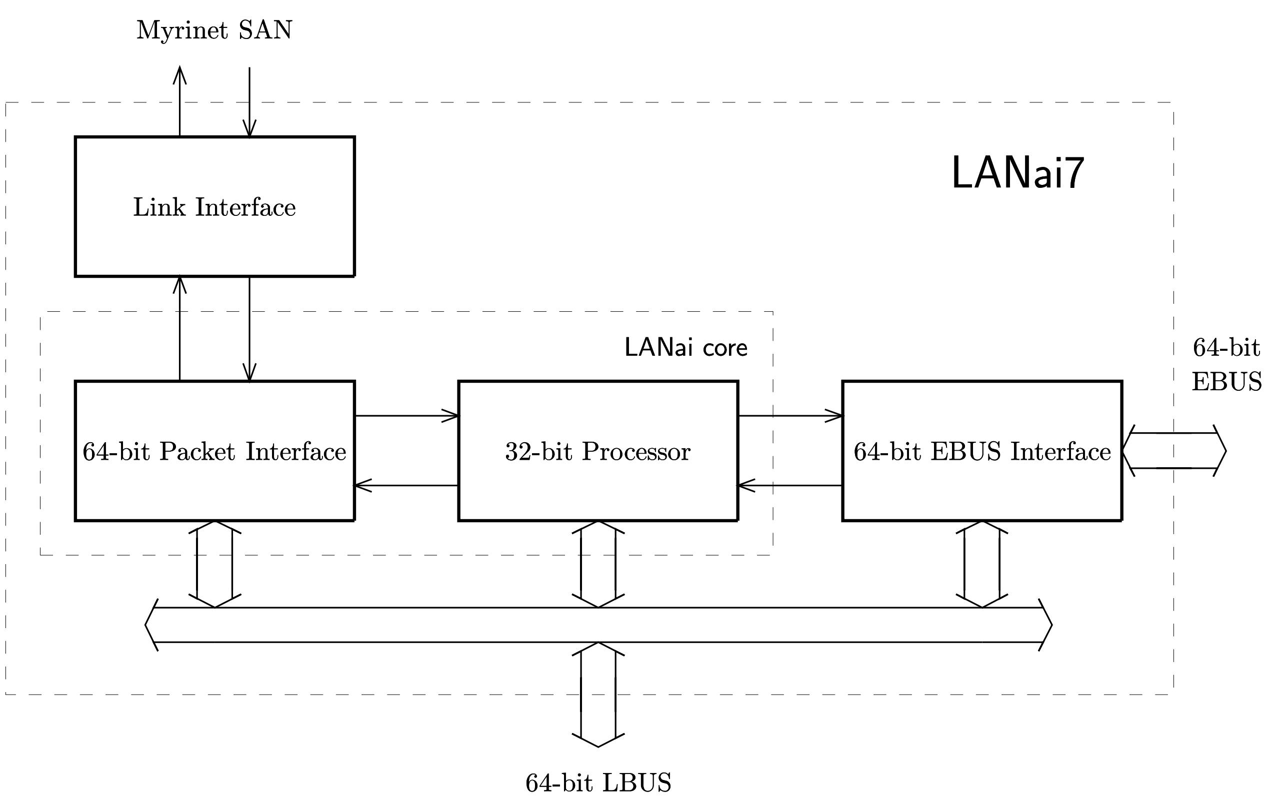
LANai 7

DRAFT
June 21, 1999

The LANai 7 is a programmable communication device that provides an interface to the Myrinet system-area network (SAN).

As illustrated below, a LANai 7 chip consists of the LANai core, with an instruction-interpreting processor and a packet interface, the Myrinet-SAN interface, and the EBUS interface.

The Local Bus (LBUS) is an interface to synchronous static SRAMs. The External Bus (EBUS) is a synchronous, pipelined interface.



HIGHLIGHTS OF LANAI7/LANAI5 DIFFERENCES

- The architecture of the CPU has been modified to enable it to work efficiently with pipelined SRAM. The load and the branch instructions have an extra shadow.

  The three-operand CPU instructions are not supported.

  The CPU has only one context and no interrupts.

  The Interrupt Status Register (ISR) from the LANai 5 is still available and renamed Interface Status Register. The ISR is a regular memory-mapped register, not a CPU register.

- The LBUS operates with zero-bus-turnaround synchronous SRAM (page 23).

  There are two memory accesses per LANai clock (no single-clocked LBUS mode).

  The LBUS interface includes parity checking (page 8).

  The LBUS write-protection mechanism has been replaced with write-detection (page 5).

- Only the core EBUS interface from the LANai 5 is supported. The LANai 7 chip is typically used with an external ASIC. This additional chip (available from Myricom) is compatible with the LANai 5, 6, and 7 chips. It supports several new features, including traversal of linked lists of EBUS-DMA requests, and EBUS DMA of arbitrary size and alignment.

  The read latency on the EBUS is 4 cycles compared to 3 cycles for the LANai 5 (page 25).

- The SAN interface on the LANai 7 chip includes several features that enable it to detect and report channel failure. These features require off-chip assistance, typically provided by a microcontroller with some associated circuitry (available from Myricom).

- There are three interval timers in the LANai 7 compared to one in the LANai 5 (page 5).

- An additional EBUS command is available, the INIT (page 28). Issuing this command while the chip is held in reset (typically, during power-on) initializes the CLOCK register to a value that, although it does not guarantee correct chip operation, does guarantee that the chip is properly reset.

