data between: 1. The 82750PB and VRAM, and 2. The Host CPU and internal 82750PB registers. During host VRAM accesses, this bus is tri-stated to allow the host to share the same VRAM data bus. During host accesses to internal 82750PB registers all 32 bits are used for data transfer.
A[31:9] A[8:2]	O I/O	The **ADDRESS BUS** is shared between the 82750PB and the host for addressing VRAM. This 30-pin bus addresses 32-bit double words in VRAM. Byte Enable signals are used to address individual bytes or words within a double word in VRAM. In addition, the address for host accesses to internal 82750PB registers are communicated to the 82750PB using the lower seven pins, A[8:2], and the BE# pins. During host access cycles to either VRAM or 82750PB internal registers, A[31:2] are tri-stated. For internal register accesses, as indicated by HREG# being low, the lower seven bits, A[8:2], are used as the host address input.
CLKOUT	O	The **CLOCK OUTPUT** signal is one of the two internal clocks and is synchronized with CLKIN. It is always driven and will have a 50% duty cycle.

Table 1-3. Pin Descriptions (Continued)

Symbol	Type	Name and Function
BE# [3:0]	I/O	The **BYTE ENABLE BUS** is shared by the 82750PB and the host for addressing VRAM down to the byte level. The correspondence between the four Byte Enable pins and the D[31:0] pins is: BE# [3]-D[31:24], BE# [2]-D[23:16], BE# [1]-D[15:8], and BE# [0]-D[7:0]. During VRAM read cycles, the 82750PB enables all four bytes. During write cycles the 82750PB only enables those bytes that are to be written. Bytes that are not enabled are not to be altered in VRAM. During host accesses to 82750PB on-chip registers, the BE# [0] pin is used as an input to select whether the even or odd word is being accessed; the double word address is provided by the host on the A[8:2] pins. BE# [0] = 0 indicates that data is transferred on D[15:0]. BE# [0] = 1 indicates that data is transferred on D[31:16].
MREQ#	O	**MEMORY REQUEST** is asserted for the first cycle, T1, of each VRAM cycle.
TRNFR#, RFSH#	O	The **MEMORY CYCLE DEFINITION SIGNALS:** Transfer, Refresh and Write Enable are asserted at the same time as MREQ#, but stay active for the entire VRAM cycle. TRNFR# active indicates a VRAM transfer cycle. RFSH# active indicates a VRAM refresh cycle. If neither TRNFR# nor RFSH# are active, a VRAM data read or write cycle is requested.
WE#	I/O	The **WRITE ENABLE** pin is used as an output during an 82750PB/VRAM cycle to drive the WE# signal, which defines the access as a VRAM read cycle (when inactive) or write cycle (when active). During Host/VRAM and Host External cycles, the 82750PB tri-states this pin to allow the host to drive the VRAM write enable signals directly. During Host/register cycles, this pin is used as an input for the Host Write Enable signal to determine whether the host is reading or writing the 82750PB register.
NXTFST#	O	The **NEXT FAST** signal indicates that the following vram cycle can be performed with a page-mode or bank-interleaved access. This signal is asserted during the first of a pair of VRAM cycles that is guaranteed to be within the same VRAM page and in opposite banks—a pair of accesses to two sequential double words in VRAM at addresses Even Address and Even Address + 1. In other words, A[2] is a zero for the first cycle and a one for the second cycle.
MRDY#	I	The **MEMORY READY** input indicates that the VRAM cycle has progressed to the point where it is ready to perform the data transfer. For a VRAM read cycle, the VRAM data can be latched by the transition of MRDY# to an active state. For a VRAM write cycle, MRDY# indicates that the data has been latched into the VRAMs.
VBUS[3:0]	I	The **VDP COMMUNICATION BUS** is used to communicate from the 82750DB to the 82750PB. Codes sent over this bus indicate interrupt requests, transfer requests, and status information. Since the 82750DB and 82750PB run asynchronously, the VBUS signals are sampled on the falling edge of CLKIN and compared with the previous sample. For a VBUS code to be detected by the 82750PB, it must be valid for two successive samples.
HALT#	I	The **HALT** signal causes the microcode processor on the 82750PB to halt prior to executing the next instruction. This signal does not halt the VRAM interface. The Halt signal will allow the design of a hardware emulator for the 82750PB based on an 82750PB chip.
TEST#	I	The **TEST** signal is used for test purposes only and must remain high for normal operation.

Table 1-3. Pin Descriptions (Continued)

Symbol	Type	Name and Function
PMFRZ#	O	The **PERFORMANCE MONITORING AND FREEZE** signal is toggled by specific microcode instructions and can be used to determine the time required to execute certain sections of microcode.
V_{CC}	I	**POWER** pins provide the +5V D.C. supply input.
V_{SS}	I	**GROUND** pins provide the 0V connection to which all inputs and outputs are referenced.

Table 1-4. Output Pins

Name	Active Level	When Floated
CLKOUT	High	Always Driven
A[31:9]	High	Reset*, Host Cycle
HBUSEN#	Low	Reset*
HRDY#	Low	Reset*
HINT#	Low	Reset*
MREQ#	Low	Reset*
TRNFR#, RFSH#	Low	Reset*
NXTFST#	Low	Reset*
PMFRZ#	Low	Reset*

*The reset state is caused by RESET# being active low.

Table 1-5. Input Pins

Name	Active Level	Synchronous/ Asynchronous
CLKIN	High	Synchronous
RESET#	Low	Asynchronous
HREQ#	Low	Asynchronous*
HREG#	Low	Synchronous
HRAM#	Low	Synchronous
MRDY#	Low	Synchronous
VBUS[3:0]	High	Asynchronous
HALT#	Low	Synchronous
HALEN#	Low	Asynchronous*

*Can be programmed to accept synchronous inputs.

Table 1-6. Input/Output Pins

Name	Active Level	When Floated	Synch/Async
D[31:0]	High	Reset*, Host Cycle	Synchronous
A[8:2]	High	Reset*, Host Cycle	Synchronous
BE#[3:0]	Low	Reset*, Host Cycle	Synchronous
WE#	Low	Reset*, Host Cycle	Synchronous

*The reset state is caused by RESET# being active low.

All output pins are floated when RESET is active low.

2.0 ARCHITECTURE

Overview

The 82750PB includes a wide instruction word processor that comprises a number of processing, storage, and input/output elements. The wide instruction word architecture allows a number of these elements to operate in parallel. The 82750PB executes one instruction every internal clock cycle or T-cycle. The various elements are connected via two 16-bit buses, the A bus and B bus, as shown in Figure 2-1. During each instruction execution cycle, data can be transferred from a bus source to a bus destination element on both buses.

Registers

{*rN; N = 0–15*}

There are 16 general-purpose data registers, each 16 bits wide, that are connected to both the A bus and B bus as both sources and destinations. These registers are designated *r0–r15*. All the registers are functionally identical except *r0*, which also includes logic for bit shifting and byte swapping. A register can source both the A bus and the B bus in the same cycle. A register cannot be the destination of both the A bus and the B bus in a single instruction. Because the registers are doubly latched, the same register may be both a source and destination in the same cycle. The result is that the data in the register prior to the current cycle will be driven on the source bus, and the data on the destination bus will be latched into the register at the end of the cycle.

Register *r0* has additional logic to allow bit shifting and byte swapping. The value in *r0* can be shifted left or right one bit position per instruction cycle. For a right shift, the new MSB is equal to the old MSB; in other words, the value is sign-extended. For left shifting, the new LSB is equal to zero. *R0* cannot be shifted and loaded in the same instruction. Byte swapping, on the other hand, only occurs when *r0* is being loaded with a value from the A bus or B bus. Byte swapping causes the most significant byte and the least significant byte of the 16-bit value being loaded into *r0* to be interchanged. Refer to Chapter 4 for a description of the SHFT microcode field that controls the shifting and swapping operations in *r0*.

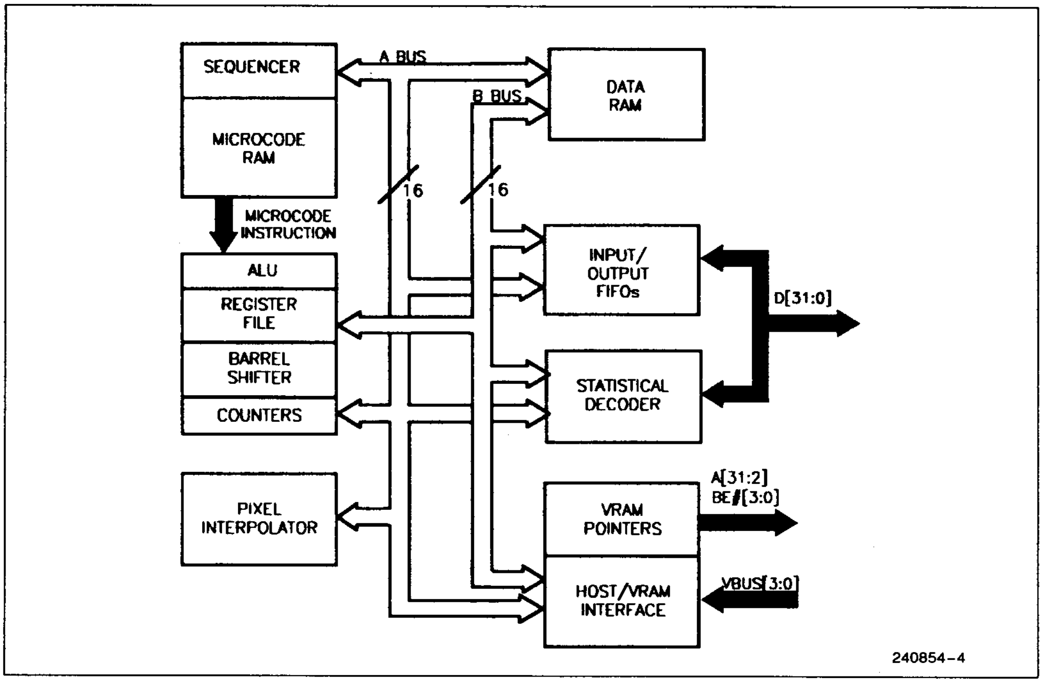


Figure 2-1. 82750PB Block Diagram

ALU

{*alu, cc*}

The ALU performs 16-bit arithmetic and logic operations, and can also be operated as two independent 8-bit ALUs for the Dual-Add-with-Saturate operation. There are two fields in the microcode instruction that affect the operation of the ALU: the ALUOP field specifies the operation to be performed, and the ALUSS field specifies the source of the two ALU inputs. Refer to Chapter 4 for further information on these fields.

The two ALU operands either come from values held in the ALU input latches or from "eavesdropping" on the A or B buses. The result of any ALU operation is latched in the ALU output register, *alu*. In a subsequent instruction this result can be transferred to any A or B destination.

The ALU has four condition flag outputs: CarryOut, Sign, Overflow, and Zero. CarryOut is the carry out of the most significant bit position. Sign is equal to the value of the most significant bit of the result. Overflow is the exclusive-OR of CarryOut and the CarryIn to the most significant bit position of the result. Zero is true (a value of 1) if all 16 bits of the ALU result are equal to zero. CarryOut and Overflow are defined as equal to zero for all logical operations. For most ALU operations, the state of these four condition flags are latched when the operation is complete. There are eight operations (nop, a*, b*, +], −], 0*, prof and int) that are exceptions. These operations are performed without disturbing the condition state of the previous ALU operation.

Microcode routines can read and write the ALU condition flag register, *cc*. This can be used to save and restore the state of these flags. The bit ordering of the ALU condition flags within *cc* are given in Table 2-1. A complete list of ALU opcodes is given in Table 2-2.

Table 2-1. Bit Assignments for *cc* Register

Bit	Condition
Bit 0	False (This bit of the *cc* is always read as a zero.)*
Bit 1	ALU Carry Out
Bit 2	ALU Overflow
Bit 3	ALU Sign
Bit 4	ALU Zero
Bit 5	Loop Counter Zero*
Bit 6	R0 LSB*
Bit 7	R0 MSB*
Bit 15:8	RESERVED. The state of these bits is undefined when read; write as zeros.

*These are read-only values and are not affected by writes to the cc register.

Table 2-2. ALU Opcodes

Operation	Mnemonic
No Operation	nop
pass a	a
pass b	b
1's Compliment of a	~a
1's Compliment of b	~b
a AND b	&
(NOT a) AND b	~&
a AND (NOT b)	&~
a OR b	|
a XOR b	^
a + b	+
a + b + 1	++
a − b	−
−a + b	−+
2's Compliment of a	−a
2's Compliment of b	−b
Increment a	a++
Increment b	b++
Decrement a	a−−
Decrement b	b−−
Dual Add with Sat.	+]
a + b + (Prev. Carry)	+<
a − b − (Prev. Borrow)	−<
−a + b − (Prev. Borrow)	−+<
Interrupt Host	int
Zero	0*
Pass a, Don't Latch Flags	a*
Pass b, Don't Latch Flags	a*
(NOT a) OR b	~|
a OR (NOT b)	|~
Dual Sub. with Sat.	−]
Performance Monitor	prof

The Dual-Add-with-Saturate operation performs independent 8-bit ADDs on the upper and lower bytes of the two ALU operands. The two bytes of the A operand are treated as unsigned binary numbers ($00:FF_{16}$ corresponds to $0:255_{10}$). The two bytes of the B operand are treated as offset binary numbers with an offset of $+128$ ($00:FF_{16}$ corresponds to $-128_{10}:127_{10}$). The upper and lower byte results are treated as 9-bit offset binary, including the carry output of each byte, with a $+128$ offset ($000:1FF_{16}$ corresponds to $-128_{10}:383_{10}$) and are saturated to a range of $0-255_{10}$. A result that is less than zero is set equal to zero or 00_{16} and a result that is greater than $+255$ is set equal to $+255$ or FF_{16}.

In fact, this operation is symmetric. Either the A operand or the B operand can be defined as the unsigned binary value, and the other operand will be treated as the offset signed binary value.

Dual-subtract-with-saturate is similar to dual-add-with-saturate. It calculates $A - B + 128$ on each 8-bit half of the two 16-bit inputs, and clamps the results to 0 and 255. This can be viewed as subtracting an offset-binary signed byte (-128 to 127) from an unsigned byte (0 to 255).

The ALU opcode INT generates the MCINT (microcode interrupt) condition. When this condition is detected by interrupt logic in the host CPU interface, and if the Enable MCINT bit in the PROCESSOR CONTROL register is set to a ONE, the host interrupt output, HINT#, will be asserted. Refer to Chapter 3 for further information on host interface.

The 'prof' opcode activates the PMFRZ# pin, and is primarily used for performance monitoring and/or debugging.

Barrel Shifter

{*shift, shift-r, shift-rl, shift-l*}

The barrel shifter performs a single cycle, n-bit left or right shift. The barrel shifter operates independent of the ALU. The three barrel shifter operations are: *Shift-r* for a right shift with sign extend; *Shift-rl* for right shift with zero fill; and *Shift-l* for a left shift with zero fill. The shift operation is invoked by writing a 4-bit value (the shift amount) to one of three A bus registers, depending on which of the three operations is to be performed. The operand is taken from the B bus, and the result is stored in the barrel shifter output register, *Shift*. Like the ALU result register, the value in *Shift* can be read onto the A bus or B bus in the following instruction cycle.

A barrel shifter operation does not affect any of the condition flags.

Data RAM

{*dramN, *dramN, ++, --; N = 1-4*}

The Data RAM holds 512, 16-bit words that are accessed using four pointers. To access a value in a particular location, the microcode routine must first load a pointer with the address to be accessed, and then perform a read or write using the same pointer. In parallel with the data RAM access, the pointer can optionally be post-incremented or post-decremented. The four pointers, referred to as *dram1–dram4*, can be written and read via the A bus. When a dram pointer, which is only 9 bits wide, is read onto the A bus, its upper seven bits are set to zeros.

NOTE:

The width of the dram pointers may change in later versions of the 82750PB. Software should not rely on the width of a pointer to, for example, mask the upper seven bits of a value to zero.

All four pointers can be used to read or write the Data RAM from either the A or B bus. Only one Data RAM access can be performed in any cycle. A Data RAM access is referred to, using C language syntax, as **dram1*. The * means "the value pointed to by". As another example, **dram3++* means access the Data RAM using the pointer *dram3* and increment *dram3*. The symbol -- in place of the ++ would indicate autodecrement.

Loop Counters

{*cnt,cnt2*}

Two 16-bit loop counters are available to microcode programs for automatically counting iterations of a microcode loop. In parallel with other operations performed in an instruction, either loop counter can be decremented, and a conditional branch can be made based on the loop counter value being equal or not equal to zero. Since the two loop counters can be written and read on the A bus, as *cnt* and *cnt2* respectively, they can also be used for variable storage when not being used as loop counters. The loop counters can be written to and decremented during the same instruction cycle. The value in the counter at the start of the next cycle will be the value written to the counter minus one.

The LC microcode bit determines the loop counter that is selected for decrementing and/or branching in an instruction. The LC microcode bit does not affect the loop counter that is written or read over the A bus, since each loop counter is separately addressable as an A bus source or destination. Refer to Chapter 4 for a description of the CNT -- microcode bit that causes the select loop counter to be decremented, and for a description of the CFSEL microcode field that is used to perform a conditional branch based on the selected loop counter's value.

Microcode RAM

{*mcode1-3, maddr, pc*}

The 82750PB executes instructions stored in an on-chip microcode RAM. This RAM holds 512 instructions and each instruction is 48 bits wide. Normally, to start the microcode processor, the host CPU will load a microcode program into the microcode RAM, point the program counter, *pc*, to the start of the program, and then release the HALT bit to start executing the microcode program. The microcode processor can also load its own microcode RAM to overlay new routines and therefore, does not require constant intervention by the host to perform multiple operations.

Writing an instruction into Microcode RAM is done by first loading the three registers *mcode3, mcode2*, and *mcode1* with the three 16-bit words of the instruction (the most significant word goes into *mcode1*), and then loading the address where the instruction should be written into *maddr*.

The host CPU can also read the Microcode RAM by first loading the *pc* with the address of the instruction to be read and then reading the three 16-bit words of the instruction from the *mcode1-mcode3* registers. Normally, this would be done by the Host CPU while the 82750PB is halted. Since *mcode1-mcode3* hold the instruction pointed to by the *pc* (i.e. the instruction that is about to be executed), normally reading these three registers from a microcode routine is not useful.

The read registers named mcode1-mcode3 and the write registers also named mcode1-mcode3 are in fact different registers. Writing values into mcode1-mcode3 and then reading the values of mcode1-mcode3 will not read back the same values just written. The read registers hold the instruction stored in the instruction latch (the instruction to be executed). The write registers hold an instruction that is about to be written into microcode RAM.

After writing to *maddr* to load an instruction into microcode RAM, a one cycle freeze occurs and during the freeze a write to the microcode RAM takes place. The instruction following the write to *maddr* can either jump to the address just loaded or start loading the *mcode1-mcode3* registers with the next instruction to be written.

Here are two examples that illustrate the fact that the 82750PB requires at least one instruction between the write to maddr and the execution of the instruction that is loaded by the write to maddr.

Example 1:

```
maddr = ADDR1         /* load instruction */
jmp addr1             /* jump to it, this is the extra inst. required between */
                      /* writing to maddr and executing the loaded inst. */
. . .
ADDR1:
???????????           /* here's where new instruction gets loaded */
```

Example 2:

```
maddr = INST
nop                   /* extra instruction */
INST:
???????????           /* instruction gets loaded here */
```

When a microcode routine writes to *pc*, one more instruction is executed before the jump to the new address takes effect. For example:

```
pc = ADDR1
r0 = r1   jmp ADDR2   /* this instruction gets executed but */
                      /* its jump to ADDR2 is ignored. */
. . .
ADDR1:
r3 = r0               /* after this instruction executes r3 = r0 = r1 */
```

When the host CPU writes to the *pc*, the instruction at the address that was written is loaded into the *mcode1–mcode3* registers and, when the microcode processor is released from its Halt condition, this is the first instruction that will be executed.

When the host CPU reads the *pc*, the result returned is the address of the instruction that will be executed when Halt is released, that is, the address of the instruction held in the *mcode1–mcode3* registers.

Horizontal Line Counter

{*lcnt*}

The 12-bit Horizontal Line Counter is updated by VBUS codes from the 82750DB to track the horizontal display line that is currently being scanned by the 82750DB. The counter is reset by a VODD code and incremented each time an HLINE code is received. A value can also be written into a Horizontal Line Counter but this is used primarily for testing the 82750PB. The upper four bits will always read zeros.

Field Counter

{*fcnt*}

The 4-bit field counter is updated by VBUS codes from the display processor to keep track of the field count being displayed by 82750DB. The counter is incremented each time a V_{ODD} code or V_{EVEN} code is received. When reading the field counter, the upper 12 bits will read zeros. This counter will not be initialized upon reset.

Input FIFOs

{*inN-lo, inN-hi, inN-c, *inN; N = 1, 2*}

There are two input channels, referred to as input FIFOs, through which the processor can read pixels or data from VRAM. Each channel automatically fetches 64-bit quad words from VRAM and breaks them into 8-bit bytes or 16-bit words that are read by microcode. Each input FIFO operates independently and can be programmed to automatically increment or decrement through bytes or words in VRAM. The FIFOs are double buffered so that while values are being extracted from one quad word (64 bits), the next quad word is being prefetched from VRAM.

The mode control register for each input FIFO, designated *in1-c* or *in2-c*, contains four mode bits as seen in Figure 2-2. The WORD/BYTE bit (bit 0) determines whether the input FIFO is in word mode (WORD/BYTE = 0) or byte mode (WORD/BYTE = 1). In byte mode, the FIFO can start reading on any byte boundary and in word mode on any word boundary.

The INC/DEC bit (bit 1) determines the order that bytes or words are read from VRAM. In INCREMENT mode, with INC/DEC = 0, the FIFO reads from the least significant byte or word to the most significant byte or word of each double word and increments through double words in VRAM. In DECREMENT mode, with INC/DEC = 1, the FIFO reads from most significant byte or word to least significant byte or word within a double word and decrements through double words in VRAM.

The AHOLD bit (Bit 2) is used by the address hold mode. When asserted, (bit 2 = 1) the automatic address increment/decrement function will be disabled and input FIFOs will not double buffer VRAM data. In other words, at the end of a VRAM cycle, when the FIFO has been updated with 64 bits of VRAM data, the input FIFO will not issue another MREQ# until there is a write to the address-lo registers OR a roll-over/roll-under read access of the input FIFO. If a roll-over/roll-under occurs, then a memory request will be issued to fetch data from the same VRAM location. If there is a write to the address-lo register, the FIFO will then fetch data from the new location.

The PREFETCH OFF bit (bit 3) specifies whether the FIFO will automatically prefetch successive quad words from VRAM or will only fetch a new quad word when a value from that quad word is requested. In PREFETCH-ON mode, bit 3 = 0, the input FIFO prefetches successive quad words from VRAM as necessary to keep its buffer full (either from ascending or descending addresses, depending on the state of the INC/DEC bit). In PREFETCH-OFF mode, the FIFO will still prefetch the first two quad words to fill its buffer (when started at a new address location), but will only fetch a new quad word when a read request is made to the FIFO for a value in the next unfetched quad word.

The CB bit (bit 4) allows circular buffers of sizes 64 kBytes, 128 kBytes, or 256 kBytes to be created in VRAM memory. The choice of different sizes of buffers are determined by programming the least significant 3 bits of the circular buffer register (cir-

bits:	15...4	5	4	3	2	1	0
	Set to Zeros	BY-32 MODE	CB	PF OFF	AHOLD	INC/DEC	WORD/BYTE

Figure 2-2. Input FIFO Control Register

cbuf). To enable this feature, the CB bit has to be set to a 1, then depending on the buffer size selected, the appropriate address pin that goes off chip will be forced to a 0 (register pointers remain unchanged). Table 2-3 shows the programming combinations of the circular buffer register.

It is important to note that the internal address counters themselves are not affected by the circbuf function. Only the selected external address pin is forced to '0'.

Table 2-3. Circular Buffer Register (circbuf)

Bits [2:0]	Buffer Size	Effect on PB Address Bus (If Function Enabled)
000	Disabled	None
100	256 kBytes	Address Pin 18 Forced to 0
010	128 kBytes	Address Pin 17 Forced to 0
001	64 kBytes	Address Pin 16 Forced to 0

In "BY-32" MODE (bit 3), the pointer increments or decrements by 32 bits, independent of whether the FIFO is in 8-bit pixel mode or 16-bit pixel mode. This mode was added to facilitate microcode that operates on one component of a 32-bit per pixel image.

The standard sequence for initializing an input FIFO is to write to the control register *(in-c)*, the high address *(in-hi)*, and then the low address *(in-lo)* of the appropriate FIFO. Refer to the access state diagram in Chapter 3. The write to *in-lo* causes the FIFO to start reading from VRAM. A byte or word is then read from **in*. Successive reads from **in* will read sequential bytes or words from VRAM. Writing to the control register each time the FIFO is started at a new address is not necessary, except to change the FIFO's mode. Also, if the new address is within the same 64 kByte page of VRAM, only the lo-address needs to be written in order to start the FIFO reading from the new address.

If microcode attempts to read a value from an empty input FIFO, the processor is frozen prior to the execution of the instruction, until the FIFO's control logic has fetched another double word from VRAM and extracted the next value. At this point, the processor is released from the frozen state, and the instruction that reads the value is executed. When the processor is frozen waiting for a particular FIFO that isn't yet ready, that FIFO's VRAM access priority is raised above all other FIFOs.

Output FIFOs

{*outN-lo, outN-hi, outN-c, *outN, outN+ +; N = 1, 2*}

There are two output channels, referred to as output FIFOs, through which the graphics processor writes pixels or data to VRAM. Each channel automatically collects bytes or words into 64-bit quad words and writes the quad words to VRAM. Each output FIFO operates independently and can be programmed to write bytes or words into sequential addresses in VRAM (either incrementing or decrementing). The FIFOs are double buffered so that while one quad word is waiting to be written to VRAM, the next quad word can be assembled from individual bytes or words.

The mode control register for each output FIFO, designated *out1-c* or *out2-c*, contains six mode bits as shown in the Figure 2-3. The WORD/BYTE bit (bit 0) determines whether the output FIFO is in word mode (WORD/BYTE = 0) or byte mode (WORD/BYTE = 1). In byte mode the FIFO can start writing on any byte boundary in VRAM and in word mode on any word boundary.

The INC/DEC bit (bit 1) determines the order that bytes or words are written to VRAM. In INCREMENT mode, with INC/DEC = 0, the FIFO writes from the least significant byte or word to the most significant byte or word in a double word and increments through double words in VRAM. In DECREMENT mode, with INC/DEC = 1, the FIFO writes from most significant byte or word to least significant byte or word within a double word and decrements through double words in VRAM.

When the AHOLD bit (bit 2) is set, the output FIFO quad word address is not incremented or decremented. In this mode, the FIFO continues to output to a single quad word in VRAM.

The FORCE-LSB bits (bits 3 and 4) are used to force the least significant bit of each byte written to VRAM to either a zero or a one. This can be used, for example, to force the LSB to the correct polarity when writing to the U bitmap during motion video decompression. In certain display modes for the 82750DB, the LSB of the 8-bit samples in the U or Y bitmap are used to select VIDEO or GRAPHICS display mode for the n x n group of display pixels corresponding to the particular U or Y sample. A one in the FORCE-

bits:	15–6	5	4	3	2	1	0
	Set to Zeros	BY-32 MODE	FORCE-LSB ENABLE	FORCE-LSB VALUE	AHOLD	INC/DEC	WORD/BYTE

Figure 2-3. Output FIFO Control Register

LSB ENABLE bit (bit 4) enables the forcing; a zero results in normal operation. The FORCE-LSB VALUE bit (bit 3) is used as the value to which the LSB is forced. Whether in byte mode or word mode, the LSB of *each byte* is forced to the FORCE-LSB value.

In "BY-32" MODE (bit 5), the pointer increments or decrements by 32 bits, independent of whether the FIFO is in 8-bit pixel mode or 16-bit pixel mode. This mode is used to facilitate microcode that operates on one component of a 32-bit per pixel image. The bytes or words that are skipped over will be unchanged in VRAM.

The standard sequence for initializing an output FIFO is to write to the control register *(out-c)*, the low address *(out-lo)*, and then the high address *(out-hi)* of the appropriate FIFO. A series of bytes or words is then written to **out*. Refer to the access state diagram in Chapter 3 (Figure 3-1).

In order to flush any remaining data in an output FIFO before changing its VRAM pointer, it is necessary to write to the control register. When pointing to a new location in VRAM, if the new address is within the same 64 kByte page of VRAM, only the lo-address needs to be written.

There must be one instruction between the write to the output FIFOs low address and the first write to *outN. Therefore, it is recommended that outN-lo be written before outN-hi. The write to outN-hi insures that this requirement is met. If only the outN-lo value is being changed, it is still necessary to have one additional instruction before the first write to *outN.

When writing bytes or words to VRAM through an output FIFO, a byte or word can be skipped over by writing to *outN+ +* instead of **outN*. When the values are written to VRAM, any byte or word that was skipped will retain its original value in VRAM, and its value is not altered by the VRAM write. This can be used when writing a series of pixels, some of which are "transparent", allowing whatever was behind them to show through.

If the microcode routine attempts to write a value to a full output FIFO, the processor is frozen prior to the execution of the instruction. The processor remains frozen until the FIFO has a chance to write one of the buffered quad words to VRAM. At that point, the processor is released from the frozen state, and the instruction that writes the value is executed. When the processor is frozen, waiting for a particular FIFO that isn't yet ready, that FIFO's VRAM access priority is raised above all other FIFOs.

Statistical Decoder

{*stat-lo, stat-hi, stat-c, stat-ram, *stat, *stat#* }

The Statistical Decoder (also referred to as the Huffman Decoder) is a specialized input channel that can read a variable-length bit sequence from VRAM and convert it into a fixed-length bit sequence that is read by the microcode processor. In image compression, as well as in other applications such as text compression, certain values occur more frequently than others. A means of compressing this data is to use fewer bits to encode more frequently occurring values and more bits to encode less frequently occurring values. This type of encoding results in a variable-length sequence in which the length of a symbol (the group of bits used to encode a single value) can range for example, from one bit to sixteen bits.

The statistical code that the statistical decoder can decode is of either of the two forms:

```
0x                            1x
10x                           01x
110xxx                        001xxx
1110xxxxx                     0001xxxxx
. . .                   or    . . .
1111110xxxxxx                 0000001xxxxxx
11111110xxxxxx                00000001xxxxxx
. . .                         . . .
```

Each symbol of a given length (one per line as shown here) consists of a run-in sequence followed by some number of x-bits. The run-in sequence is defined as a series of zero or more ONEs followed by a ZERO or, as in the code on the right above, zero or more ZEROs followed by a ONE. The remainder of this description will use examples of the code on the left. A bit in the decoder's control register determines the polarity of the run-in sequence bits.

In the example on the left, there would be two symbols of length two: 00 and 01. Each x-bit can take on a ZERO or ONE value. The number of x-bits following a run-in sequence can range from zero to six. Since the goal, in general, is to have a few short codes and a larger number of long codes, typically, codes with fewer run-in bits will have fewer x's following. However, this is not a hardware constraint. A code of this form is completely described by a code description table indicating: for each length of run-in sequence, R = the number of ONEs in the run-in, and how many x-bits follow the ZERO. The value of R is used as an index into the code description table. Due to the hardware implementation, the number actually stored in the table is 2^x, where x is the number of x-bits.

For the example above, the corresponding code description values are given in Table 2-4.

Table 2-4. Sample Code Description Table

R	X	2^x(dec.)	2^x(bin.)
0	1	2	000 0010
1	1	2	000 0010
2	3	8	000 1000
3	5	32	010 0000
...			
7	6	64	100 0000

Note that the table only goes up to symbols with seven ONEs in the run-in. For symbols with more than seven ONEs, the value of X and 2^x for seven ONEs is used for all symbols having seven or more ONEs in the run-in sequence. For example, in the code above a symbol with eight or more ONEs in the run-in sequence has six x-bits following the ZERO, which is the same as symbols having seven ONEs.

For each different symbol, including all symbols of the same run-in length with different x-bit values, the decoder generates a unique fixed-length, 16-bit value. Some of the decoded values for the sample code given above are provided in Table 2-5.

Table 2-5. Decoded Values

Symbol*	Decoded Value
0**0**	0
0**1**	1
10**0**	2
10**1**	3
110**000**	4
110**001**	5
110**010**	6
...	...
110**111**	11
1110**00000**	12
...	...
1110**11111**	43
...	...

*The x-bits of the symbol are in **boldface** for clarity.

The algorithm for generating a decoded value from a symbol is as follows: all symbols of a given run-in length are assigned a base value, B; the value corresponding to a particular symbol is equal to B plus the binary value of the x-bits in the symbol. The base valule B for a symbol with a run-in length of R is calculated by:

$$B(R) = SUM[2^{X(r)}] \text{ with } r = 0 \text{ to } R - 1,$$

where X(r) corresponds to the X value in the table entry corresponding to R = r.

For example, in the above code:
$B(0) = 0$, B (0) is always zero
$B(1) = 0 + 2 = 2$
$B(2) = 0 + 2 + 2 = 4$
$B(3) = 0 + 2 + 2 + 8 = 12$
$B(4) = 0 + 2 + 2 + 8 + 32 = 44$

This is one of the reasons that the table holds 2^X instead of X. The calculation of B(R) are easier to implement in logic.

There are two enhancements that are made to this coding scheme in the implementation on the 82750PB. These two modes are referred to as END mode and SHORT mode. If neither END nor SHORT mode are enabled, the decoding is performed as described above. SHORT mode allows the decoder to be switched easily to a simpler code format without having to reload the code description table. In the SHORT form, all symbols have the same number of x-bits, as though all entries in the table had been filled with the same value of 2^X. When SHORT mode is invoked, this value of 2^X is obtained from a field in the statistical decoder's CONTROL word, instead of from the individual table entries.

END mode is added in recognition of the fact that, for codes with few symbols, some increase in efficiency is possible by not having to place a zero at the end of the longest run-in sequence. For example, consider the code:

0
10x
110x

The END mode allows us to shorten the last symbol to 11x instead of 110x. The trailing ZERO is not required because the decoder has been told that the maximum length of a run-in is two ONEs. The resulting symbol set and corresponding decoded values are given in Table 2-6.

Table 2-6. END Mode Decoded Values

Symbol	Decoded Value
0	0
100	1
101	2
110	3
111	4

The number of x-bits must be constant for all symbols of the same run-in length. Therefore, a code such as:

```
0
10xx
11xxx  ←  NOT CORRECT! . . . Must be 11xx.
```

is not allowed. The last symbol (11xxx, in this case) uses the same table entry for 2^X as the next to last symbol (10xx) and, therefore, the last symbol will be 11xx.

The maximum length of the run-in sequence in END mode is specified by placing an END flag in the code description table. For example, a code and the corresponding table is shown in Table 2-7.

Table 2-7. END Flag Decoded Values

Code	Table Entries		
	Index	**END Bit**	**2^X**
0	0	0	0
10xx	1	0	4
110xxx	2	1	8
111xxx	3	-	-
	4	-	-
	5	-	-
	6	-	-
	7	-	-

The hyphens indicate that those table entries aren't used to decode this code. Note that the symbol 111xxx has three x-bits because of the value of 2^X in Index 2; it is not based on the 2^X value in Index 3.

The SHORTED and END modes can be invoked simultaneously, resulting in a code such as:

```
0x
10x
110x
111x
```

with a SHORT -2^X value = 2 (for 1 x-bit in each symbol) and the END bit set in Index 2.

Packed binary fields with one to seven bits per field can be read using the statistical decoder by setting the END bit in Index 0 and by programming the X value to be N − 1, where N is the number of bits per field. For example, packed three-bit fields could be decoded as shown in Table 2-8.

Table 2-8. Packed 3-Bit Field Decbded Values

Code	Table Entries		
	Index	**END Bit**	**2^X**
0xx	0	1	4{N = 3, so X = 2}
1xx	1	-	-
	2	-	-
	3	-	-
	4	-	-
	5	-	-
	6	-	-
	7	-	-

The unpacked bits are in reverse order relative to how they are stored in VRAM. For example, if three-bit values are packed in VRAM, the pattern 110 in VRAM is read from right to left and gives an unpacked or decoded value of 3.

The CONTROL register for the statistical decoder *(stat-c)* is used to specify the mode to use for decoding, as well as to invoke certain modes for writing and reading the code description table. Refer to the bit assignments for this register below. To write to the code description table, the WRITE bit (bit 4) is set to a ONE; the starting table index is reset to zero. Each write to the table causes the index to increment by one. This index will wrap around from seven back to zero. For example, to write all eight table entries the user would write a value of 0x10 to *stat-c* register and then write eight 8-bit values to the register *stat-ram*. The most significant bit of each 8-bit value is the END bit, and the lower seven bits are the values of 2^X. To read the code description table, the TEST bit (bit 4) of the CONTROL register is set to a one. The table entries are then read from the decoder's data register *(*stat)*. Reads and writes always start at table entry zero.