MEMORY INTERFACE

In the remainder of this specification, we shall refer to 8-bit data units as bytes, to 16-bit units as half-words, to 32-bit units as words, and to 64-bit units as double-words. Although the internals of the LANai 7 chip support 32-bit addresses, pin-count limitations restrict the LBUS to a maximum of 8M bytes.

The LANai 7 LBUS operates at twice the chip-clock speed (two LBUS memory cycles for every clock cycle). The external-access bus (EBUS), the packet-interface receive DMA, and the packet-interface send DMA each request a maximum of one memory access per clock cycle. The on-chip processor requests up to two memory accesses per clock cycle (instruction and data). The two memory accesses are assigned based on the following priority (highest to lowest): EBUS, receive DMA, send DMA, and the processor. Since every EBUS memory request is granted, the LANai 7 chip along with the memory on its LBUS appears as a block of synchronous memory when observed from the EBUS.

Both the LBUS and the EBUS addresses are byte addresses, and the byte order is big-endian (the most-significant byte is stored at the lowest byte address).

The 2-, 4-, and 8-byte memory accesses on the LBUS must be aligned; any least-significant bits of an address that would make a memory access non-aligned are ignored.

The LANai chip provides a rudimentary memory-protection mechanism that can detect write accesses to a programmable-size memory segment (page 5).

The LANai core cannot access the EBUS directly.

PACKET SENDING

Please consult pages 11 through 16 for a set of send and receive examples.

A data-communication, flow-control unit is called a flit, and consists of eight data bits plus a tail bit. Packets are of arbitrary length (in flits), and the tail bit marks the last flit of every packet. The byte order in the communication network is big-endian, *i.e.,* the most-significant byte appears first in the network.

After the LANai 7 chip is out of reset, no Myrinet-network access is allowed until the link0_int bit of the special register ISR becomes 1, signaling that the SAN link has been initialized (page 7).

Packets are injected into the Myrinet network by initiating the send DMA. The following 32-bit, special registers control the send DMA:

Register	Description
SMP	**Send-Message Pointer:** Address of the first double-word of the send-DMA memory buffer. This register is incremented by 8 by the packet interface as each double-word is appended to the outgoing packet, and, upon completion, equals SML+8.
SA	**Send-Message Align:** The three least-significant bits of this register specify how many leading flits (0-7) of the contents of the next-specified send-DMA memory buffer should NOT be appended to the outgoing packet. This register may be used to keep the payload portion of the message 8-byte aligned, even when the length of the routing header is not a multiple of 8. Only the first send DMA following a write into this register is affected.
SMH	**Send-Message Header:** Address of the last double-word of the routing header. When the CRC-32 is enabled (page 6), writing SMH instructs the packet interface NOT to include the routing header in the CRC-32 for that packet (the routing header will be stripped by the Myrinet switches). Only the first send DMA following a write into this register is affected.
SMC	**Send-Message Header CRC:** Address of the double-word in the send-DMA memory buffer which is NOT appended to the outgoing packet; the optional, partial CRC-32, followed by four zero bytes, is sent instead. Only the first send DMA following a write into this register is affected.
SML	**Send-Message Limit:** Writing this register initiates a send DMA that appends to the outgoing packet, one double-word at a time, the contents of the memory buffer starting with the double-word at address SMP (except for the leading bytes, if specified by SA) and ending with the double-word at address SML. If SMH was written, the CRC-32 computation starts with the double-word at address SMH+8. If SMC was written, the partial CRC-32, followed by four zero flits, is sent instead of the double-word pointed to by SMC.
SMLT	**Send-Message Limit, with the Tail:** The same as SML, but, instead of appending the double-word at address SML, the following byte(s) complete the outgoing packet: 1) if the CRC-32 is enabled (page 6), the word equal to the CRC-32 of the outgoing packet (not including any partial CRC-32s), and 2) the tail flit equal to the CRC-8 of the outgoing packet (including all CRC-32 words and sets of four padding zero bytes accompanying partial CRC-32s).

Upon completion of a send DMA, the send_int bit of the special register ISR is set (page 8).

Since the send DMA accesses 64 bits at a time, the send memory buffer must be aligned on a double-word boundary. Hence, the three least-significant bits of SMP, SMH, SMC, and SML(T) are hard-wired to zero.

SA is a write-only register, and reading it produces an undefined value. Reading any other send register is a valid operation, but one should note that SML and SMLT are stored in the same physical register. If any of the send registers are written during a send DMA, the resulting behavior is undefined.

PACKET RECEIVING

Please consult pages 11 through 16 for a set of send and receive examples.

A data-communication, flow-control unit is called a flit, and consists of eight data bits plus a tail bit. Packets are of arbitrary length (in flits), and the tail bit marks the last flit of every packet. The byte order in the communication network is big-endian, *i.e.,* the most significant byte of a word (or of a half-word) appears first in the network.

After the LANai 7 chip is out of reset, no Myrinet-network access is allowed until the link0_int bit of the special register ISR becomes 1, signaling that the SAN link has been initialized (page 7). After the link0_int bit becomes 1, the SAN interface injects two tail flits into the incoming-message channel, to complete any previously truncated message. Consumption of these two, bogus messages is the responsibility of the programmer.