NOTE:

*When reading the code description table, it is necessary to wait one instruction time between the write to stat-c and the first read from *stat. An access diagram showing all legal sequences for read and write FIFC registers is shown in Chapter 3 (Figure 3-1).*

The code for reading the eight table entries into the first eight locations of data RAM would be:

```
        dram3 = 0        stat-c = 0x20     /test mode to read the stat-ram (the table)
        cnt = 8                            /wait one inst. before first read
        LOOP:
                         *dram3++ = *stat cnt--
                         jcp loop          /two inst. loop necessary to wait one inst.
                                           /between each read from *stat.
```

Bits	15	14	13	12:8	7	6	5	4	3	2:0
	POL	RSVD*	CB	SVAL	SHORT	END	TEST	WRITE	RSVD*	Starting *Stat-ram* ADDRESS

* Reserved: write zeros to these bits.

Figure 2-4. Statistical Decode CONTROL Register

END mode is enabled by setting the END bit (bit 6) in the CONTROL register to a ONE. The SHORT mode is enabled by setting the SHORT bit (bit 7) in the CONTROL register to a ONE. When in SHORT mode, the five SVAL bits (bits 12:8) in the CONTROL register are used as the SHORT $- 2^X$ value.

The POL bit (bit 15) determines the polarity of the run-in sequence bits. If bit 15 = 0, then ONEs ending in ZERO (e.g., 1110xxx) sequence is selected. If bit 15 = 1, the ZEROs ending in ONE (e.g., 0001xxx) sequence is selected.

The CB bit (bit 13) allows circular buffers of sizes 64 kBytes, 128 kBytes, or 256 kBytes to be created in memory, as in the case of the input FIFO. The choice of different sizes of buffers are determined by programming the least significant 3 bits of the circular buffer register (circbuf). To enable this feature, the CB bit has to be set to a 1, then depending on the buffer size selected, the appropriate address pin that goes off chip will be forced to a 0 (register pointers remain unchanged). Table 2-3 shows the programming combination of the circular buffer register.

The decoding parameters may be changed between symbols by writing to the CONTROL register and, if necessary, writing new values into the code description table. The correct procedure for changing the code type or decode mode is to read the last value from the decoder prior to the change, using **stat#* instead of **stat*. This keeps the decoder from automatically starting to decode the next symbol. At this point, the code description table and the SHORT and END mode bits can be changed as desired. The next time the CONTROL register is written with both TEST = 0 and WRITE = 0, the decoder will begin to decode the next symbol using the new parameters.

The statistical decoder buffers one quad word read from VRAM so that the decoding of bits in one 32-bit word and the fetch of the next 32-bit word may overlap. As with the input and output FIFOs, the decoder has a VRAM pointer associated with it that points to the location in VRAM from which it is reading data. This pointer increments twice each time a new quad word is read; there is no decrement mode. When the least significant word of the decoder's pointer *(stat-lo)* is written, any data that had previously been prefetched from VRAM is ignored, and the decoder fetches one quad word starting from this new location.

The 82750PB assumes that the statistically encoded bitstream in VRAM starts with the least significant bit of a *double* word. That is, the two LSBs of the address written to start-lo are ignored.

The statistical decoder decodes data at a rate of one bit per T-cycle. To a first approximation, the decode time for an N-bit symbol is:

$$\text{decode time (in T-cycles)} = N + 1$$

Since it takes at least 64 T-cycles to decode data from one quad word, which is the time required for eight quad word reads from VRAM, the decoder should rarely run out of data. Therefore, the above estimate should very accurately model the actual decoding rate of the statistical decoder.

The statistical decoder always begins to read the bitstream from the least significant bit of the double word found at the starting location in VRAM. That is, the decoder does not start on a byte or word boundary as an input FIFO or output FIFO does, but only on double word boundaries. The bitstream moves from the least significant bit to the most significant bit of a double word and then to the least significant bit of the next double word (at the next higher address location). For the x-bits, the first x-bit read from the bitstream becomes the most significant bit of the x-bit field when it is interpreted as a binary number. The example below shows a code definition, a bitstream stored in VRAM, and the resulting decoded values.

The code definition and range of values for each symbol length are indicated in Table 2-9.

Table 2-9. VRAM Bitstream Decode Values

Symbol	Values	Comments
0	0	
10x	1, 2	100 = 1, 101 = 2
110xx	3–6	11000 = 3, . . . , 11011 = 6
1110xxx	7–14	1110000 = 7, . . . , 1110111 = 14

Decoding starts at address 0 in this example. The two double words at addresses 0 and 1 are:

0: 0xAC98E14D

1: 0x372E74CB

The bitstream in VRAM, with colons dividing the symbols (read from right to left starting at LSB of address 0) is shown in Figure 2-5.

Table 2-10 lists the symbols, in the order they are encountered in the bitstream, and the corresponding decoded values.

Table 2-10. Decoding Symbols

Symbol	Value	Comments
101	2	Starts at LSB, Address 0, Scanning Left
100	1	
101	2	
0	0	
0	0	
0	0	
0	0	
1110**001**	8	
100	1	
100	1	
110**10**	5	
1110**100**	11	Spans First and Second Double Word
110**01**	4	
0	0	
1110**011**	10	
101	2	
0	0	
0	0	
1110**110**	13	
. . .	. . .	

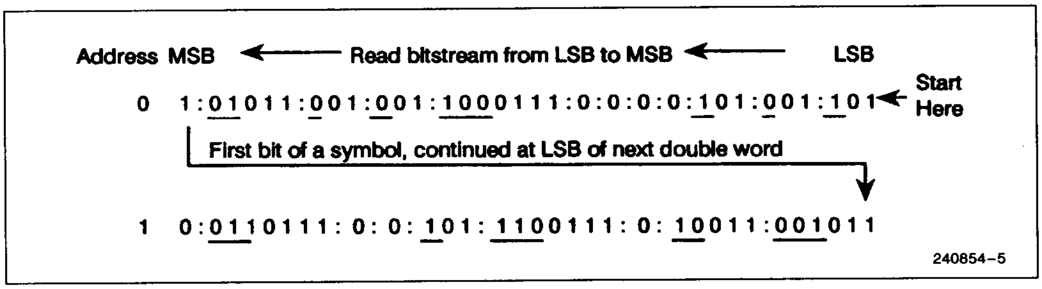


Figure 2-5. VRAM Bitstream Decoding Addresses

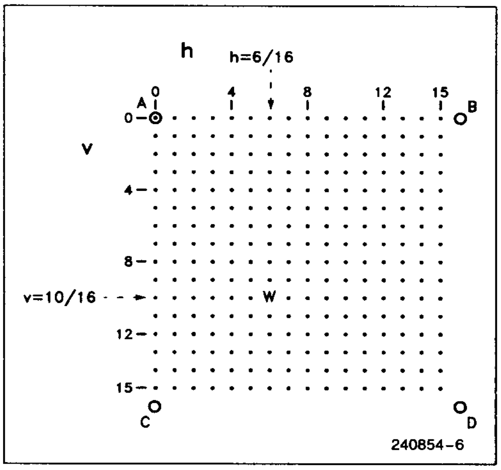


Figure 2-6. Pixel Interpolation

Pixel Interpolator

{*Pixint-c, Pixint*}

The pixel interpolator performs bilinear interpolation on four 8-bit pixels to generate, in effect, a pixel shifted by a fraction of a pixel position. See Figure 2-6. If the four pixels have values of A, B, C, and D; and the horizontal weight and vertical weight are h and v, respectively, the interpolated value W, ignoring any quantization effects, is given by:

$$W = A^*(1-h)(1-v) + B^*h(1-v) + C^*(1-h)v + D^*hv$$

The values of h and v are even multiples of 1/16. Figure 2-6 illustrates pixel interpolation with an h weight of 6/16 or 3/8 and a v weight of 10/16 or 5/8.

The pixel interpolar can operate in two modes: sequential-2D and random-2D. Sequential-2D mode is used for motion video decoding and when an array of pixels are interpolated with a common weighting. Random-2D mode is used either when the pixel arrays to be interpolated are not adjacent pixels in two rows or when the weight is changed for each interpolation. (The word random is used here to mean non-sequential.)

The example in Figure 2-7 shows a single row of pixels being interpolated in Sequential-2D mode using two rows from the original (source) bitmap. The h and v weighting are constant for all the interpolated pixels. In this case, the weights appear to be approximately h = 10/16 and v = 6/16.

A		B		E		F		I	...	—First Input Row
	W		X		Y		Z		...	—Interpolated Row
C		D		G		H		K	...	—Second Input Row

Figure 2-7. Sequential-2D Pixel Interpolation

The pixel interpolator is pipelined and requires some startup sequence to fill the pipeline. Once filled, the pixel interpolator generates a new interpolated pixel every two T-cycles when in Sequential-2D mode. Source pixels are written into the interpolator as pixel pairs. In the case above, the pixel pair BA would be written first, followed by the pixel pair DC. It would seem more natural to refer to the pixel pair as AB, but because of the way 8-bit pixels are arranged in 16-bit words in VRAM, the left-most pixel on the screen is the least significant byte position. For example, if pixel A had a hex value of 0xAA and B had a value of 0xBB, the 16-bit word containing pixels A and B would have a value of 0xBBAA.

Then, two pixels are read from the interpolator. Because the pipeline isn't full yet, these pixels are read and discarded. This loop of writing two pixel pairs and reading two output pixels continues four times. The two pixels that are read this fourth time are the first two valid output pixels: W and X. The interpolator may also collect output (interpolated) pixels into pixel pairs. For example, pixels W and X, instead of being output separately, would be combined into a 16-bit pixel pair XW. Since there are two possible phase relationships between the input pixel pairs and output pixel pairs, the desired phasing (either X and W paired or Y and X paired) can be specified.

```
bits    15      14      13      12      11      10      9       8       7:4     3:0
        ^RESERVED—Write as ZERO
                ^Pipelining Select (1 = Fast, 0 = Standard)
                        ^Phase (0 = In Phase, 1 = Opposite Phase)
                                ^RESERVED—Write as ZERO
                                        ^Pairing (1 = Output Pixel Pairs, 0 = Single Pixels)
                                                ^Reset Bit (1 = Reset, 0 = Normal)
                        Mode Select Bits -----------------^-------^
                                                                Vertical Weight ----^
                                                                    Horizontal Weight ------^
```

Figure 2-8. Pixel Interpolator Control Register

Random-2D interpolation is used either when the pixels to be interpolated are not in horizontal rows or when the weight is changed for each interpolated pixel. Examples for this are smooth warping or smooth scaling operations. In the case of Random-2D, the processing for successive interpolated pixels cannot take advantage of pipelining; each pixel is considered to be the first pixel of a Sequential mode interpolation. The weight and the two input pixel-pairs are written into the interpolator. After waiting at least 10 T-cycles, the one interpolated pixel can be read. (The delay is 10 cycles when in the standard mode (bit 14 = 0) and 6 T-cycles when in the fast mode (bit 14 = 1).) Then, the next two input pixel-pairs and if necessary, the new weight value, are written, and 10 cycles later the next interpolated pixel can be read.

The h and v weight values, the mode selection, and other control bits are written to the pixel interpolator control register *(avg-c)*. The bit assignment for this register is in Figure 2-8. The least significant byte holds the 4-bit v value (bits 7:4) and the 4-bit h value (bits 3:0).

NOTE:

The values used for h and v here are numerators of the fraction where the implied denominator is 16.

MODE SELECT

Bits 8 and 9 are used to select one of four operating modes, of which only two are presently defined. These modes are given in Table 2-11.

Table 2-11. Mode Select Operating Modes

Bits 9:8	Mode
00	RANDOM-2D
01	Sequential-2D
10	RESERVED
11	RESERVED

RESET

Writing a ONE to bit 10 resets the pixel interpolator. The pixel interpolator must be reset prior to changing modes.

PAIRING

A ZERO in bit 11 causes the pixel interpolator to output individual pixels. A ONE causes the interpolator to collect adjacent pixels (in Sequential-2D mode) into 16-bit pixel pairs. This feature assists in motion video decoding, when combined with the ALU's dual-add-with-saturate operation, by allowing two pixels to be processed each cycle. The phasing used in collecting the pixel pairs is determined by the Phase bit described below.

PHASE

When output pixels are collected into pixel pairs, there are two possible alignments of the input pixel pairs to the output pixel pairs. The Phase bit (bit 13) selects the alignment to be used, based on the relative word alignment of the source and destination bitmaps in VRAM. When the Phase bit is set to a ZERO, this indicates that the bitmaps are in-phase. In this case, the first two output pixels are grouped into one 16-bit pixel pair (with the first pixel in the least significant byte). When the Phase bit is set to a ONE, the bitmaps are out-of-phase. In this case, the first pixel is placed in the most significant byte of the first pixel pair, with invalid data in the least significant byte, and the second and third output pixels are collected into the second pixel pair. This is illustrated in Figure 2-9.

PIPELINING

A ZERO in bit 14 causes the pixel interpolator to use the standard amount of pipeline delay. A ONE in this field will select the fast mode that has less pipeline delay. Table 2-12 shows the pipelining delay for both modes. Note that the effect of the phase bit is to add an extra pixel delay.

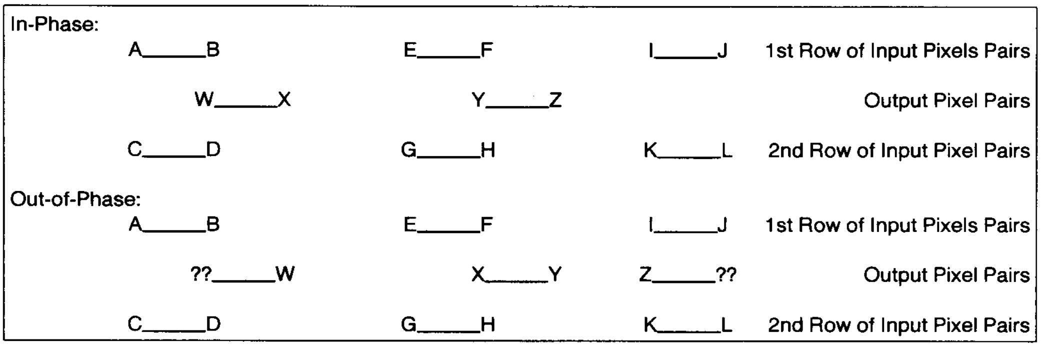


Figure 2-9. Pixel Pair Phases

Table 2-12. Pipelining Delay for Sequential-2D NON-PAIR Mode

Pipelining Bit (Bit 14)	Phase Bit (Bit 13)	Pipeline Delay in Output Pixels
0	0	6
0	1	7
1	0	2
1	1	3

When in PAIR mode (with bit 11 = one), the amount of pixel delay does not change, but half as many reads and writes are required to fill the pipeline because each read or write of the averager transfers two pixels. For example, when in the standard mode (bit 14 = 0), with zero phase (bit 13 = 0) and pair mode (bit 11 = 1), three indeterminate pixel pairs must be read before the first good pixel pair is read. In the same case but with the phase bit = 1, the fourth pixel pair read contains one good pixel and one indeterminate pixel, and the fifth pixel pair read contains two good pixels.

RESERVED

Bits 15 and 12 are reserved for future use. Write ZEROs into these bit positions.

Signature Register

{*hwid*}

The signature register can be read either by the host CPU or by microcode to determine the version of the 82750PB. The value of the signature register can be used to distinguish between the 82750PB in the 82750PA emulation mode, and the 82750PB in native mode. The currently defined signature values given in Table 2-13.

Table 2-13. Signature Values

Value	Definition
0xFFFE	The 82750PB Emulating the 82750PA
0XFFFC	The 82750PB in Native Mode

All other signature values are presently undefined but may be used in the future to denote other versions of the 82750 architecture.

Display Format Registers

{*yeven, yodd, vu, vptr*}

The 82750PB's processor can write to the display registers in the VRAM interface. These registers are pointers and pitch values that address display bitmaps and 82750DB register loads in VRAM. Pointers are 32-bit values that specify the specify the starting byte address of a bitmap or register load within a 4GByte address space. The bottom two address bits are ignored since display bitmaps and register loads must start on a double word boundary. Therefore, the internal representation of a pointer is a 30-bit value. The pitch value associated with each pointer indicates the number of bytes between the start of two lines of a display bitmap or between the start of two register loads. The pitch is a single 16-bit value with its two least significant bits ignored, since the pitch must be an integer number of double words. Currently, there is also a restriction in the 82750DB limiting all display bitmap pitches to powers of two; so, the maximum display bitmap pitch is $\pm 2^{14}$ Bytes = ± 16 kBytes. The display registers are described in Table 2-14.

Table 2-14. Display Registers

Register	Description
yeven-lo, hi	This register pair points to the start of the Y bitmap or main bitmap that is to be displayed during an even field scan.
yodd-lo, hi	This register pair points to the start of the Y bitmap or main bitmap that is to be displayed during the odd field scan.
ypitch	The value in this register is added to the current Y bitmap pointer value each time a Y transfer is performed.
vu-lo, hi	This register pair points to the start of the VU bitmap. This bitmap is read to generate the VU values for both odd and even field scans.
vupitch	This value is added to the current VU bitmap pointer value each time a VU transfer is performed.
vptr-lo, hi	This register pair points to the start of a series of 82750DB register loads stored in VRAM.
vpitch	This value is added to the current 82750DB register load pointer each time a 82750DB register load is performed. The pitch is equal to the number of bytes from the start of one register load to the start of the next register load.

3.0 HARDWARE INTERFACE

VRAM Interface

The VRAM interface performs the following operations:

- Maintains VRAM pointers for the two input FIFOs, the two output FIFOs, the statistical decoder, the Y (main) bitmap, the VU bitmap, and the 82750DB register load.
- Decodes VBUS codes and takes appropriate actions such as generating a transfer cycle, scheduling refresh cycles, or generating interrupt conditions.
- Arbitrates VRAM accesses between the two input FIFOs, the two output FIFOs, the statistical decoder, the transfer request logic, the VRAM refresh logic, and the external VRAM access logic.
- During a memory cycle, performs appropriate address arithmetic on the VRAM pointer used for that memory cycle.
- As a result of certain VBUS codes, performs a shadow copy that consists of copying display-related VRAM pointer values from shadow registers (that are loaded by the host CPU or the microcode processor) to working registers where the various pointers are used for transfer cycles when the 82750DB is refreshing the display screen.

Table 3-1. VRAM Interface Signals

Signal	Description
MREQ#	**MEMORY REQUEST** is asserted during the first cycle of a VRAM memory access.
TRNFR#	The **TRANSFER** output indicates the current memory cycle is a result of a 82750DB transfer request.
RFSM#	The **REFRESH** output indicates the current memory cycle is a result of a 82750DB refresh request.
NXTFST#	The **NEXT FAST** output indicates the next memory access will use the same row address as the current memory access. This facilitates the use of page mode memory accesses.
MRDY#	The **MEMORY READY** input indicates the availability of valid data on the D[31:0] pins.

VRAM ACCESSES

The 82750PB can initiate five different types of memory accesses: FIFO read, FIFO write, transfer read, transfer write, and refresh. In addition, the 82750PB supports VRAM accesses by external logic. During an external access VRAM cycle, the 82750PB tri-states its VRAM address and data buses and performs a host VRAM read or host VRAM write cycle. There is another operation performed by the 82750PB, a shadow copy, that is not a VRAM cycle but is arbitrated as though it were, since no VRAM cycles can take place during a shadow copy.

The seven types of VRAM cycles initiated by the 82750PB, including host VRAM read and host VRAM write, begin with the 82750PB asserting a combination of its three VRAM cycle definition outputs: TRNFR#, RFSH#, and WE#. External logic detects the state of these signals, validated by MREQ#, and produces the appropriate sequence of VRAM control signals (RAS, CAS, etc.) to perform the type of memory cycle the 82750PB has requested. The 82750PB requires that each of these VRAM cycles take a minimum of two T-cycles, or T-states, denoted T1 and T2. External logic can insert additional T2 states in order to stretch the VRAM cycle to more than two T-cycles. The start of a new VRAM access cycle is signaled by the assertion of MREQ# for the first T-cycle, T1. The VRAM access cycle definition signals, TRNFR#, RFSH#, and WE#, are asserted at the start of T1 and remain asserted until the end of the last T2. Other VRAM operations can be described similarly by sequences of T-states. Refer to Figures 3-4 and 3-5 on page 42 for timing diagrams.

Table 3-2 defines the states used for all VRAM access operations. A state diagram for the VRAM/Host Interface is provided in Figure 3-1. This diagram includes the FIFO access states

Table 3-2. 82750PB VRAM Access States

State	Description
Ti	Idle State, No VRAM Activity
T1, TF1	First State of a VRAM FIFO Cycle
T2, TF2	Last State of a VRAM FIFO Cycle
TSC	The T-State required to perform a shadow copy
TTX1	First State of a VRAM Transfer Cycle
TTX2	Last State of a VRAM Transfer Cycle
TRF1	First State of a VRAM Refresh Cycle
TRF2	Last State of a VRAM Refresh Cycle

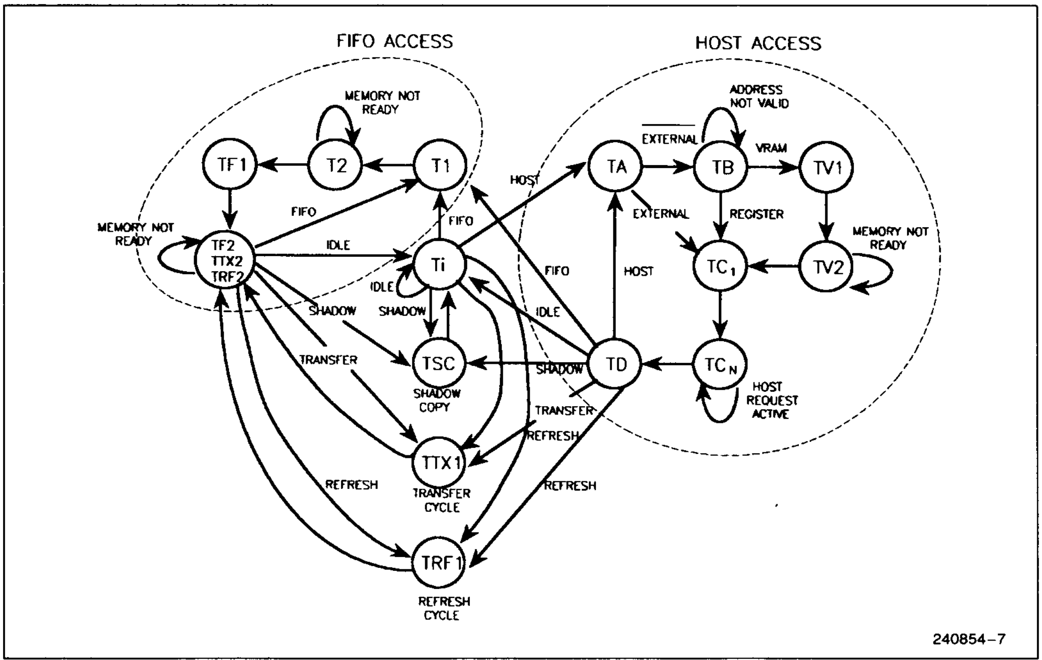


Figure 3-1. Access State Diagram

Note that during successive VRAM cycles it is not necessary to go back to the idle state, Ti, between each cycle; the T_{F2} state can be followed directly by a T1 state, starting at the next VRAM cycle. This results in efficient utilization of the 82750PB/VRAM bandwidth by allowing a VRAM cycle time of 2 T-states.

FAST VRAM CYCLES

When the 82750PB performs Data Read or Data Write VRAM cycles for the input or output FIFOs, it performs two 32-bit accesses to read or write one 64-bit value. These accesses are always performed in a sequence of EvenAddress followed by EvenAddress + 1, which guarantees both that the two sequential accesses will be in opposite banks and that the two accesses will be within the same VRAM page. This allows external logic to use either bank-interleaving or a page-mode access to complete the second access of the sequence and improve the VRAM bandwidth. However, the second access does not need to be handled differently from the first. Except for the assertion of the NXTFST# signal, both accesses are treated as standard VRAM accesses. External logic can ignore the NXTFST# signal, though, and treat the two accesses as two normal data read or data write cycles. Note that NXTFST# is not asserted for transfer, refresh, or host memory accesses.

The NXTFST# output signal is provided for cases when external logic can generate a faster access for the second access of the two sequential accesses. During such a pair of accesses, NXTFST# is asserted during the first of the two accesses in order to provide sufficient time for the external logic to generate the appropriate fast memory cycle for the second access. Refer to the timing diagrams in Figures 3-4 and 3-5 (page 42) for examples illustrating the use of the NXTFST# signal.

VBUS CODES

Transfer request, interrupt, and synchronization codes are sent over the BUS from the 82750DB to the 82750PB. The codes recognized by the 82750PB are listed in Table 3-3, along with the actions taken by the 82750PB as a result of receiving each code. Codes that cause TRANSFER cycles must be asserted for at least two clock cycles of the 82750PB to insure that, in the worst case, the 82750PB completes the transfer cycle before the code is released and the 82750DB starts shifting data from the VRAM shift registers. Other codes must also be asserted for a minimum of two 82750PB clock cycles. Only the codes given in the Table 3-3 are valid codes for the VBUS. Other codes are reserved for future use and should not be used. Once a transfer cycle code is sent to the 82750PB, any non-transfer code may be sent immediately. A subsequent transfer cycle code should be sent only after the current transfer cycle is completed.

Table 3-3. VBUS Codes

Binary	Name	Action
0000	YBMX	TXRD Cycle Using Yc; Yc = Yc + Yp*
0001	VUBMX	TXRD Cycle Using VUc; VUc = VUc + VUp
0010	REGX	TXRD Cycle Using Vc; Vc = Vc + Vp
0011	WRDIGX	TXWR Cycle Using Yc; Yc = Yc + Yp
0100	YNPBMX	TXRD Cycle Using Yc; Yc = Yc
0101	Reserved	Reserved
0110	Reserved	Reserved
0111	WRDIGNPX	TXWR Cycle Using Yc; Yc = Yc
1000	DFL	DFL Int; Shadow Copy**
1001	82750DBSD	82750DB Shutdown Interrupt
1010	REFRESH	Schedule N Refresh Cycles
1011	Reserved	Reserved
1100	VODD	VBI Int; OF Int; Shadow Copy Odd; Hline = 0***
1101	VEVEN	VBI Int; EF Int; Shadow Copy Even; Hline = 0***
1110	HLINE	lcnt+ + (Increment Line Counter)
1111	NULL	No Action

NOTES:
*Yc—Y bitmap pointer, current; Yp—Y bitmap pitch; VU—VU bitmap; V—82750DB register load.
**Shadow Copy with Yc = Y-start-odd in odd field; Yc = Y-start-even in even field.
***Hline—Horizontal Line Counter.

PRIORITY

Each time the VRAM state machine completes a VRAM operation and returns to the Ti state, it examines all pending VRAM access requests and selects the highest priority request for the next VRAM operation. The priority ordering of these requests are listed in Table 3-4.

Table 3-4. Priority of VRAM Operations

Request Type	Priority
Transfer Cycle	Highest
Shadow Copy	•
Host Access	•
VRAM Refresh	•
FIFO Read/Write	Lowest

NOTE:
The shadow copy is treated as a VRAM operation even though it does not result in an access to VRAM.

The VRAM refresh operation is placed low on the priority list to reduce the latency in servicing transfer requests and external VRAM requests. Since a single REFRESH code from the 82750DB schedules a number of refresh cycles, a higher priority for refresh would cause all the refresh cycles to occur in a burst that would lock out all lower priority requests until all refresh cycles completed. Instead, the following restriction applies to all request types with higher priority than refresh: high priority requests, such as transfer cycles, shadow copies, and external VRAM access must occur infrequently enough to allow proper refresh of the VRAM chips. Transfer cycles and shadow copies, by their nature, occur infrequently so they are not generally a problem.

There is a separate priority scheme for the five FIFO channels. The scheme used is rotating priority with automatic override and single cycle arbitration. Rotating priority means that the priority is assigned in a fixed cyclic order with the lowest priority given to the FIFO channel that "won" the last FIFO access. There is only one level of memory , so the order that requests arrive is not a factor in the arbitration. The cyclic order is given in Figure 3-2.

As an example, if input FIFO 0 (abbreviated if0) was the last channel to perform a cycle, the priority order for the next FIFO access (from highest to lowest) would be: if1, sd, of0, of1, and if0.

Automatic override is available so that the rotating cyclic priority can be bypassed if there is an URGENT condition for one of the channels. A channel is urgent if the microcode processor is frozen because the processor is waiting for that channel to be ready. The channel can be either an input channel that is empty or an output channel that is full. In this case, the urgent channel gets the next available cycle. However, the priority will still be lower than non-FIFO requests, such as refresh cycles.

Single clock cycle arbitration means that the selection of the next channel that will get an access occurs in a single T-cycle or T-state, either in a Ti state or during the last T2 state of the previous VRAM cycle.

VRAM POINTERS

The VRAM interface maintains VRAM pointers for the FIFOs, as well as display-related pointers for the 82750DB. Internally each pointer or address is stored as a 30-bit value addressing a double word in VRAM. The pointer values are read and written as two 16-bit words representing a 32-bit byte address (refer to the Figure 3-3). With a 30-bit double word address, the 82750PB can decode a VRAM address space of 1G double words or 4GBytes.

Input and output FIFOs can address down to a single word or byte in VRAM. A FIFO's pointer is post-incremented or post-decremented in parallel with its VRAM read or write cycle.

The statistical decoder can only start decoding bit-streams on double word boundaries in VRAM and can only increment through VRAM. The decoder's pointer is post-incremented in parallel with each of its VRAM read cycles.

Display-related pointers are updated by adding a pitch value to the current value during the corresponding transfer cycle.

If a VRAM pointer appears on the B-Bus as source or as a destination then the following rules apply:

Rule 1

If a B-Bus destination refers to an address that is both Even and >0x1f, then the source is restricted to "-lo" pointers if the source refers to a pointer.

Rule 2

If a B-Bus destination refers to an address that is both Odd and >0x1f, then the source is restricted to "-hi" pointers if the source refers to a pointer.

SHADOW COPY

When a VODD, VEVEN, or DFL code is received from the 82750DB over the VBUS, a shadow copy is scheduled. The actual shadow copy will occur as soon as the priority logic allows. Any VRAM access in progress must complete and a pending transfer cycle, if any, must be performed before the shadow copy can start. During the operation, shadow registers for the Y-START, Y-PITCH, VU-START, VU-PITCH, 82750DB-START, and 82750DB-PITCH are copied into the corresponding working registers. During display refresh, the address arithmetic is performed on the working registers. The shadow registers can be loaded by the host CPU or by a microcode routine with less critical timing constraints, and then copied instantly by a shadow copy when it is time to update the registers, either prior to the next field or during the active display for split screen effects.

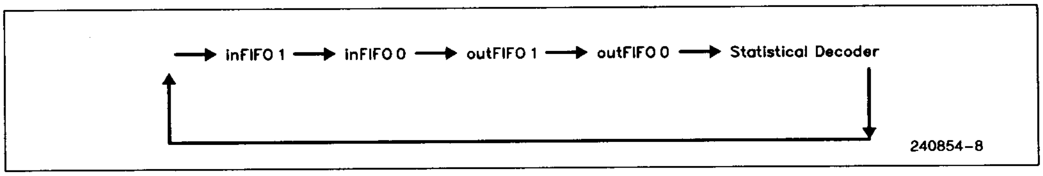


Figure 3-2. Cyclic Ordering of FIFOs

```
31  30  29       ..       24  23         ..          16  15           ..                3   2   1   0
<- - - - - - - - - - - - - - - - - - - VRAM  Address - - - - - - - - - - - - - - - - 30  bits - - - - - - - - - - - - >
                           Byte Address within Double-Word . . . . . . . . . . . . . . . . . . . . . . . <--->
<- Most Sig. Word of VRAM Address. ->                |          <- Least Sig. Wd. of VRAM Addr. ->
```

Figure 3-3. VRAM Addressing

There are actually two shadow registers for Y-START. One for start of odd fields and one for start of even fields. A VODD code causes Y-START-ODD to be copied into the working register Y-CURRENT. Similarly, a VEVEN code causes the Y-START-EVEN to be copied into Y-CURRENT. A DFL code causes the Y-START-ODD value to be copied if the most recent start of field code received is a VODD, or a Y-START-EVEN value if the most recent start of field code was a VEVEN. This allows a simple interlaced or non-interlaced display to be refreshed with no host CPU intervention. For more complex displays, such as split screens, the host CPU must update the shadow registers prior to each shadow copy. A shadow copy operation requires 2 T-cycles.

Host Interface

The Host Interface provides the following functions:

- Arbitrates host CPU and 82750PB access to VRAM.
- Provides the host access to external devices.
- Provides the host access to 82750PB internal registers and memories.

Signals specific to the Host Interface are listed in Table 3-5.

Table 3-5. Host Interface Signals

Signal	Description
HREQ#	**HOST REQUEST:** Asynchronous request from the host for all types of host access. Used both to request and release system buses.
HREG#	**HOST REGISTER:** Single-ranked control to request host access to 82750PB internal registers in concert with HRAM#.
HRAM#	**HOST VRAM:** Single-ranked control to request host access to VRAM in concert with HREG#.
HALEN#	**HOST ADDRESS LATCH ENABLE:** Asynchronous status from the host indicating the presence of valid address, write enable (transaction direction control), and the byte enables at the interface of the 82750PB.
HBUSEN#	**HOST BUS ENABLE:** 82750PB synchronous status granting the host access to the address, write enable, data bus, and byte enables at the interface of the 82750PB.
HRDY#	**HOST READY:** 82750PB synchronous status to the host indicating the presence of valid data appearing at the 82750PB's databus for VRAM and register accesses and optionally for external accesses.
HINT#	**HOST INTERRUPT:** 82750PB synchronous interrupt to the host, set under direct or indirect microprogram control.

Signals common to the host, VRAM, and external device interfaces are listed in Table 3-6.

Table 3-6. Host, VRAM, and External Device Interfaces

Signal	Description
A[31:2]	**ADDRESS BUS:** System address bus used to select unique VRAM, the 82750PB register, and external device locations that will be accessed under host control. The lower seven bits A[8:2] are bidirectional and are used during register accesses
D[31:0]	**DATA BUS:** Bidirectional system data bus used to transfer data to and from all sources and destinations. When transferring 16-bit host register values, the data bus MSH and LSH will both carry identical values.
WE#	**WRITE ENABLE:** Bidirectional, single-ranked signal used to determine the data transfer direction. When active during host register cycles, data flows from the host to an 82750PB destination. During host VRAM cycles, WE# active will define the data direction to be from the host to VRAM.
BE[3:0]#	**BYTE ENABLE:** Bidirectional signals used to select the bytes that will be modified during data transactions. All host register transactions are performed 16 bits at a time, while VRAM may be modified 8 bits at a time.

As with VRAM operations, host operations are described through a sequence of T-states. Table 3-7 defines the T-states used to implement all host transactions with VRAM, external devices, and the 82750PB.

The master execution state diagram that defines the VRAM/Host transactions is provided in Figure 3-1.

Table 3-7. 82750PB Host Transaction States

State	Description
TA	First state of any host transaction. Entry into TA will be granted after HREQ# has been asserted. During this state, the 82750PB will tri-state its address, data bus, write enable, and byte enable signals to provide a full cycle of "dead-band" before the assertion of HBUSEN#. In the state immediately following TA HBUSEN# will assert, allowing the host to drive the host buses.
TB	First cycle in which the host is granted bus access for register or VRAM transactions. The sequencer will remain in TB until HALEN# is received, indicating that the address write enable and byte enable signals are stable at the 82750PB pins.
TC1	First cycle that output data is valid.
TCn	This state is entered to wait for the completion of the current host cycle. The cycle is defined as complete when HREQ# deasserts. HRDY# is asserted along with valid data until the transition to state TD occurs.
TD	The last cycle of a host transaction. HBUSEN# is deasserted allowing one dead-band cycle to allow control of the address, data, write enable, and byte enable signals to be returned to the 82750PB.
TV1	First cycle of a Host VRAM transaction. Memory is requested and is followed by a transition to TV2.
TV2	Last cycle of a Host VRAM transaction. The sequencer will remain in TV2 until MRDY# is received.

A single stage of input synchronization is employed for HREG#, HRAM#, WE#, and BE[0]#, while HREQ# and HALEN# are programmable to have one or two stages by bit 12 of the Microcode Processor Control Register. See Table 3-10. T-state transitions are caused by the synchronized versions of these signals.

The synchronized versions of HREG# and HRAM# must be stable before entry into T-state TA. The synchronized versions of WE#, BE[0]#, and HALEN# should be stable before exiting T-State TB. Once asserted, all of the above signals should remain stable until the deassertion of HBUSEN#.

The type of host cycle to perform is determined by the states of HREG# and HRAM# as indicated in Table 3-8.

Table 3-8. Host Cycle Types

HREG#	HRAM#	Host Cycle Type
1	1	External
0	1	Register
1	0	VRAM
0	0	Reserved

HOST REGISTER ACCESS

The host has access to the 82750PB's internal registers and memories to monitor and control the operation of the microcode processor, provide a means of debugging microprogram routines, and to function as the primary test port for production testing.