An incoming packet is accepted from the network by initiating the receive DMA. The following 32-bit, special registers control the receive DMA:

Register	Description
RMP	**Receive-Message Pointer:** Address of the first double-word of the receive-DMA memory buffer. This register is incremented by 8 by the packet interface as each double-word is written into the buffer. After an entire packet has been received, RMP points to the first aligned double-word past the end of the packet.
RMC	**Receive-Message Header CRC:** Points to the slot in the memory for the optional, partial CRC-32. When the CRC-32 is enabled (page 6) and RMC is written, if the message arrives with the correct partial CRC-32, zero is written into the double-word pointed to by RMC. Only the first receive DMA following a write into this register is affected. When the receive DMA has written the incoming message into the memory up to and including the double-word pointed to by RMC, the head_int bit of ISR is set (page 8).
RMW	**Receive-Message Header Wakeup:** This is the same physical register as RMC. However, when writing to RMW there is no side effect of requesting that the partial CRC-32 be verified. The intended use of RMW is to be able to set up an early warning of an incoming message, while the receive DMA is possibly still going on, even when the message header does not carry the partial CRC-32. When the receive DMA has written the incoming message into the memory up to and including the double-word pointed to by RMW, the head_int bit of ISR is set (page 8).
RML	**Receive-Message Limit:** Writing into RML enables a receive DMA and instructs the packet interface to put the (remainder of the) incoming packet, one double-word at a time, into the memory buffer that starts at RMP and ends at RML. If the message arrives with the correct CRC-8, zero is written into the last byte of the message. When the CRC-32 is enabled (page 6), if the message arrives with the correct CRC-32, zero is written into the four bytes preceding the tail byte.

When an entire incoming packet has been transferred into the receive memory buffer, the recv_int bit of the special register ISR is set (page 8). If the receive memory buffer has been exhausted (the last double-word written is at the location pointed to by RML, and RMP=RML+8), buff_int bit of ISR is set. After a receive DMA is initiated, one must not initiate another receive DMA until the recv_int bit, the buff_int bit, or both, have been set.

Since the receive DMA accesses 64 bits at a time, the receive memory buffer must be aligned on a double-word boundary. Hence, the three least-significant bits of RML, RMP, and RMC (RMW) are hard-wired to zero.

If the length in bytes (flits) of the incoming packet (including any CRC-32 and/or CRC-8) is not a multiple of 8, the overrun bits of ISR will be set (page 8). The packet following the currently accessed packet is guaranteed not to be corrupted. The bytes corresponding to the flits past the tail flit are undefined.

Reading any receive register is a valid operation. If any of these registers is written during a receive DMA, the resulting behavior is undefined.

COUNTERS/TIMERS

There are four real-time counters on the LANai 7 chip, all of which use the time reference that is equal to 40 times the period of the transmit clock of the Myrinet-SAN interface (page 32). Nominally, this is an 80 MHz clock, so the time reference is equal to 1/2 microsecond.

Register	Description
RTC	**Real-Time Clock:** This is a 32-bit counter that is incremented every time-reference period.
IT0, IT1, IT2	**Interval Timers:** These are 32-bit counters that are decremented every time-reference period. Whenever an interval timer makes a transition from 0x00000000 to 0xFFFFFFFF, the corresponding time_int bit of the special register ISR is set (page 8).

MEMORY PROTECTION

The LANai 7 chip provides a rudimentary memory-protection mechanism that can detect write accesses to a memory segment of programmable size.

Register	Description
MP	**Memory Protection:** If the WE (Write Enable) bit is 1, write to any memory location is allowed (no memory protection). Upon reset, this is the default value of the WE bit. If the WE bit is 0, the A12-A23 bits (the A bits) define the region(s) of memory in which the LANai is allowed to write: a write to a memory location is allowed if a bit in the address of that memory location is 1 and the corresponding A bit is 1.

Bit	31	30	29	28	27	26	25	24
Name	WE	-	-	-	-	-	-	-
Bit	23	22	21	20	19	18	17	16
Name	A23	A22	A21	A20	A19	A18	A17	A16
Bit	15	14	13	12	11	10	9	8
Name	A15	A14	A13	A12	-	-	-	-
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-

A typical use of this mechanism is to protect a memory segment at the bottom of the memory (where the code for the LANai processor is usually kept), and allow writes to addresses up to the highest available memory on the LBUS.

For example, writing the value 0x000FE000 to the MP special register protects the lowest 8KB and allows writes to addresses up to 1MB-1.

Pin-count limitations restrict the LBUS of the LANai 7 chip to a maximum of 8M bytes.

The MP special register is a write-only register, and reading it produces an undefined value.

CONFIGURATION

<table>
<tr><th>Register</th><th>Description</th></tr>
<tr><td>TIMEOUT</td><td>Incoming-Message Blocking Timeout: the four least-significant bits of the TIMEOUT special register specify the timeout period of the LANai watchdog timer. If the LANai 7 chip fails to consume an incoming message from the network for the duration of the timeout period, the watchdog timer sets the nres_int bit of the special register ISR (page 8), and, if the NRES_ENABLE bit of the special register MYRINET is set, it resets the LANai chip.
<table>
<tr><th>Value</th><th>Timeout Period</th></tr>
<tr><td>0</td><td>4 seconds</td></tr>
<tr><td>1</td><td>16 seconds</td></tr>
<tr><td>2</td><td>8 seconds</td></tr>
<tr><td>3</td><td>4 seconds</td></tr>
<tr><td>4</td><td>2 seconds</td></tr>
<tr><td>5</td><td>1 seconds</td></tr>
<tr><td>6</td><td>512 milliseconds</td></tr>
<tr><td>7</td><td>256 milliseconds</td></tr>
<tr><td>8</td><td>128 milliseconds</td></tr>
<tr><td>9-15</td><td>64 milliseconds</td></tr>
</table></td></tr>
<tr><td>MYRINET</td><td>Myrinet-Link Configuration: The two least-significant bits of this register enable the error-handling features of the Myrinet-SAN interface.
<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>NRES_ENABLE</td><td>When the LANai 7 chip fails to consume an incoming message for the duration of the period selected by the TIMEOUT register, the nres_int bit of the special register ISR is set (page 8). If the NRES_ENABLE is set, the chip will be reset when the nres_int bit is set.</td></tr>
<tr><td>1</td><td>CRC32_ENABLE</td><td>Enables the CRC-32 computation.</td></tr>
</table></td></tr>
<tr><td>DEBUG</td><td>Hardware-Debug Register: The five least-significant bits of this register select one of 32 internal signals to be output on the WIN pin, for timing observation.</td></tr>
<tr><td>CLOCK</td><td>Internal-Clock Phase-Adjusting Register: This special register controls the on-chip clock generation. During the power-on reset, the system-specific value must be written to this register from the EBUS (page 10). This register may be modified only while the chip is in reset (page 22).</td></tr>
</table>