Register access is initiated by the host asserting HREQ#, HREG#, and HRAM# as shown in Table 3-8 and in the timing diagrams on pages 42 through 45. After the host has been granted bus access by an active HBUSEN# in state TB, the address, write enable, and byte enables may be driven. After these signals have stabilized HALEN# is asserted, enabling a read or a write operation to occur.

NOTE:

Once HREQ# has been recognized by the 82750PB, a HBUSEN# will always be generated. HREQ# is recognized on the rising edge of TA, but note that it is only possible to know this AFTER state TA has been entered. Designs which need to "abort" requests must be prepared to ignore the possible HBUSEN#. Also, it is not possible in the general case to change the type of host request because HREG# and HRAM# are also recognized on the rising edge of TA.

In the case of a register read, state TC1 is entered and the data bus is driven with the internal value. One cycle later, a transition to state TC occurs, and HRDY# activates, signaling the presence of stabilized data at the 82750PB data pins. This state (TC) will be maintained until the host deasserts HREQ#, signaling the completion of the cycle that caused a transition to state TD.

In the case of a register write, TC1 is again entered (from TB), but the data bus may now be driven by the host. (During host cycles, data bus drive activity is indirectly controlled by WE# and an additional dead-band is provided by entry into state TC1 to allow for internal WE# stabilization.) Stable data at the 82750PB interface, as well as the completion of the write cycle, is signaled by the deassertion of HREQ#. As with reads, the deactivation of HRDY# signals the transition to state TD.

As state TD is entered, HRDY# and HBUSEN# deassert, the address data, write enable, and byte enables tri-state, and bus control is returned to the 82750PB in the following cycle.

HOST VRAM ACCESS

Because the 82750PB is so closely coupled with VRAM, host accesses to VRAM are arbitrated and controlled by the 82750PB. VRAM access is initiated by the host asserting HREQ#, HREG#, and HRAM# as shown in the Host Cycle Table above and in the Timing Diagrams on pages 38 through 45. After the host has been granted bus access by an active HBUSEN#, the address, write enable, and byte enables may then be driven. After these signals have stabilized at the memory devices (or longest relevant propagation path), HALEN# is asserted, enabling a read or write operation to occur.

Because VRAM will not drive the data bus until after a memory request, a transition into state TC1 to allow for data bus direction stabilization is not required. Instead, a transition to state TV1 occurs, which asserts MREQ# for a single cycle and is followed by a transition to TV2. TV2 will remain the current state until the reception of an active MRDY#.

In the case of a VRAM read, the memory data bus will be driven during TV1, and valid data will appear in state TV2. Data will be guaranteed valid coincident with the deassertion of MRDY# from memory.

In the case of a VRAM write, the memory data bus is driven with valid data during TV1. Again the reception of MRDY# will serve to indicate the completion of the memory operation.

NOTE:
The host device must be able to transmit or receive memory data in order to be valid at the trailing edge of MRDY# at the data's destination (memory or host).

After MRDY# becomes active, a transition from TV2 into TC1 is accomplished to allow time to propagate data to the host. TC is then entered to await the deassertion of HREQ# (if it has not already occurred). TD is then entered, duplicating the dead-banding previously described.

HOST EXTERNAL ACCESS

In addition to VRAM and register host access, an external device access mechanism is provided. During this access, upon the receipt of HREQ# with HREQ# and HRAM# inactive, the 82750PB releases the address, data, write enable, and byte enables in state TA.

The difference here is that state TC1 is directly entered from TA, thereby ignoring any transitions of HALEN#. Since the 82750PB also ignores the data bus direction control (write enable) the host and an external device may communicate unencumbered by the 82750PB.

Entry into state TC directly follows TC1 in the expected sequence and remains there until HREQ# is released. This is followed by entry into TD. HBUSEN# is asserted during the timing that TC1 and TCN are active.

During an external access, HRDY# is not asserted unless the external logic asserts MRDY# as shown in Figure 3-7.

HOST REGISTER ADDRESS MAPPING

Table 3-9 shows the host address mapping of the on-chip registers and memories, in terms of the offset in bytes, from the base address for 82750PB accesses. Note that the 82750PB only supports word accesses to these registers. Therefore, the least significant bit of the byte offset should be set to zero. The 82750PB forms the register address from inputs on the A[31:2] pins and BE#[3:0] pins. The A[31:2] specify the double word address of the register, and combinations of the BE# pins determine which of the two words with the double word is being addressed. BE#[3:0] = 1100_2 selects the least significant word within a double word, and BE#[3:0] = 0011_2 selects the most significant word within a double word. These are the only two valid patterns for BE# inputs during a host register access cycle.

Table 3-9. Host Address Mapping

Byte Address	Description
0x000–0x07E	(a) A source and destination registers
0x080–0x0FE	(b) B source and destination registers
0x100–0x17E	(c) Microcode processor control and status registers
0x180–0x1FE	(d) VRAM pointer RAM

NOTE:

The host should only perform 16-bit word reads or writes to 82750PB registers. The 82750PB does not support byte reads or writes or double word reads or writes to on-chip registers.

When the host CPU reads or writes to areas (a, b, or d) and the 82750PB is not already in a HALT state, the microcode processor is automatically HALTED for the one T-cycle actually required to complete the data transfer, and then the processor is restarted after the transfer is complete. If the 82750PB is in a HALT state when the host access is initiated, it will remain in the HALT state following the completion of the access. This is transparent to both the host CPU and the microcode processor.

During an access to areas (a) or (b), bits 6:1 of the byte offset should be set to the source or destination code for the register that will be read or written. The coding is the same as used in the microcode instruction word. Bit 0 is always set to a zero. Refer to the 82750PB Source and Destination Coding Table found in Chapter 4.

Area (c) contains one write-only register, the CONTROL register, and two read-only registers, the INTERRUPT FLAG register and the microcode PROCESSOR STATUS register. The CONTROL register is used to halt or single-step the microcode processor, which enables or masks interrupts to the host CPU, selects the signal that is output via the PMON/FRZ pin, and enables or disables the 82750PA emulation mode. The bit assignments for the CONTROL register are given in Table 3-10.

During reset of the 82750PB, the HALT bit is set to a one, the six Interrupt Enable bits are reset to zero, the Disable SYNC bit is set to zero, the PMON/FRZ bit is set to zero (so that the FRZ signal is output), and the Enable 82750PB bit is reset to zero (so that on reset, the 82750PB starts in a 82750PA emulation mode).

Table 3-10. Bit Assignments for Microcode Processor CONTROL Register {Write-Only, Byte Offset = 0x100}

Bit	Name	Description
Bit 0	HALT	1 = Microcode Processor Halt 0 = Microcode Processor Run
Bit 1	SINGLE-STEP	1 = Execute One Instruction and then Halt (Only when Already Halted, Bit 0 = 1) 0 = No Action
Bit 2	Enable MCINT	1 = Enable Microcode Interrupts to Host CPU 0 = Mask Microcode Interrupts
Bit 3	Enable VBI	1 = Enable Vertical Blanking Interrupt to Host CPU† 0 = Mask Vertical Blanking Interrupt
Bit 4	Enable DFL	1 = Enable DFL Interrupt to Host CPU 0 = Mask DFL Interrupt
Bit 5	Enable SD	1 = Enable 82750DB Shutdown Interrupt to Host 0 = Mask SD Interrupt
Bit 6	Enable OFI	1 = Enable Odd Field Interrupt† 0 = Mask OF Interrupt
Bit 7	Enable EFI	1 = Enable Even Field Interrupt† 0 = Mask EF Interrupt
Bits 8–11*		1 = RESERVED; Write as Zeros
Bit 12	Disable SYNC	1 = Disable Synchronizers for HREQ#/HALEN# 0 = Enable Synchronizers for HREQ#/HALEN#
Bit 13	PMON/FRZ	1 = Output FRZ# Signal on PMFRZ# Pin 0 = Output PMON# Signal on PMFRZ# Pin
Bit 14		1 = RESERVED; Write as Zero
Bit 15	Enable 82750PB	1 = Enable 82750PB Mode 0 = Enable 82750PA Emulation Mode

*All other bits are reserved for future use, and should be written as zeros.
†Only one of these bits should be set.
Bit 3, when set, enables an interrupt when either V_{ODD} or V_{EVEN} V_{BUS} codes are received.
Bit 6, when set, enables an interrupt when V_{ODD} V_{BUS} codes are received.
Bit 7, when set, enables an interrupt when V_{EVEN} V_{BUS} codes are received.

The INTERRUPT FLAG register holds a flag for each of the six interrupt sources. A flag bit is set to a one when the interrupt condition is detected (independent of the state of the corresponding Interrupt Enable/Mask bit in the CONTROL register), and all flags are cleared to zero each time the INTERRUPT FLAG register is read. If this register is read during the same cycle that an interrupt condition is detected, the flag bit corresponding to that interrupt condition will remain at a one. This new interrupt condition will then be seen by the host processor when it next reads the INTERRUPT FLAG register. The flag insures that an interrupt is not lost if it occurs at the same cycle that the INTERRUPT FLAG register is read (and reset). In addition, the Microcode Interrupt source has an overflow flag that indicates if more than one Microcode Interrupt has occurred since the Interrupt Flag register was last read. The bit assignments for the INTERRUPT FLAG register are listed in Table 3-11.

The PROCESSOR STATUS register holds four status bits: HALT, FREEZE, PMON, and SYNC status. HALT indicates that the processor is HALTED due to a HALT bit in the CONTROL register being set to a ONE or due to the HALT# pin being asserted. FREEZE indicates that the processor is waiting for one of the VRAM channels to become ready or is waiting for an access to the VRAM pointer RAM. PMON is a signal that can be toggled by a special ALU opcode or a special B source code. This signal can be used for performance monitoring of microcode. SYNC status bit indicates the presence or absence of the internal synchronizers for HREQ# and HALEN# inputs. In addition, the Interrupt Mask bits that are written into the PROCESSOR CONTROL register can be read from this register. These mask bits are read in the same polarity that they are written, but note that the bit positions and bit ordering are not consistent with the PROCESSOR CONTROL register. The bit assignments for this register are given in Table 3-12.

Address mapping for areas (a), (b), and (d) are given in Tables 3-13 to 3-15.

Table 3-11. Bit Assignments for INTERRUPT FLAG Register (Read-Only, Byte Offset = 0x100)

Bit	Description
Bit 8:0	Not Used, the State of These Bits Are Not Specified
Bit 9	EF Interrupt Flag
Bit 10	OF Interrupt Flag
Bit 11	MCINT Overflow Flag
Bit 12	82750DB Shutdown Interrupt
Bit 13	MCINT Microcode Interrupt
Bit 14	VBI Vertical Blanking Interrupt
Bit 15	DFL Display Format Load Interrupt

Table 3-12. Bit Assignments for PROCESSOR STATUS Register (Read-Only, Byte Offset = 0x102)

Bit	Description
Bit 0	HALT (1 = Halted, 0 = Running)
Bit 1	FREEZE (1 = Frozen, 0 = Running)
Bit 2	PMON (1 = Active, 0 = Inactive)
Bit 3	Synchronizers on HREQ#/HALEN# (0 = Enabled, 1 = Disabled)
Bit 9:4	Not Used, the State of These Bits is Not Specified
Bit 10	MCINT Microcode Interrupt Mask
Bit 11	VBI Vertical Blanking Interrupt Mask
Bit 12	DFL Display Format Load Interrupt Mask
Bit 13	82750DB Shutdown Interrupt Mask
Bit 14	OF Interrupt Mask
Bit 15	EF Interrupt Mask

Table 3-13. 82750PB A Bus Source/Destination Address Mapping

Address (Hex)	ADST	ASRC
0x000	Null	Null
0x002		hwid
0x004		cc
0x006	maddr	
0x008		alu
0x00A	cnt	cnt
0x00C	cnt2	cnt2
0x00E	lcnt	lcnt
0x010	r0	r0
0x012	r1	r1
0x014	r2	r2
0x016	r3	r3
0x018	r4	r4
0x01A	r5	r5
0x01C	r6	r6
0x01E	r7	r7
0x020	mcode3	mcode3
0x022	mcode2	mcode2
0x024	mcode1	mcode1
0x026	pc	pc
0x028	pixint-c	
0x02A	pixint	pixint
0x02C	*dram1	*dram1
0x02E	*dram2	*dram2
0x030	*dram1 + +	*dram1 + +
0x032	*dram2 + +	*dram2 + +
0x034	*dram1 − −	*dram1 − −
0x036	*dram2 − −	*dram2 − −
0x038	dram1	dram1
0x03A	dram2	dram2
0x03C	dram3	dram3
0x03E	dram4	dram4
0x040	*out1	*in1
0x042	out1 + +	*in2
0x044	shift-hi	*stat
0x046	out1-hi	*stat#
0x048	*out2	
0x04A	out2 + +	
0x04C	shift-r	
0x04E	out2-hi	
0x050	out1-c	
0x052	in1-c	
0x054	shift-l	
0x056	in1-hi	
0x058	out2-c	
0x05A	in2-c	
0x05C		
0x05E	in2-hi	
0x060	r8	r8
0x062	r9	r9
0x064	r10	r10
0x066	r11	r11
0x068	r12	r12
0x06A	r13	r13
0x06C	r14	r14
0x06E	r15	r15
0x070	cc	shift
0x072	fcnt	fcnt
0x074	*dram3	*dram3
0x076	*dram4	*dram4
0x078	*dram3 + +	*dram3 + +
0x07A	*dram4 + +	*dram4 + +
0x07C	*dram3 − −	*dram3 − −
0x07E	*dram4 − −	*dram4 − −

Table 3-14. 82750PB B Bus Source/Destination Address Mapping

Address (Hex)	BDST	BSRC
0x080	Null	Null
0x082		alu
0x084	*dram3	*dram3
0x086	*dram4	*dram4
0x088	*dram3 + +	*dram3 + +
0x08A	*dram4 + +	*dram4 + +
0x08C	*dram3 – –	*dram3 – –
0x08E	*dram4 – –	*dram4 – –
0x090	r0	r0
0x092	r1	r1
0x094	r2	r2
0x096	r3	r3
0x098	r4	r4
0x09A	r5	r5
0x09C	r6	r6
0x09E	r7	r7
0x0A0	r8	*in1
0x0A2	r9	*in1
0x0A4	r10	*stat
0x0A6	r11	*stat#
0x0A8	r12	circbuf
0x0AA	r13	
0x0AC	r14	
0x0AE	r15	
0x0B0	circbuf	literal 0
0x0B2		literal 1
0x0B4	*dram1	literal 2
0x0B6	*dram2	literal 3
0x0B8	*dram1 + +	literal 4
0x0BA	*dram2 + +	literal 5
0x0BC	*dram1 – –	literal 6
0x0BE	*dram2 – –	literal 7
0x0C0	*out1	prof
0x0C2	out1 + +	
0x0C4	out1-lo	out1-lo
0x0C6	out1-hi	out1-hi
0x0C8	*out2	stat-lo
0x0CA	out2 + +	stat-hi
0x0CC	out2-lo	out2-lo
0x0CE	out2-hi	out2-hi
0x0DO	out1-c	out1-c
0x0D2	in1-c	in1-c
0x0D4	in1-lo	in1-lo
0x0D6	in1-hi	in1-hi
0x0D8	out2-c	out2-c
0x0DA	in2-c	in2-c
0x0DC	in2-lo	in2-lo
0x0DE	in2-hi	in2-hi
0x0E0	stat-ram	r8
0x0E2	stat-c	r9
0x0E4	stat-lo	r10
0x0E6	stat-hi	r11
0x0E8	yeven-lo	r12
0x0EA	yeven-hi	r13
0x0EC	yodd-lo	r14
0x0EE	yodd-hi	r15
0x0F0	ypitch	shift
0x0F2		stat-c
0x0F4	vu-lo	*dram1
0x0F6	vu-hi	*dram2
0x0F8	vupitch	*dram1 + +
0x0FA	vpitch	*dram2 + +
0x0FC	vptr-lo	*dram1 – –
0x0FE	vptr-hi	*dram2 – –

1

Table 3-15. VRAM Pointer RAM Mapping

Byte Address	Name	Description
0x180 0x182	Yw-lo Yw-hi	Working Copy of Y Pointer
0x184 0x186	out1-lo out1-hi	Output FIFO 1 Pointer
0x188	Yw-pitch	Working Copy of Y Pitch
0x18A		RESERVED
0x18C 0x18E	out2-lo out2-hi	Output FIFO 2 Pointer
0x190 0x192	VUw-lo VUw-hi	Working Copy of VU Pointer
0x194 0x196	in1-lo in1-hi	Input FIFO 1 Pointer
0x198	VUpitchw	Working Copy of VU Pitch
0x19A	vpitchw	Working Copy of 82750DB Pitch
0x19C 0x19E	in2-lo in2-hi	Input FIFO 2 Pointer
0x1A0 0x1A2	vptrw-lo vptrw-hi	Working Copy of 82750DB Pointer
0x1A4 0x1A6	stat-lo stat-hi	Working Copy of Statistical Decoder Pointer
0x1A8 0x1AA	Yeven-lo Yeven-hi	Shadow Copy of Y Start Even Pointer
0x1AC 0x1AE	Yodd-lo Yodd-hi	Shadow Copy of Y Start Odd Pointer
0x1B0	Ypitch	Shadow Copy of Y Pitch
0x1B2	rfcnt	RFSH Cycles per RFSH Code from 82750DB
0x1B4 0x1B6	VU-lo VU-hi	Shadow Copy of VU Start Pointer
0x1B8	VUpitch	Shadow Copy of VU Pitch
0x1BA	vpitch	Shadow Copy of 82750DB Pitch
0x1BC 0x1BE	vptr-lo vptr-hi	Shadow Copy of 82750DB Pointer

NOTE:
Register rfcnt is a write-only register and should never be read.

Initializing the 82750PB

The 82750PB is placed in a RESET state by asserting RESET# for at least ten T-cycles. In the RESET state, which continues until RESET# is released, all of the 82750PB's outputs are tri-stated for compatibility with board test requirements.

Proper initialization of the 82750PB requires that the 82750PB is held in a RESET state by keeping RESET# active for at least 10 T-cycles, and then releasing RESET#. This is referred to as the INITIAL state. In the INITIAL state:

- The microcode processor is halted.
- All six interrupts are masked, and the interrupt latches are cleared.
- The 82750PA/82750PB instruction format select bit is set to the 82750PA.
- The VRAM interface is ready to service VRAM requests; however, none of the VRAM pointers are valid.

- The number of refresh cycles that will be generated each time a RFSH code is received from the 82750DB is set to 14 cycles.
- All bidirectional I/O pins are tri-stated.