PROGRAMMABLE OUTPUTS

Register	Description
LED	**LED Register:** The least-significant bit of this special register is driven to the LED output pin.
PULSE	**PULSE Register:** The three least-significant bits of this special register correspond to the three output pins: P0, P1, and P2. When a value of 1 is written into such a bit, a one-clock-cycle-long pulse is generated on the corresponding output pin.

The special registers described on this page are write-only registers, and reading any of them produces an undefined value.

INTERFACE-STATUS REGISTER

Register	Description
ISR	**Interface Status Register:** Contains the chip-status information. The ISR bits with the _sig (signal) postfix are included for simple host-LANai communication. A signal bit can be set only by the LANai processor and reset only from the EBUS, or vice versa. The ISR bits with the _int (interrupt) postfix are set by the packet interface or the EBUS interface when their corresponding events occur. These bits can be reset directly — by writing a 1 into them, or indirectly — in a bit-specific way.
EIMR	**External-Interrupt Mask Register:** When a bit of ISR is equal to 1 and the corresponding bit of EIMR is equal to 1, the INT output pin is asserted.

The ISR and EIMR consist of the following bits (bit 31 is the most significant):

Bit	Name	Description
31	debug_bit	This bit is always equal to 1, for compatibility with the earlier LANai versions.
30	host_sig	This bit is set when the LANai processor writes a 1 into it, and reset when a 1 is written into it from the EBUS.
29-27	0	Reserved.
26	link2_int	This bit is set by the Myrinet-SAN interface if the SAN link detects a message-length problem (LANai sending a packet for too long, transmit into the network blocked for too long, or a received packet is too long). This bit is cleared by the programmer, only directly — by writing a 1 into it.
25	link1_int	This bit is set by the Myrinet-SAN interface when the SAN link goes down (it has been disconnected, received illegal symbols, or detected an invalid reference voltage). This bit is cleared by the programmer, only directly — by writing a 1 into it.
24	link0_int	This bit is set by the Myrinet-SAN interface when the SAN link goes up. This bit is cleared by the programmer, only directly — by writing a 1 into it. Upon reset, no Myrinet-network access is allowed until the link0_int bit becomes 1.
23-16	lan7_sig - lan0_sig	Each of these 8 bits is set when a 1 is written into it from the EBUS, and reset when the LANai processor writes a 1 into it.

Bit	Name	Description
15	par_int	This bit is set when a parity error is detected on the LBUS. This bit is cleared by the programmer, only directly — by writing a 1 into it. After a power-on reset, the entire LBUS memory should be initialized to avoid false parity-error warnings.
14	mem_int	This bit is set when a memory-protection violation is detected (page 5). This bit is cleared by the programmer, only directly — by writing a 1 into it.
13	time2_int	This bit is set by the IT2 timer whenever it makes a transition from 0x00000000 to 0xFFFFFFFF. This bit is cleared by the programmer, either directly — by writing a 1 into it, or indirectly — when IT2 is written.
12	wake_int	This bit is set when the WAKE input pin is asserted. This bit is cleared by the programmer, only directly — by writing a 1 into it.
11	nres_int	This bit is set by the Myrinet-SAN interface whenever the LANai chip fails to consume an incoming message from the Myrinet network for the duration of the period specified by the TIMEOUT special register. If the NRES_ENABLE bit of the MYRINET special register is 1, the LANai chip is also reset. By examining the nres_int bit, one can distinguish between the reset-pin-induced and NRES-induced reset. This bit is cleared by the programmer, only directly — by writing a 1 into it.
10 - 8	orun4_int orun2_int orun1_int	These bits are set by the packet interface when an overrun condition is detected, *i.e.,* when the length in bytes (flits) of the incoming packet (including any CRC-32 and/or CRC-8) is not a multiple of 8. The three bits taken together represent the number of bytes in the receive buffer beyond the tail byte (0 through 7). For example, if the tail is received in the most-significant byte of a double-word, all three bits will be set, indicating that the values of the 7 least-significant bytes are undefined. These bits are cleared by the programmer, either directly — by writing a 1 into them, or indirectly — when RML is written.
7	time1_int	This bit is set by the IT1 timer whenever it makes a transition from 0x00000000 to 0xFFFFFFFF. This bit is cleared by the programmer, either directly — by writing a 1 into it, or indirectly — when IT1 is written.
6	time0_int	This bit is set by the IT0 timer whenever it makes a transition from 0x00000000 to 0xFFFFFFFF. This bit is cleared by the programmer, either directly — by writing a 1 into it, or indirectly — when IT0 is written.
5-4	lan9_sig - lan8_sig	Each of these 2 bits is set when a 1 is written into it from the EBUS, and reset when the LANai processor writes a 1 into it.
3	send_int	This bit is set by the packet interface to signal the completion of a send DMA, *i.e.,* when the contents of the send memory buffer and any associated CRCs have been appended to the outgoing packet. This bit is cleared by the programmer, either directly — by writing a 1 into it, or indirectly — when SML(T) is written. After a send DMA is initiated, one must not initiate another send DMA until the send_int bit becomes 1.
2	buff_int	This bit is set by the packet interface when the receive-DMA buffer has been exhausted (the last double-word written is at the location pointed to by RML, and RMP=RML+8). This bit is cleared by the programmer, either directly — by writing a 1 into it, or indirectly — when RML is written. After a receive DMA is initiated, one must not initiate another receive DMA until the recv_int bit, the buff_int bit, or both, become 1.
1	recv_int	This bit is set by the packet interface to signal the completion of a receive DMA, *i.e.,* when the entire incoming packet has been transferred into the receive memory buffer. This bit is cleared by the programmer, either directly — by writing a 1 into it, or indirectly — when RML is written. After a receive DMA is initiated, one must not initiate another receive DMA until the recv_int bit, the buff_int bit, or both, become 1.
0	head_int	This bit is set by the packet interface to signal that the head of a packet (up to and including the double-word pointed to by RMW (RMC)) has been received into the memory. The receive DMA continues until: 1) an entire incoming packet has been transferred into the receive memory buffer, or 2) the receive-DMA buffer has been exhausted. This bit is cleared by the programmer, either directly — by writing a 1 into it, or indirectly — when RML is written.