After the 82750PB has been initialized, i.e., placed in the INITIAL state, but prior to releasing the 82750DB's reset signal, the following operations must be performed:

- Load the REFRESH-CYCLES-PER-LINE register with the appropriate value (the equation for the value is: VALUE = ($2^N - 1$), where N is the number of cycles; for example, 5 refresh cycles would result in VALUE = $2^5 - 1 = 31_{10} = 001F_{16}$. The refresh register is 14 bits wide and the way it works is to generate one refresh every time a right shift results in a '1' bit. It continues the right sifting until it finds a '0' bit and halts. Hence from programming point of view: $001F_{16} = FFDF_{16} =$ 5 refresh cycles per line.
- Load the shadow copies of Y, VU, and 82750DB pointers and pitches.
- Load the appropriate 82750DB Register Load list into VRAM starting at the address pointed to by the 82750DB pointer.

Prior to releasing the microcode processor from its HALTed state to run a microcode program, the following operations must be performed:

- If 82750PB code is to be executed, bit 15 of the 82750PB CONTROL register must be set to a one.
- Load a microcode program into microcode RAM on the 82750PB by writing to the three instruction word registers (*mcode1* — the most significant word of the instruction, *mcode2,* and *mcode3* — the least significant word of the instruction, the one containing the next address field) and then writing to *maddr,* the address in microcode RAM where the instruction will be loaded.
- Load the PC with the address in microcode RAM of the first instruction to be executed.
- Write to the 82750PB CONTROL register with the HALT bit (bit 0) set to zero, causing the processor to start executing an instruction sequence, or with the SINGLE-STEP bit (bit 1) set to a one (keeping HALT also set to one), causing the processor to execute a single instruction.

Performance Monitoring

Two signals, FRZ# and PMON#, which are useful for microcode performance monitoring, are available both as external signals, multiplexed on a single output pin, and as bits in the Processor Status register. FRZ# is active for each T-cycle when the microcode processor is frozen, waiting for access to VRAM or to the VRAM Pointer RAM. PMON# can be toggled by a special ALU opcode or a special B bus source code. This allows PMON# to be used to indicate what particular segment of microcode is being executed. The PMON/FRZ bit in the Processor Control register selects the signal that is being output.

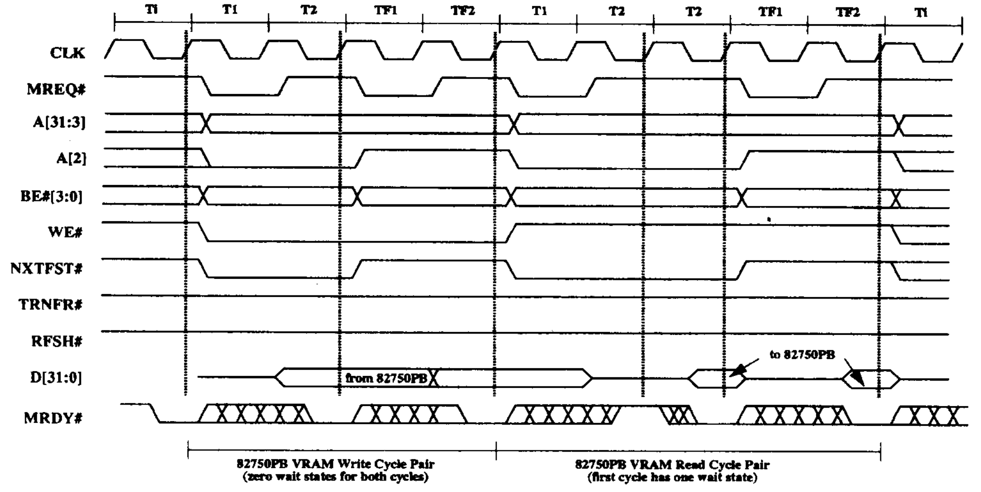
Freezes may indicate that the microcode routine is not making the most efficient use of the input and output FIFO buffering. This is particularly important for the inner loops of graphics and video routines that are memory-bandwidth limited. Ideally, inner loops should be balanced so that the rate pixels are processed is equal to the rate that they can be read from and written to VRAM with no freezes. The buffering in the input and output FIFOs serve to make sequential reads and writes to VRAM more efficient by performing full 64-bit reads and writes, instead of individual 8-bit or 16-bit accesses. This has the effect of averaging the VRAM read/write rate over a number of instruction times. For example, if the 82750PB is performing a 64-bit read or write every 8 T-cycles, for an average of 8 bits per T-cycle, a two instruction inner loop could read one 8-bit pixel and write one 8-bit pixel without any freezes occurring (assuming the source pixels and the destination pixels are each sequential).

The PMON# provides a more standard performance monitoring capability by indicating when a particular segment of microcode, bracketed by special instructions that toggle the PMON# signal, is being executed. This allows either absolute execution-time measurement or measurement of the fraction of the total execution time that is required by the segment. Either the ALU opcode "prof" or the B bus source code "prof" will toggle the PMON signal.

An external HALT pin is provided on the 82750PB to allow external debugging hardware to immediately halt the microcode processor. Activating this input causes the microcode processor to halt prior to executing the next instruction. When the processor is halted, the VRAM interface portion of the 82750PB continues to operate normally, performing transfer cycles, refresh cycles, and shadow copies as requested by the 82750DB.

Host/VRAM Timing Diagrams

Figures 3-4 through 3-8 are Host/VRAM Timing Diagrams.

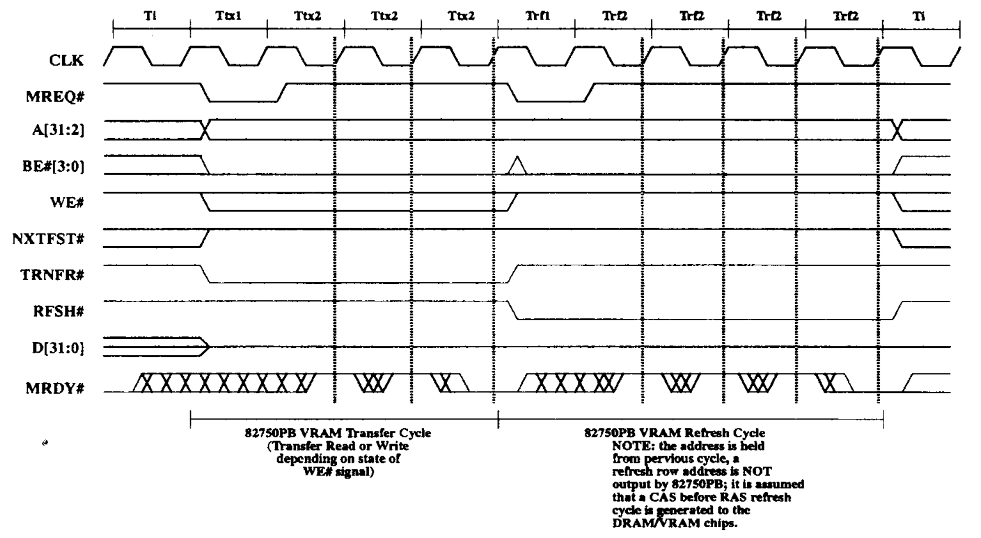


Ti
T1
T2
TF1
TF2
T1
T2
T2
TF1
TF2
Ti
CLK
MREQ#
A[31:3]
A[2]
BE#[3:0]
WE#
NXTFST#
TRNFR#
RFSH#
D[31:0]
MRDY#
from 82750PB
to 82750PB
82750PB VRAM Write Cycle Pair (zero wait states for both cycles)
82750PB VRAM Read Cycle Pair (first cycle has one wait state)

240854-9

NOTES:
1. Address pin A[2] is always ZERO for the first cycle of a cycle pair and ONE for the second cycle.
2. The two cycles of a cycle pair are both writes or both reads.
3. Beware that the address changes (A[2] and BE[3:0]) on state TF1. This is one cycle after the data changes on a write cycle.
4. The minimum number of cycles for a VRAM read of two words is 4, the sequence illustrated shows one extra T2.

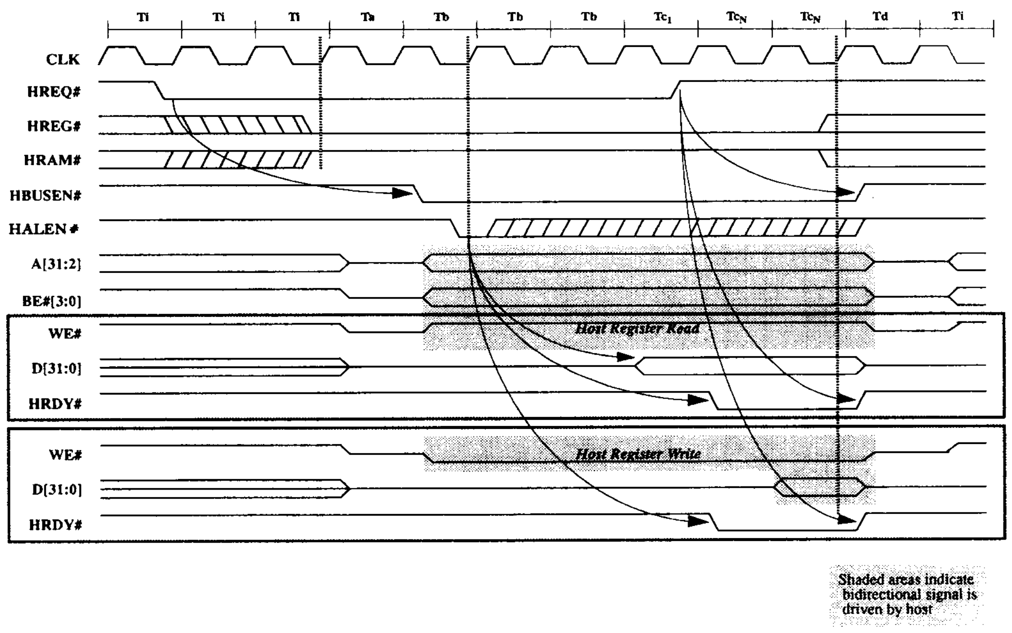
Figure 3-4 VRAM Read and Write Cycles



Ti
Ttx1
Ttx2
Ttx2
Ttx2
Trf1
Trf2
Trf2
Trf2
Trf2
Ti
CLK
MREQ#
A[31:2]
BE#[3:0]
WE#
NXTFST#
TRNFR#
RFSH#
D[31:0]
MRDY#
82750PB VRAM Transfer Cycle (Transfer Read or Write depending on state of WE# signal)
82750PB VRAM Refresh Cycle
NOTE: the address is held from pervious cycle, a refresh row address is NOT output by 82750PB; it is assumed that a CAS before RAS refresh cycle is generated to the DRAM/VRAM chips.

240854-10

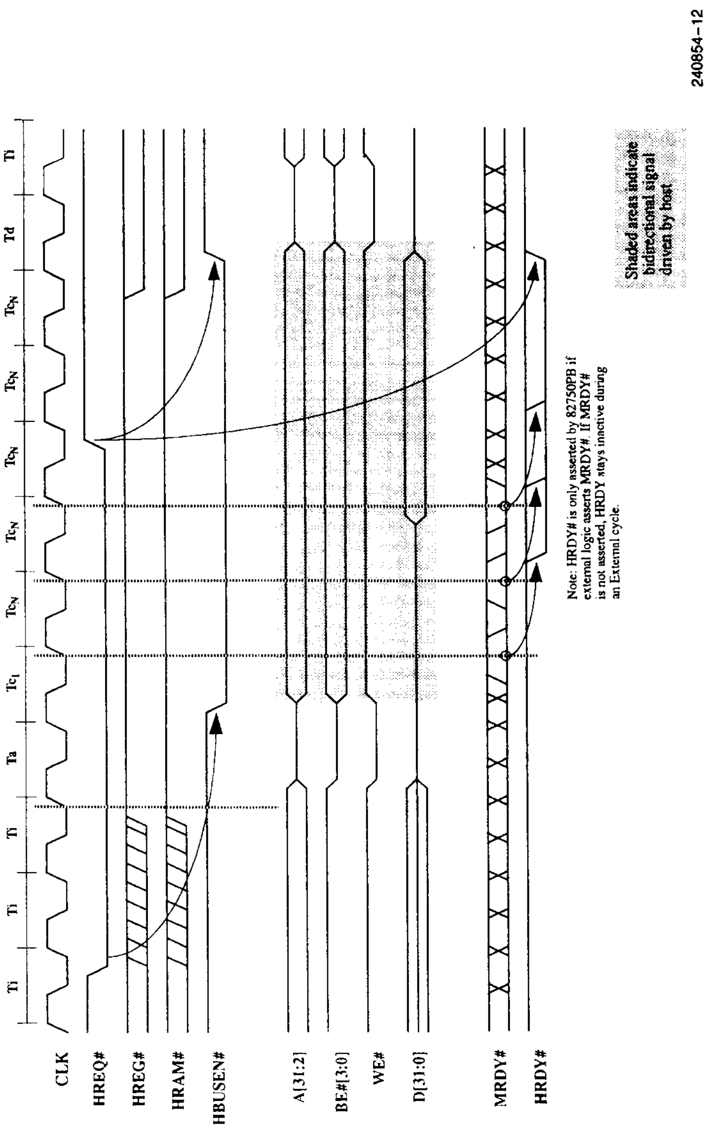
Figure 3-5. VRAM Transfer and Refresh Cycles



240854-11

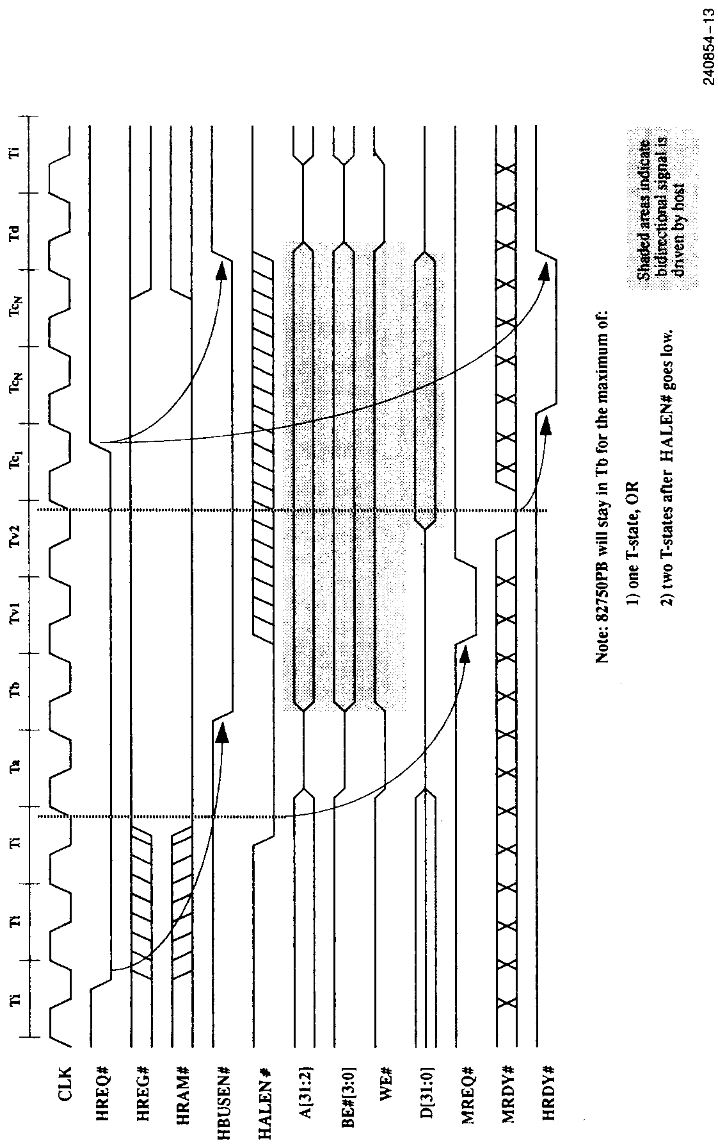
NOTES:
1. MREQ#, RFSH#, TRNFR#, and NXTFST# remain inactive during Host Register Read and Write cycles.
2. If HALEN#/HREQ# synchronizers are disabled then the second Ti and Tb states will be missing.
3. Please see note on page 32.

Figure 3-6. Host Register Read and Write Cycles


Ti
Ti
Ti
Ta
Tc1
TcN
TcN
TcN
TcN
TcN
Td
Ti
CLK
HREQ#
HREG#
HRAM#
HBUSEN#
A[31:2]
BE#[3:0]
WE#
D[31:0]
MRDY#
HRDY#
Note: HRDY# is only asserted by 82750PB if external logic asserts MRDY#. If MRDY# is not asserted, HRDY stays inactive during an External cycle.
Shaded areas indicate bidirectional signal driven by host
240854-12

NOTES:
1. MREQ#, RFSH# TRNFR#, and NXTFST# remain inactive during Host External Read and Write cycles.
2. If the Synchronizer on HREQ# is disabled, then the second Ti state will be missing.
3. Please see note on page 32.

Figure 3-7. Host External Read and Write Cycles


Ti
Ti
Ti
Ta
Tb
Tv1
Tv2
Tc1
TcN
TcN
Td
Ti
CLK
HREQ#
HREG#
HRAM#
HBUSEN#
HALEN#
A[31:2]
BE#[3:0]
WE#
D[31:0]
MREQ#
MRDY#
HRDY#
Note: 82750PB will stay in Tb for the maximum of:
1) one T-state, OR
2) two T-states after HALEN# goes low.
Shaded areas indicate bidirectional signal is driven by host
240854-13

NOTES:
1. RFSH#, TRNFR#, and NXTFST# remain inactive during Host VRAM Read and Write cycles.
2. If the Synchronizers on HREQ#/HALEN# is disabled, then the second Ti state will be missing.
3. Please see note on page 32.

Figure 3-8. Host VRAM Read and Write Cycles

4.0 MICROCODE INSTRUCTION FORMAT

Overview

The 82750PB executes two slightly different instruction formats: one that is backward compatible with the 82750PA and another that allows full access to the microcode resources of the 82750PB. The 82750PA/82750PB bit in the 82750PB processor control register determines which instruction format is in effect (see Chapter 3). On reset, the 82750PB is placed in 82750PA instruction format mode. In this mode the 82750PB will execute binary microcode originally assembled for the 82750PA in a manner that is functionally equivalent to the 82750PA.

The following description applies to the 82750PB instruction format. Exact definitions of 82750PB instruction formats and field codings are shown in Figure 4-2 and Table 4-5.

Instruction Sequencing

The instruction word for 82750PB's microcode processor is 48 bits wide. The Microcode RAM holds 512 instructions. Nine bits of each instruction specify the address of the next instruction to be executed. Each instruction fetch reads two instructions (an odd address and even address pair) using the upper eight bits of the 9-bit instruction address. Both the LSB of the instruction address and a Condition Flag bit, selected from eight possible branching conditions, are used to determine whether the next instruction to be executed is the even address instruction or odd address instruction, according to the logic table shown as Table 4-1.

Table 4-1. Microcode Next Instruction Selection

LSB of Address	Condition Flag State	Next Instruction
0	0 (FALSE)	EVEN
0	1 (TRUE)	EVEN
1	0 (FALSE)	ODD
1	1 (TRUE)	EVEN

For an unconditional branch, the condition flag FALSE (which is always zero) is selected; this causes the LSB of the address to be passed through to select the next instruction: LSB = 0 selects EVEN and LSB = 1 selects ODD. This allows unconditional branching to any of the 512 instructions in the RAM. For a conditional branch, the LSB of the address is set to a one; this causes the state of the condition flag to select the next instruction: FALSE selects the ODD instruction and TRUE selects the EVEN instruction. Therefore, a conditional branch jumps to either the odd or even instruction of an odd/even pair depending on the state of the condition.

Instruction Word Field Descriptions

Each field of the microcode instruction format is described in the following sections.

NADDR—NEXT INSTRUCTION ADDRESS FIELD

This field holds the address of the next instruction to be executed. Taking advantage of the fact that the microcode RAM is physically organized as 256 deep by 96 wide (two instructions are fetched per read cycle), a zero delay two-way branch can be achieved. The only case in which this field is not used to determine the address of the next instruction to be executed is when an instruction writes to the PC. (The term PC refers to the register that holds the address of the next instruction to be executed.) When an instruction loads the PC a one instruction delay occurs before the load takes effect. Therefore, the instruction pointed to by the next instruction field of the instruction that loads the PC is executed before the jump to the new address occurs. This is shown in Table 4-2.

There are no restrictions on the instruction following a PC load; it will always be executed, even while single stepping the processor, or if the processor is frozen on that instruction.

CFSEL—CONDITION FLAG SELECT FIELD

This field selects which condition flag will be used with the LSB of NADDR to select the next instruction from the odd/even pair. The condition flag assignment is given in Table 4-3.

Table 4-2. PC Load Example

Addr	Instruction	NADDR	Comments
10	pc = 0	55	Load PC with zero.
55	r0 = 1	X	This instruction is executed but its next address field is ignored.
0	r1 = r0	25	PC load takes effect after a one instructon delay, the result is that r1 = r0 = 1.

Table 4-3. Condition Flag Select Field Assignments

Value	Flag	Description
000	FALSE	Select for Unconditional Branch
001	CARRY	Carry Out from ALU Condition Flag Latch
010	OVF	Overflow from ALU Condition Flag Latch
011	SIGN	Sign from ALU Condition Flag Latch
100	ZERO	Zero from ALU Condition Flag Latch
101	LCNTZ	TRUE if Selected Loop Counter = 0
110	LSB	LSB of Data Register r0
111	MSB	MSB of Data Register r0

NOTE:
The ALU condition flags (CARRY, OVF, SIGN, and ZERO) are latched in the ALU Condition Flag register. This register is updated for most—but not all—ALU operations. The remaining flags (LCNTZ, LSB, and MSB) are updated and latched each cycle.

ASRC—A BUS SOURCE SELECT FIELD

This field selects the element that should drive its data onto the A bus during the execution of this instruction. The mapping for this and the following three fields is provided in Chapter 6.

ADST—A BUS DESTINATION SELECT FIELD

This field selects which element should latch data from the A bus during the execution of this instruction. See ASRC above.

BSRC—B BUS SOURCE SELECT FIELD

Same as ASRC, but for B bus. See ASRC above.

BDST—B BUS DESTINATION SELECT FIELD

Same as ADST, but for B bus. See ADST above.

CNT—DECREMENT LOOP COUNTER BIT

A one in this bit position causes the selected Loop Counter (selected by LC, the loop counter select bit) to be decremented. The new value of the loop counter and the updated LCNTZ condition flag are not ready until the next instruction cycle. Therefore, in a loop where the loop counter is decremented and tested for zero in the same instruction (typically in a one instruction loop), the start value for the loop counter should be one less than the number of times the loop should be executed.

LIT—LITERAL SELECT BIT

When this bit is a one, the ASRC and CFSEL fields are replaced with a 9-bit literal value that is driven as a source in the least significant 9 bits of the A bus. In this case, the upper 7 bits of the A bus are forced to zeros. The mapping of bits from the literal field to the A bus is shown in Figure 4-1.

NOTE:

A conditional branch and a literal on the A bus are not allowed in the same instruction. A 3-bit literal can be placed on the B bus in any instruction.

A bus bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Inst. Word Bits	← ----- Forced to Zero ---- →							17	16	15	14	13	12	11	10	9
ASRC Field								← ----------------- →								
CFSEL Field														← -- →		

Figure 4-1. Literal Field Mapping onto a Bus

SHFT—SHIFT CONTROL FIELD

This field controls the bit shifting and byte swapping logic associated with register *r0*. The encoding of this field is given in Table 4-4.

Table 4-4. SHIFT Control Field Coding

SHFT	Operation
00	No Shift or Swap Operation
01	Shift r0 Right One Bit Position, Sign Extend
10	Shift r0 Left One Bit Position, Zero Fill
11	Byte Swap the Value Being Loaded into r0*

*Byte swapping only works when r0 is the destination on the A bus or the B bus. It does not swap data held in r0, only data being loaded. In order to byte swap data in register r0, r0 must be both a source and destination for either the A or B bus.

ALUSS—ALU SOURCE SELECT BITS

These two bits are used as enables for the two ALU input latches. Bit 39 enables the latch that connects to the A bus; bit 38 enables the latch connected to the B bus. A one in either bit position causes the corresponding input latch to latch the value on the bus to which it is connected (the A or B bus). A zero on either bit causes the corresponding latch to hold its current content. This allows the ALU operands either to come from "eavesdropping" on the A or B bus transfers occurring in the current instruction cycle or to be held for multiple instruction cycles in either the A or B input latch.

ALUOP—ALU OPERATION CODE FIELD

This field specifies the ALU instruction to be performed during the current instruction cycle. The encoding of this field is given in Figure 4-2. Normally, at the end of the instruction execution, the result of the ALU operation is latched in the ALU output latch that can be a source on either the A or B buses. However, if a NOP is selected for the ALU operation, the ALU output latch is not latched. The data is held from the previous instruction. In addition to NOP, certain other ALU opcodes do not actually perform ALU operations and therefore, do not latch the ALU results. They are INT (microcode interrupt) and the PROF instruction.

LC—LOOP COUNTER SELECT BIT

This bit selects which of the two loop counters is to be used for decrementing or Loop-Counter-Zero conditional branching in the current instruction. A zero selects loop counter zero and a one selects loop counter one.

Refer to the Intel *82750PB Microcode Programming Guide* for more information on microcode programming, order #466718-001.

Table 4-5. 82750PB Source/Destination Coding

Address (Hex)	BDST	BSRC	ADST	ASRC
0x0	Null	Null	Null	Null
0x1		alu		hwid
0x2	*dram3	*dram3		cc
0x3	*dram4	*dram4	maddr	
0x4	*dram3++	*dram3++		alu
0x5	*dram4++	*dram4++	cnt	cnt
0x6	*dram3−−	*dram3−−	cnt2	cnt2
0x7	*dram4−−	*dram4−−	lcnt	lcnt
0x8	r0	r0	r0	r0
0x9	r1	r1	r1	r1
0xA	r2	r2	r2	r2
0xB	r3	r3	r3	r3
0xC	r4	r4	r4	r4
0xD	r5	r5	r5	r5
0xE	r6	r6	r6	r6
0xF	r7	r7	r7	r7
0x10	r8	*in1	mcode3	mcode3
0x11	r9	*in2	mcode2	mcode2
0x12	r10	*stat	mcode1	mcode1
0x13	r11	*stat#	pc	pc
0x14	r12	circbuf	pixint-c	
0x15	r13		pixint	pixint
0x16	r14		*dram1	*dram1
0x17	r15		*dram2	*dram2
0x18	circbuf	literal 0	*dram1++	*dram1++
0x19		literal 1	*dram2++	*dram2++
0x1A	*dram1	literal 2	*dram1−−	*dram1−−
0x1B	*dram2	literal 3	*dram2−−	*dram2−−
0x1C	*dram1++	literal 4	dram1	dram1
0x1D	*dram2++	literal 5	dram2	dram2
0x1E	*dram1−−	literal 6	dram3	dram3
0x1F	*dram2−−	literal 7	dram4	dram4
0x20	*out1	prof	*out1	*in1

Table 4-5. 82750PB Source/Destination Coding (Continued)

Address (Hex)	BDST	BSRC	ADST	ASRC
0x21	out1++		out1++	*in2
0x22	out1-lo	out1-lo	shift-rl	*stat
0x23	out1-hi	out1-hi	out1-hi	*stat#
0x24	*out2	stat-lo	*out2	
0x25	out2++	stat-hi	out2++	
0x26	out2-lo	out2-lo	shift-r	
0x27	out2-hi	out2-hi	out2-hi	
0x28	out1-c	out1-c	out1-c	
0x29	in1-c	in1-c	in1-c	
0x2A	in1-lo	in1-lo	shift-l	
0x2B	in1-hi	in1-hi	in1-hi	
0x2C	out2-c	out2-c	out2-c	
0x2D	in2-c	in2-c	in2-c	
0x2E	in2-lo	in2-lo		
0x2F	in2-hi	in2-hi	in2-hi	
0x30	stat-ram	r8	r8	r8
0x31	stat-c	r9	r9	r9
0x32	stat-lo	r10	r10	r10
0x33	stat-hi	r11	r11	r11
0x34	yeven-lo	r12	r12	r12
0x35	yeven-hi	r13	r13	r13
0x36	yodd-lo	r14	r14	r14
0x37	yodd-hi	r15	r15	r15
0x38	ypitch	shift	cc	shift
0x39		stat-c	fcnt	fcnt
0x3A	vu-lo	*dram1	*dram3	*dram3
0x3B	vu-hi	*dram2	*dram4	*dram4
0x3C	vupitch	*dram1++	*dram3++	*dram3++
0x3D	vpitch	*dram2++	*dram4++	*dram4++
0x3E	vptr-lo	*dram1−−	*dram3−−	*dram3−−
0x3F	vptr-hi	*dram2−−	*dram4−−	*dram4−−

bit coding	LC SEL	SHFT CNTL	ALU OPCODE	ALU SS	LIT	CNT	B Bus Destination	B Bus Source
	47	46 45	44 43 42 41 40	39 38	37	36	35 34 33 32 31 30	29 28 27 26 25 24
			mcode 1					mcode2
	15	14 13	12 11 10 9 8	7 6	5	4	3 2 1 0 15 14	13 12 11 10 9 8
	1	2	5	2	1	1	6	6
0x0	cnt	nop	NOP	hold	nop	nop	null	null
0x1	cnt2	shft r	ZERO	lat b	lit	dec		alu
0x2		shft l	a	lat a			*dram3	*dram3
0x3		swap	b	both			*dram4	*dram4
0x4			~ a				*dram3 + +	*dram3 + +
0x5			~ b				*dram4 + +	*dram4 + +
0x6			&				*dram3 − −	*dram3 − −
0x7			~ &				*dram4 − −	*dram4 − −
0x8			& ~				r0	r0
0x9			+ +				r1	r1
0xA			|				r2	r2
0xB			~ |				r3	r3
0xC			| ~				r4	r4
0xD			− <				r5	r5
0xE			−				r6	r6
0xF			− + <				r7	r7
0x10			+				r8	*in1
0x11			−				r9	*in2
0x12			− +				r10	*stat
0x13			− a				r11	*stat #
0x14			− b				r12	circbuf
0x15			a + +				r13	
0x16			b + +				r14	
0x17			a − −				r15	
0x18			b − −				circbuf	literal 0
0x19			int					literal 1
0x1A			prof				*dram1	literal 2
0x1B			a*				*dram2	literal 3
0x1C			b*				*dram1 + +	literal 4
0x1D			+ <				*dram2 + +	literal 5
0x1E			+]				*dram1 − −	literal 6
0x1F			−]				*dram2 − −	literal 7
0x20							*out1	prof
0x21							out1 + +	
0x22							out1 − lo	out1 − lo
0x23							out1 − hi	out1 − hi
0x24							*out2	stat-lo
0x25							out2 + +	stat-hi
0x26							out2 − lo	out2 − lo
0x27							out2 − hi	out2 − hi
0x28							out1 − c	out1 − c
0x29							in1 − c	in1 − c
0x2A							in1 − lo	in1 − lo
0x2B							in1 − hi	in1 − hi
0x2C							out2 − c	out2 − c
0x2D							in2 − c	in2 − c
0x2E							in2 − lo	in2 − lo
0x2F							in2 − hi	in2 − hi
0x30							stat − ram	r8
0x31							stat − c	r9
0x32							stat − lo	r10
0x33							stat − hi	r11
0x34							yeven − lo	r12
0x35							yeven − hi	r13
0x36							yodd − lo	r14
0x37							yodd − hi	r15
0x38							ypitch	shift
0x39								stat − c
0x3A							vu − lo	*dram1
0x3B							vu − hi	*dram2
0x3C							vupitch	*dram1 + +
0x3D							vpitch	*dram2 + +
0x3E							vptr − lo	*dram1 − −
0x3F							vptr − hi	*dram2 − −

Figure 4-2. 82750PB Instruction Word Format

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
mcode 2								mcode 3															
7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bit coding	A Bus Destination	A Bus Source	Cond Flag Select	Next Address
	6	6	3	9
0x0	null	null	FALSE	
0x1		hwid	CARRY	
0x2		cc	OVERFLOW	
0x3	moddr		SIGN	
0x4		alu	ZERO	
0x5	cnt	cnt	CNT0	
0x6	cnt2	cnt2	LSB r0	
0x7	lcnt	lcnt	MSB r0	
0x8	r0	r0		
0x9	r1	r1		
0xA	r2	r2		
0xB	r3	r3		
0xC	r4	r4		
0xD	r5	r5		
0xE	r6	r6		
0xF	r7	r7		
0x10	mcode3	mcode3		
0x11	mcode2	mcode2		
0x12	mcode1	mcode1		
0x13	pc	pc		
0x14	pixint − c			
0x15	pixint	pixint		
0x16	*dram1	*dram1		
0x17	*dram2	*dram2		
0x18	*dram1 + +	+dram1 + +		
0x19	*dram2 + +	+dram2 + +		
0x1A	*dram1 − −	+dram1 − −		
0x1B	*dram2 − −	+dram2 − −		
0x1C	dram1	dram1		
0x1D	dram2	dram2		
0x1E	dram3	dram3		
0x1F	dram4	dram4		
0x20	*out1	*in1		
0x21	out1 + +	*in2		
0x22	shift − rl	*stat		
0x23	out1 − hi	*stat#		
0x24	*out2			
0x25	out2 + +			
0x26	shift − r			
0x27	out2 − hi			
0x28	out1 − c			
0x29	in1 − c			
0x2A	shift − 1			
0x2B	in1 − hi			
0x2C	out2 − c			
0x2D	in2 − c			
0x2E				
0x2F	in2 − hi			
0x30	r8	r8		
0x31	r9	r9		
0x32	r10	r10		
0x33	r11	r11		
0x34	r12	r12		
0x35	r13	r13		
0x36	r14	r14		
0x37	r15	r15		
0x38	cc	shift		
0x39	fcnt	fcnt		
0x3A	*dram3	*dram3		
0x3B	*dram4	*dram4		
0x3C	*dram3 + +	*dram3 + +		
0x3D	*dram4 + +	*dram4 + +		
0x3E	*dram3 − −	*dram3 − −		
0x3F	*dram4 − −	*dram4 − −		

Figure 4-2. 82750PB Instruction Word Format (Continued)

5.0 ELECTRICAL DATA

Maximum Ratings

Table 5-1 is a stress rating only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in the DC and AC Characteristics (Tables 5-2, 5-3, 5-4, and 5-5).

Exposure to Maximum Ratings may affect device reliability. Furthermore, although the 82750PB contains protective circuitry to resist damage from static electrical discharge, always take precautions to avoid high static voltages or electric fields.

DC Characteristics

Table 5-1. Absolute Maximum Requirements

Condition	Maximum Requirement
Case Temperature under Bias	−65°C to +110°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−0.5V to V_{CC} + 0.5V
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V

Table 5-2. DC Characteristics V_{CC} = 5V ±10%; T_{CASE} = 0°C to +90°C

Symbol	Parameter	Min	Typ	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3		0.8	V	(Note 1)
V_{IH}	Input HIGH Voltage	2.0		V_{CC} + 0.3	V	(Note 1)
V_{OL}	Output LOW Voltage		0.2	0.4	V	I_{OL} = 4.0 mA(1)
V_{OH}	Output HIGH Voltage	2.4	3.0		V	I_{OH} = −1.0 mA(1)
I_{IL}	Input Leakage Current	−10		+10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	Output Leakage Current	−10		+10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		150	200	mA	25 MHz(2)
C_{IN}	Input Capacitance			10.0	pF	F_C = 1 MHz(3)
C_{OUT}	Output Capacitance			12.0	pF	F_C = 1 MHz(3)
C_{CLKIN}	CLKIN Input Capacitance			20.0	pF	F_C = 1 MHz(3)

NOTES:
1. Measured with CLKIN = 8 MHz.
2. Typical current value measured under typical conditions. Maximum current value guaranteed with 50 pF maximum output loading.
3. Not 100% tested.

AC Characteristics

Table 5-3. AC Characteristics at 25 MHz V_{CC} = 5V ±10%; T_{CASE} = 0°C to +90°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	25	MHz		1xClock
t_1	CLKIN Period	40	125	ns	5-1	
t_2	CLKIN High Time	14	26	ns	5-1	(Note 1)
t_3	CLKIN Low Time	14	26	ns	5-1	(Note 1)
t_4	CLKIN Fall Time		4	ns	5-1	
t_5	CLKIN Rise Time		4	ns	5-1	
t_{6a}	A[31:2], BE#[3:0], WE#, D[31:0], HINT#, PMFRZ# Valid Delay	3	25	ns	5-2	
t_{6b}	MREQ#, TRNFR#, RFSH#, NXTFST#, HBUSEN#, HRDY# Valid Delay	3	18	ns	5-2	
t_7	A[31:2], BE#[3:0], WE#, D[31:0] Float Delay		30	ns	5-2	(Note 2)
t_8	MRDY# Setup	10		ns	5-3	
t_9	MRDY# Hold	6		ns	5-3	
t_{10}	HREQ#, VBUS[3:0], RESET#, HALEN#, HALT# Setup	8		ns	5-3	
t_{11}	HREQ#, VBUS[3:0], RESET#, HALEN#, HALT# Hold	6		ns	5-3	
t_{12}	A[8:2], BE#[3:0], WE#, D[31:0] Setup	4		ns	5-3	(Note 3)
t_{13}	A[8:2], BE#[3:0], WE#, D[31:0] Hold	6		ns	5-3	(Note 3)
t_{14}	HREG#, HRAM# Setup	10		ns	5-3	
t_{15}	HREG#, HRAM# Hold	6		ns	5-3	
t_{16}	CLKOUT Valid Delay		18	ns	5-4	
t_{17}	CLKOUT High Time	$\frac{1}{2} t_1 - 6$	$\frac{1}{2} t_1 + 6$	ns	5-4	

NOTES:
1. This assumes 40 ns period. For other speeds these values should fall between 40% to 60% duty cycle.
2. Not 100% tested. Guaranteed by design characterization.
3. Inputs must remain valid throughout all cycles of host accesses. See Figures 3-6 through 3-8.
4. All A.C. specifications are measured at the 1.5V crossing point with a 50 pF load.