SPECIAL-REGISTER SUMMARY

Register	Read		Write		**Description**	Offset	Page
	EBUS	LANai	EBUS	LANai			
CLOCK			+		Clock Configuration	0x1F8	6
CTR	+	+	+	+	EBUS-DMA Counter	0x78	25
DEBUG			+	+	Hardware Debugging	0x138	6
EIMR	+	+	+	+	External-Interrupt Mask	0x58	7
ISR	+	+	+	+	Interface Status	0x50	7
IT0	+	+	+	+	Interval Timer	0x60	5
IT1	+	+	+	+	Interval Timer	0xC0	5
IT2	+	+	+	+	Interval Timer	0xC8	5
LAR	+	+	+	+	EBUS-DMA LBUS address	0x70	25
LED			+	+	LED Output Pin	0x140	6
MP			+	+	Memory Protection	0x150	5
MYRINET			+	+	Myrinet-Link Configuration	0x130	6
PULSE			+	+	P0, P1, P2 Output Pins	0xB8	6
RMC	+	+	+	+	Receive-DMA Header CRC	0xD8	4
RML	+	+	+	+	Initiate Receive DMA	0xE8	4
RMP	+	+	+	+	Receive-DMA Buffer	0xE0	4
RMW	+	+	+	+	Receive-DMA Header	0xD0	4
RTC	+	+	+	+	Real-Time Clock	0x68	5
SA			+	+	Send-DMA Alignment	0x118	3
SMC	+	+	+	+	Send-DMA Header CRC	0x110	3
SMH	+	+	+	+	Send-DMA Routing Header	0xF8	3
SML	+	+	+	+	Initiate Send DMA	0x100	3
SMLT	+	+	+	+	Initiate Send DMA with Tail	0x108	3
SMP	+	+	+	+	Send-DMA Buffer	0xF0	3
TIMEOUT			+	+	NRES-Timeout Selection	0x128	6

All special registers are memory-mapped, and can be accessed both by the LANai on-chip processor and from the EBUS (except for the special register CLOCK, that can be accessed only from the EBUS).

To access a special register from the LANai processor one should use the address of 0xFFFFFE00 plus the offset of that special register. The base address for EBUS access of special registers is application-specific; consult system documentation for details.

When accessing special registers, the regular memory arbitration mechanism described on page 2 applies. The mutual exclusion at any higher level is the responsibility of the programmer.

INITIALIZATION

During the power-on reset, the system-specific value listed below must be written to the CLOCK special register from the EBUS, to initialize the on-chip clock generation, and the chip must be held in reset a minimum of 10 milliseconds to allow the on-chip PLL to stabilize.

The entire memory on the LBUS should be initialized with arbitrary values, so that the parity bits of LBUS SRAM are initialized.

After chip reset, the on-chip processor begins executing code starting from the address 0.

The state of the nres_int bit in the ISR upon reset indicates whether the reset has been a regular, reset-pin initialization (0), or an NRES-induced reset (1).

All the remaining bits of ISR are equal to 0, except the debug_bit, which is equal to 1.

The MP register is initialized to the no-memory-protection state.

The special register EIMR is undefined and should be initialized by the programmer.

All other special registers are initialized to 0 upon reset.

After the chip is out of reset, no Myrinet-network access is allowed until the link0_int bit of the special register ISR becomes 1, signaling that the SAN link has been initialized (page 7). After the link0_int bit becomes 1, the SAN interface injects two tail flits into the incoming-message channel, to complete any previously truncated message. Consumption of these two, bogus messages is the responsibility of the programmer.