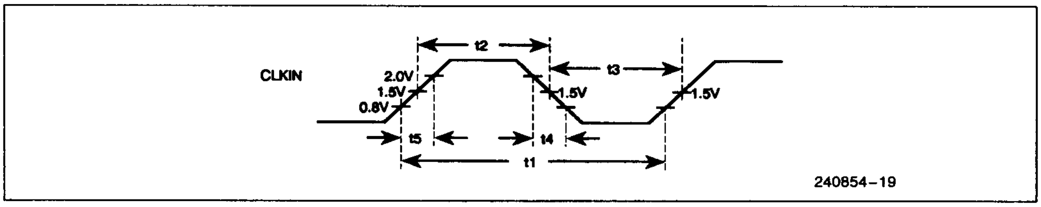


Figure 5-1. Clock Waveforms

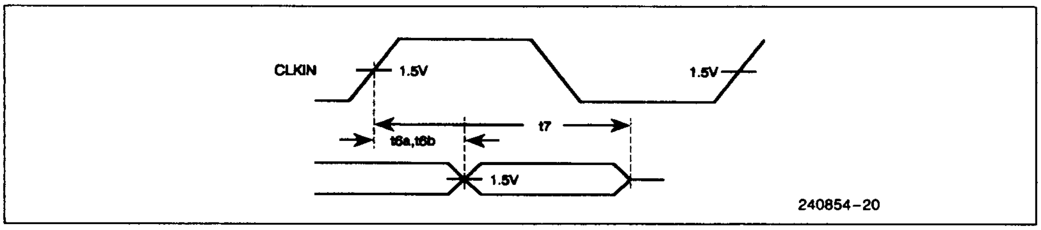
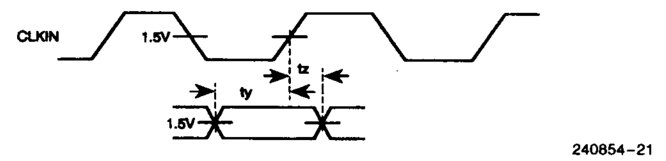


Figure 5-2. Output Waveforms



NOTES:
ty = t8, t10, t12, t14 (setup times)
tz = t9, t11, t13, t15 (hold times)

Figure 5-3. Input Waveforms

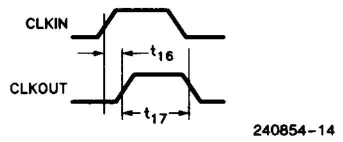
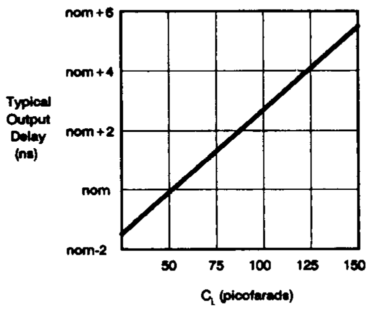


Figure 5-4. CLKOUT Waveforms

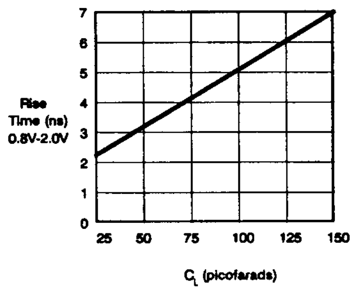
Output Delay and Rise Time Versus Load Capacitance



240854-22

NOTE:
This graph will not be linear outside of the C_L range shown.
nom = nominal value given in A.C. Characteristics table.

Figure 5-5. Typical Output Valid Delay Versus Load Capacitance under Worst Case Conditions



240854-23

NOTE:
This graph will not be linear outside of the C_L range shown.

Figure 5-6. Typical Output Rise Time Versus Load Capacitance under Worst Case Conditions

6.0 MECHANICAL DATA

Packaging Outlines and Dimensions

Intel packages the 82750PB in a Plastic Quad Flat Pack (PQFP). Table 6-1 gives the symbol list for the PQFP.

Table 6-1. PQFP Symbol List

Letter or Symbol	Description of Dimensions
A	Package Height: Distance from Seating Plane to Highest Point of Body
A_1	Standoff: Distance from Seating Plane to Base Plane
D/E	Overall Package Dimension: Lead Tip to Lead Tip
D_1/E_1	Plastic Body Dimension
D_2/E_2	Bumper Distance
D_3/E_3	Footprint
L_1	Foot Length
N	Total Number of Leads

The PQFP has the following specifications:

1. All dimensions and tolerances conform to ANSI Y14.5M-1982.
2. Datum plane —H— is located at the mold parting line and coincident with the bottom of the lead where lead exits plastic body.
3. Datums A–B and —D— are to be determined where center leads exit plastic body at datum plane —H—.
4. Controlling dimension is the inch.
5. Dimensions D_1, D_2, E_1, and E_2 are measured at the mold parting line and do not include mold protrusion. Allowable mold protrusion is 0.18 mm (0.007 in.) per side.
6. Pin 1 identifier is located within one of the two zones indicated.
7. Measured at datum plane —H—.
8. Measured at seating plane datum —C—.

Table 6-2 provides outline characteristics for 0.025 in. pitch.

Table 6-2. Intel Case Outline Drawings for PQFP at 0.025 Inch Pitch

Symbol	Description	Min	Max
N	Leadcount	132	132
A	Package Height	0.160	0.170
A_1	Standoff	0.020	0.030
D, E	Terminal Dimension	1.075	1.085
D_1, E_1	Package Body	0.0947	0.953
D_2, E_2	Bumper Distance	1.097	1.103
D_3, E_3	Lead Dimension	0.800 REF	0.800 REF
L_1	Foot Length	0.020	0.030

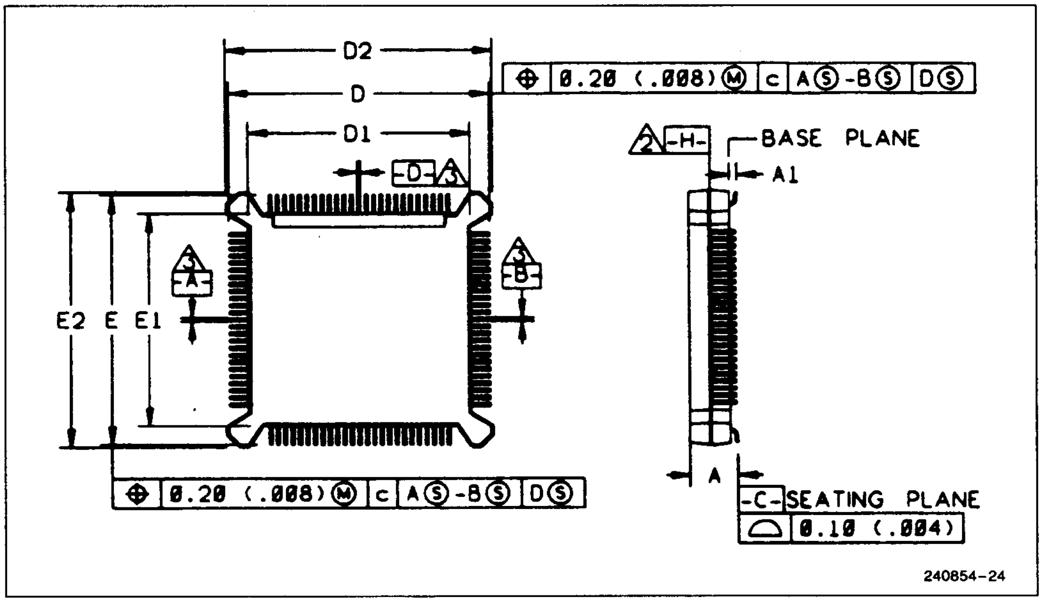


Figure 6-1. Principal Dimensions of the 82750PB in the 132-Lead PQFP Package

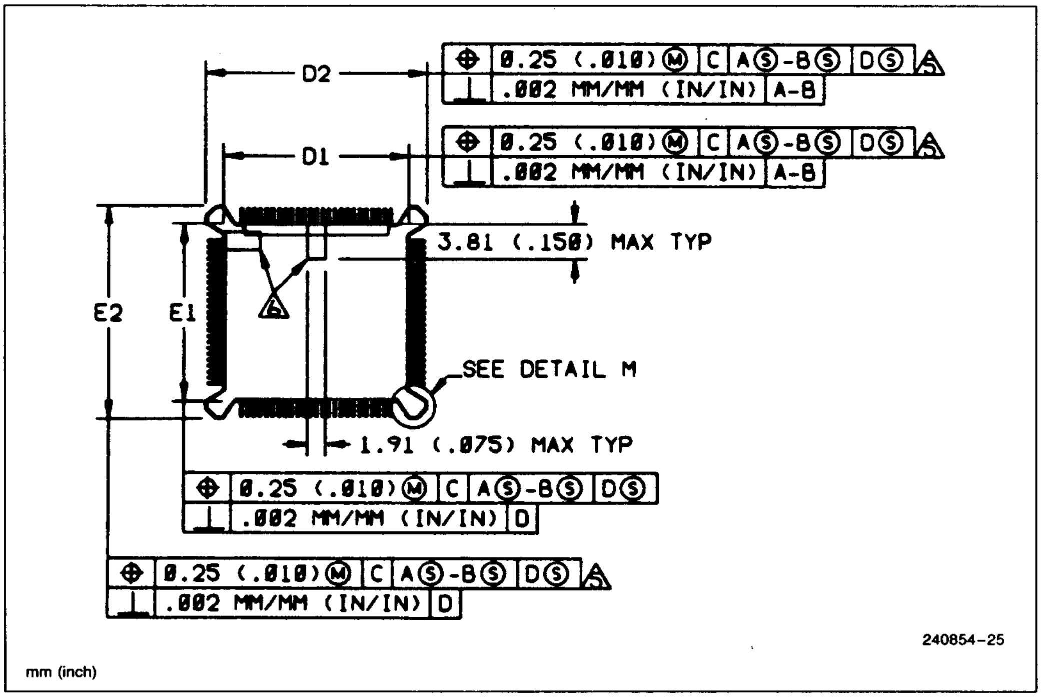


Figure 6-2. Detailed Dimensions of the 82750PB in the 132-Lead PQFP—Molding Details

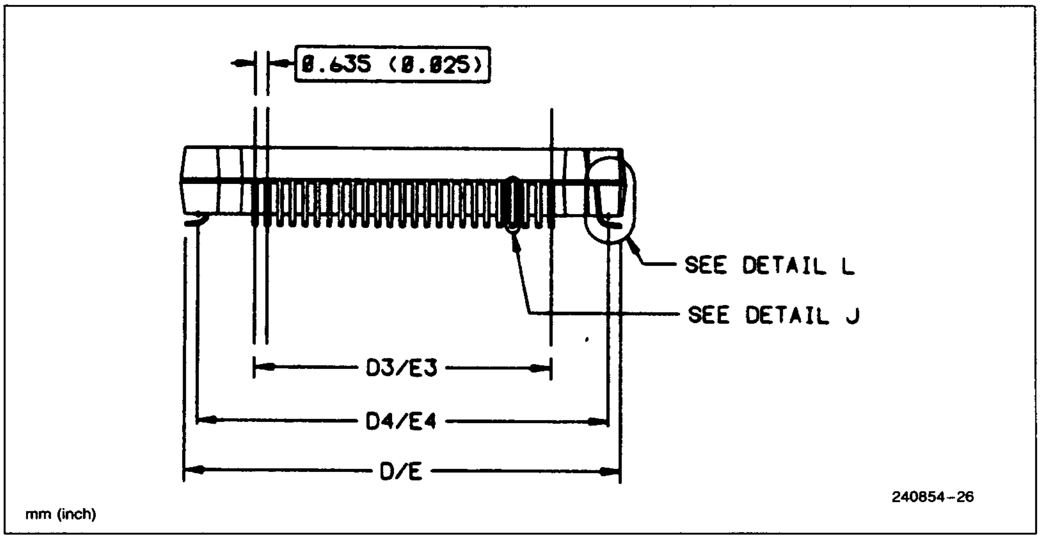


Figure 6-3. Detailed Dimensions of the 82750PB in the 132-Lead PQFP—Terminal Details

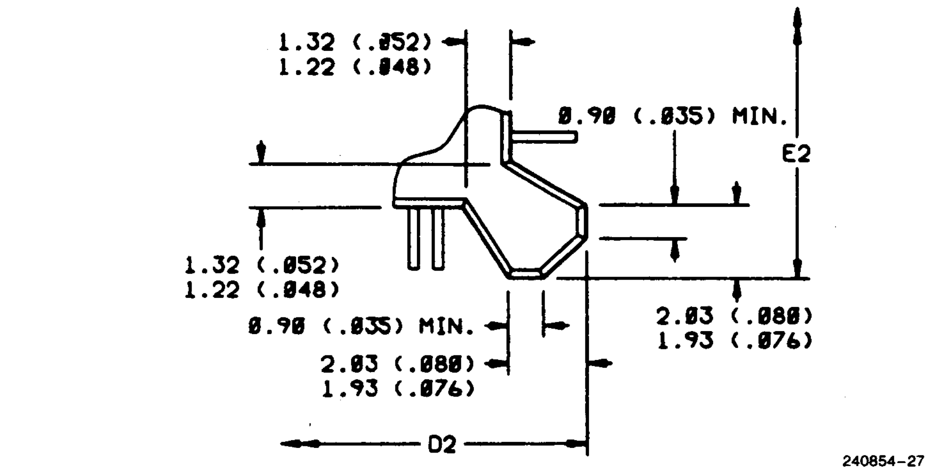


Figure 6-4. 132-Lead PQFP Mechanical Package Detail—Protective Bumper

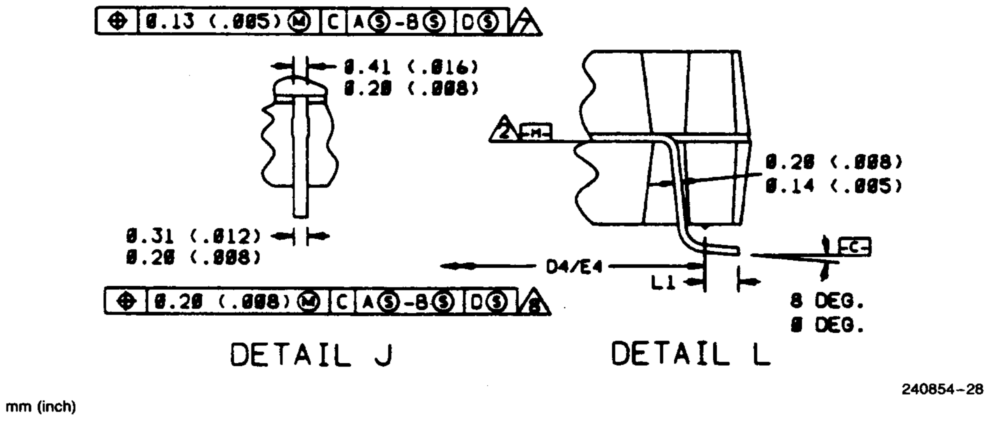


Figure 6-5. 132-Lead PQFP Mechanical Package Detail—Typical Lead

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1982
2. DATUM PLANE $\boxed{-H-}$ LOCATED AT THE MOLD PARTING LINE AND COINCIDENT WITH THE BOTTOM OF THE LEAD WHERE LEAD EXITS PLASTIC BODY
3. DATUMS $\boxed{A-B}$ AND $\boxed{-D-}$ TO BE DETERMINED WHERE CENTER LEADS EXIT PLASTIC BODY AT DATUM PLANE $\boxed{-H-}$
4. CONTROLLING DIMENSION, INCH
5. DIMENSIONS D1, D2, E1 AND E2 ARE MEASURED AT THE MOLD PARTING LINE. D1 AND E1 DO NOT INCLUDE AN ALLOWABLE MOLD PROTRUSION OF 0.18 MM (.007 IN) PER SIDE. D2 AND E2 DO NOT INCLUDE A TOTAL ALLOWABLE MOLD PROTRUSION OF 0.18 MM (.007 IN) AT MAXIMUM PACKAGE SIZE.
6. PIN 1 IDENTIFIER IS LOCATED WITHIN ONE OF THE TWO ZONES INDICATED
7. MEASURED AT DATUM PLANE $\boxed{-H-}$
8. MEASURED AT SEATING PLANE DATUM $\boxed{-C-}$

240854-29

Package Thermal Specifications

The 82750PB is specified for operation when T_C (the case temperature) is within the range of 0°C to 90°C. T_C may be measured in any environment to determine whether the 82750PB is within specified operation range. The case temperature should be measured at the center of the top surface.

T_A (the ambient temperature) can be calculated from θ_{CA} (thermal resistance from case to ambient) with the following equation:

$$T_A = T_C - P * \theta_{CA}$$

Typical values for θ_{CA} at various airflows are given in Table 6-3 for the 132-lead PQFP package. Table 6-4 shows the maximum T_A allowable (wihout exceeding T_C) at various airflows. The power dissipation (P) is calculated by using the typical supply current at 5V as shown in Table 5-2.

Table 6-3. Thermal Resistance (°C/W)

	θ_{CA} Versus Airflow—ft/min (m/sec)					
Package	**0 (0)**	**200 (1.01)**	**400 (2.03)**	**600 (3.04)**	**800 (4.06)**	**1000 (5.07)**
132-Lead PQFP	26.0	17.5	14.0	11.5	9.5	8.5

Table 6-4. Maximum T_A at Various Airflows (°C)

	T_A Versus Airflow—ft/min (m/sec)						
Package	**Frequency (MHz)**	**0 (0)**	**200 (1.01)**	**400 (2.03)**	**600 (3.04)**	**800 (4.06)**	**1000 (5.07)**
132-Lead PQFP	25	70	76	80	81	83	84