SYSTEM-SPECIFIC INITIALIZATION

Version	CLOCK
LANai7.0, LANai7.1 66MHz CLK 133MHz synchronous, pipelined, zero-bus-turnaround SRAM	0x03000381

SIMPLE MESSAGE SENDING AND RECEIVING EXAMPLES

In all the examples, data is represented by the following symbols:

```
'm'   -  message byte, used in CRC-32 computation
'h'   -  header byte, not used in CRC-32 computation
'.'   -  don't-care byte
'c8'  -  CRC-8 byte
'c32' -  one of four CRC-32 bytes
'v8'  -  a verified CRC-8 byte (zero if CRC-8 caught no errors)
'v32' -  one of four verified CRC-32 bytes (zero if CRC-32 caught no errors)
```

Example #1:

```
        CRC-32   off
                                                        sender memory
                                        msb                             lsb
sender ()                               +---+---+---+---+---+---+---+---+
{                               SMP ->  | m | m | m | m | m | m | m | m |
  SMP  = 0x1000;              (start)   +---+---+---+---+---+---+---+---+
  SMLT = 0x1010;                        | m | m | m | m | m | m | m | m |
}                                       +---+---+---+---+---+---+---+---+
                                SMLT -> | . | . | . | . | . | . | . | . |
                                        +---+---+---+---+---+---+---+---+
                                SMP ->  | . | . | . | . | . | . | . | . |
                                 (end)  +---+---+---+---+---+---+---+---+


                                                       bytes in Myrinet
                                        +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                                        | c8|
                                        +---+


                                                       receiver memory
                                        msb                             lsb
receiver ()                             +---+---+---+---+---+---+---+---+
{                               RMP ->  | m | m | m | m | m | m | m | m |
  RMP = 0x2000;               (start)   +---+---+---+---+---+---+---+---+
  RML = 0x3000;                         | m | m | m | m | m | m | m | m |
}                                       +---+---+---+---+---+---+---+---+
                                        | v8| . | . | . | . | . | . | . |
                                        +---+---+---+---+---+---+---+---+
                                RMP ->  | . | . | . | . | . | . | . | . |
                                 (end)  +---+---+---+---+---+---+---+---+
                                          .   .   .   .   .   .   .   .
                                          .   .   .   .   .   .   .   .
                                          .   .   .   .   .   .   .   .
                                        +---+---+---+---+---+---+---+---+
                                RML ->  | . | . | . | . | . | . | . | . |
                                        +---+---+---+---+---+---+---+---+
```

Example #2:

```
        CRC-32   on
                                                        sender memory
                                              msb                            lsb
sender ()                                     +---+---+---+---+---+---+---+---+
{                                     SMP ->  | m | m | m | m | m | m | m | m |
  SMP  = 0x1000;                    (start)   +---+---+---+---+---+---+---+---+
  SMLT = 0x1010;                              | m | m | m | m | m | m | m | m |
}                                             +---+---+---+---+---+---+---+---+
                                      SMLT -> | . | . | . | . | . | . | . | . |
                                              +---+---+---+---+---+---+---+---+
                                      SMP ->  | . | . | . | . | . | . | . | . |
                                       (end)  +---+---+---+---+---+---+---+---+


                                                       bytes in Myrinet
                                              +---+---+---+---+---+---+---+---+
                                              | m | m | m | m | m | m | m | m |
                                              +---+---+---+---+---+---+---+---+
                                              | m | m | m | m | m | m | m | m |
                                              +---+---+---+---+---+---+---+---+
                                              |c32|c32|c32|c32| c8|
                                              +---+---+---+---+---+


                                                       receiver memory
                                              msb                            lsb
receiver ()                                   +---+---+---+---+---+---+---+---+
{                                     RMP ->  | m | m | m | m | m | m | m | m |
  RMP = 0x2000;                     (start)   +---+---+---+---+---+---+---+---+
  RML = 0x3000;                               | m | m | m | m | m | m | m | m |
}                                             +---+---+---+---+---+---+---+---+
                                              |v32|v32|v32|v32| v8| . | . | . |
                                              +---+---+---+---+---+---+---+---+
                                      RMP ->  | . | . | . | . | . | . | . | . |
                                       (end)  +---+---+---+---+---+---+---+---+
                                                .   .   .   .   .   .   .   .
                                                .   .   .   .   .   .   .   .
                                                .   .   .   .   .   .   .   .
                                              +---+---+---+---+---+---+---+---+
                                      RML ->  | . | . | . | . | . | . | . | . |
                                              +---+---+---+---+---+---+---+---+
```

Example #3:

```
        CRC-32   on
        SMH      used
                                                      sender memory
                                                 msb                          lsb
sender ()                                        +---+---+---+---+---+---+---+---+
{                                       SMP ->   | h | h | h | h | h | h | h | h |
  SMP  = 0x1000;                       (start)   +---+---+---+---+---+---+---+---+
  SMH  = 0x1008;                        SMH ->   | h | h | h | h | h | h | h | h |
  SMLT = 0x1020;                                 +---+---+---+---+---+---+---+---+
}                                                | m | m | m | m | m | m | m | m |
                                                 +---+---+---+---+---+---+---+---+
                                                 | m | m | m | m | m | m | m | m |
                                                 +---+---+---+---+---+---+---+---+
                                        SMLT ->  | . | . | . | . | . | . | . | . |
                                                 +---+---+---+---+---+---+---+---+
                                        SMP ->   | . | . | . | . | . | . | . | . |
                                         (end)   +---+---+---+---+---+---+---+---+


                                                     bytes in Myrinet
                                                 +---+---+---+---+---+---+---+---+
                                                 | h | h | h | h | h | h | h | h |
                                                 +---+---+---+---+---+---+---+---+
                                                 | h | h | h | h | h | h | h | h |
                                                 +---+---+---+---+---+---+---+---+
                                                 | m | m | m | m | m | m | m | m |
                                                 +---+---+---+---+---+---+---+---+
                                                 | m | m | m | m | m | m | m | m |
                                                 +---+---+---+---+---+---+---+---+
                                                 |c32|c32|c32|c32| c8|
                                                 +---+---+---+---+---+


                                                     receiver memory
                                                 (assuming the first 16 bytes
                                                  have been stripped by switches)
                                                 msb                          lsb
receiver ()                                      +---+---+---+---+---+---+---+---+
{                                       RMP ->   | m | m | m | m | m | m | m | m |
  RMP = 0x2000;                        (start)   +---+---+---+---+---+---+---+---+
  RML = 0x3000;                                  | m | m | m | m | m | m | m | m |
}                                                +---+---+---+---+---+---+---+---+
                                                 |v32|v32|v32|v32| v8| . | . | . |
                                                 +---+---+---+---+---+---+---+---+
                                        RMP ->   | . | . | . | . | . | . | . | . |
                                         (end)   +---+---+---+---+---+---+---+---+
                                                   .   .   .   .   .   .   .   .
                                                   .   .   .   .   .   .   .   .
                                                   .   .   .   .   .   .   .   .
                                                 +---+---+---+---+---+---+---+---+
                                        RML ->   | . | . | . | . | . | . | . | . |
                                                 +---+---+---+---+---+---+---+---+
```

```
Example #4:

        CRC-32   on
        SMH      used
        SA       used
                                                       sender memory
                                              msb                            lsb
sender ()                                     +---+---+---+---+---+---+---+---+
{                                     SMP ->  | . | . | . | . | . | h | h | h |
  SMP  = 0x1000;                     (start)  +---+---+---+---+---+---+---+---+
  SMH  = 0x1008;                      SMH ->  | h | h | h | h | h | h | h | h |
  SA   = 0x5;                                 +---+---+---+---+---+---+---+---+
  SMLT = 0x1020;                              | m | m | m | m | m | m | m | m |
}                                             +---+---+---+---+---+---+---+---+
                                              | m | m | m | m | m | m | m | m |
                                              +---+---+---+---+---+---+---+---+
                                      SMLT -> | . | . | . | . | . | . | . | . |
                                              +---+---+---+---+---+---+---+---+
                                      SMP ->  | . | . | . | . | . | . | . | . |
                                       (end)  +---+---+---+---+---+---+---+---+


                                                      bytes in Myrinet
                                                                  +---+---+---+
                                                                  | h | h | h |
                                              +---+---+---+---+---+---+---+---+
                                              | h | h | h | h | h | h | h | h |
                                              +---+---+---+---+---+---+---+---+
                                              | m | m | m | m | m | m | m | m |
                                              +---+---+---+---+---+---+---+---+
                                              | m | m | m | m | m | m | m | m |
                                              +---+---+---+---+---+---+---+---+
                                              |c32|c32|c32|c32| c8|
                                              +---+---+---+---+---+


                                                      receiver memory
                                              (assuming the first 11 bytes
                                               have been stripped by switches)
                                              msb                            lsb
receiver ()                                   +---+---+---+---+---+---+---+---+
{                                     RMP ->  | m | m | m | m | m | m | m | m |
  RMP = 0x2000;                      (start)  +---+---+---+---+---+---+---+---+
  RML = 0x3000;                               | m | m | m | m | m | m | m | m |
}                                             +---+---+---+---+---+---+---+---+
                                              |v32|v32|v32|v32| v8| . | . | . |
                                              +---+---+---+---+---+---+---+---+
                                      RMP ->  | . | . | . | . | . | . | . | . |
                                       (end)  +---+---+---+---+---+---+---+---+
                                                .   .   .   .   .   .   .   .
                                                .   .   .   .   .   .   .   .
                                                .   .   .   .   .   .   .   .
                                              +---+---+---+---+---+---+---+---+
                                      RML ->  | . | . | . | . | . | . | . | . |
                                              +---+---+---+---+---+---+---+---+
```

```
Example #5:

        CRC-32   on
        SMH      used
        SA       used
        SMC      used
        RMC      used
                                                  sender memory
                                        msb                            lsb
sender ()                               +---+---+---+---+---+---+---+---+
{                               SMP ->  | . | . | . | . | . | h | h | h |
  SMP   = 0x1000;             (start)   +---+---+---+---+---+---+---+---+
  SMH   = 0x1008;               SMH ->  | h | h | h | h | h | h | h | h |
  SA    = 0x5;                          +---+---+---+---+---+---+---+---+
  SMC   = 0x1020;                       | m | m | m | m | m | m | m | m |
  SMLT = 0x1038;                        +---+---+---+---+---+---+---+---+
}                                       | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                                SMC ->  | . | . | . | . | . | . | . | . |
                                        +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                                SMLT -> | . | . | . | . | . | . | . | . |
                                        +---+---+---+---+---+---+---+---+
                                SMP ->  | . | . | . | . | . | . | . | . |
                               (end)    +---+---+---+---+---+---+---+---+


                                                 bytes in Myrinet
                                                            +---+---+---+
                                                            | h | h | h |
                                        +---+---+---+---+---+---+---+---+
                                        | h | h | h | h | h | h | h | h |
                                        +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                partial CRC-32 ->       |c32|c32|c32|c32| 0 | 0 | 0 | 0 |
                (on m's up to here)     +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                                        | m | m | m | m | m | m | m | m |
                                        +---+---+---+---+---+---+---+---+
                full CRC-32 ->          |c32|c32|c32|c32| c8|
                (on m's up to here)     +---+---+---+---+---+
```

```
                                                   receiver memory
                                          (assuming the first 11 bytes
                                           have been stripped by switches)
                                          msb                             lsb
receiver ()                               +---+---+---+---+---+---+---+---+
{                                 RMP ->  | m | m | m | m | m | m | m | m |
  RMP = 0x2000;                  (start)  +---+---+---+---+---+---+---+---+
  RMC = 0x2010;                           | m | m | m | m | m | m | m | m |
  RML = 0x3000;                           +---+---+---+---+---+---+---+---+
}                                 RMC ->  |v32|v32|v32|v32| 0 | 0 | 0 | 0 |
                                          +---+---+---+---+---+---+---+---+
                                          | m | m | m | m | m | m | m | m |
                                          +---+---+---+---+---+---+---+---+
                                          | m | m | m | m | m | m | m | m |
                                          +---+---+---+---+---+---+---+---+
                                          |v32|v32|v32|v32| v8| . | . | . |
                                          +---+---+---+---+---+---+---+---+
                                  RMP ->  | . | . | . | . | . | . | . | . |
                                   (end)  +---+---+---+---+---+---+---+---+
                                            .   .   .   .   .   .   .   .
                                            .   .   .   .   .   .   .   .
                                            .   .   .   .   .   .   .   .
                                          +---+---+---+---+---+---+---+---+
                                  RML ->  | . | . | . | . | . | . | . | . |
                                          +---+---+---+---+---+---+---+---+
```

ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Symbol	Rating	Value	Unit
V_{dd}	Power Supply Voltage	-0.5 to +4.6	V
V_{in}, V_{out}	Terminal Voltage (except Vdd)	-0.5 to V_{dd}+0.5	V
I_{out}	Output Current	100	mA
P_D	Power Dissipation	3	W
T_{bias}	Temperature Under Bias	-55 to 125	^{o}C
T_{stg}	Storage Temperature	-55 to 125	^{o}C
T_A	Operating Temperature	0 to 70	^{o}C

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Min	Typ	Max	Unit
V_{dd}	Power Supply Voltage	3.0	3.3	3.6	V
V_{IH}	Input High Voltage	2.2	-	Vdd+0.3	V
V_{IL}	Input Low Voltage	-0.3	-	0.8	V

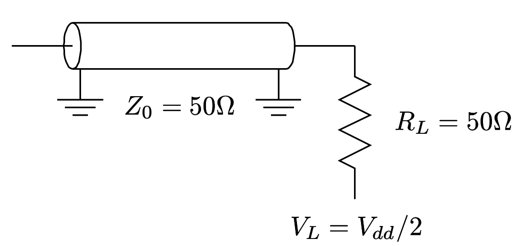
DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit
I_{LI}	Input Leakage Current	-	±1.0	μA
I_{LO}	Output Leakage Current	-	±1.0	μA
$V_{OL}(IOL = 5mA)$	Output Low Voltage	-	0.4	V
$V_{OH}(IOH = -5mA)$	Output High Voltage	2.4	-	V

CAPACITANCE ($T_A = +25^oC, f = 1.0MHz, dV = 3V$)

Symbol	Parameter	Max	Unit
C_I	Input Capacitance	5	pF
$C_{I/O}$	I/O Capacitance	8	pF

AC TEST LOADS



OUTPUT DRIVERS

The output drivers and the tri-state drivers are of three different strengths. For purely capacitive loads, the following are the typical delay values at $V_{dd} = 3.3V, T_a = 25^oC$:

Output	**Delay**
LCLK, LCLK1, A03 - A22, $\overline{\text{A20}}$, $\overline{\text{SW}}$, $\overline{\text{WE0}}$ - $\overline{\text{WE7}}$	$out_L = 0.2ns + C_{\text{load}}[pF] * 0.015ns/pF$
ED00 - ED63	$out_E = 0.2ns + C_{\text{load}}[pF] * 0.030ns/pF$
otherwise	$out = 0.2ns + C_{\text{load}}[pF] * 0.042ns/pF$

For power-supply voltage of $3.3V \pm 10\%$, ambient temperature from 0^o to 70^oC, and the acceptable manufacturing-process variation, output-driver delays have $\pm 50\%$ variation.

The full hspice model is available upon request.

PINOUT

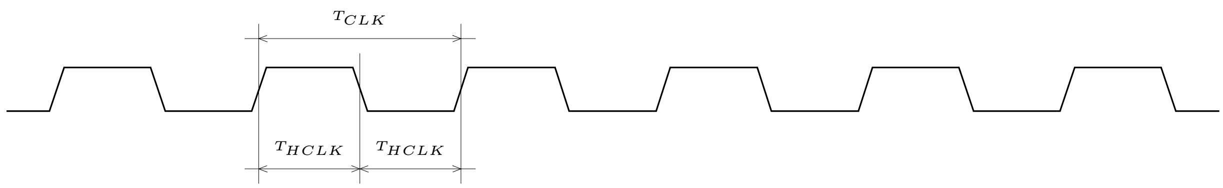
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
A	Vdd	GND	I7	I4	I0	GND	O3	GND	O6	OD	$\overline{\text{SLAT}}$	GND	A04	A08	A11	GND	A17	GND	A22	$\overline{\text{WE2}}$	$\overline{\text{WE5}}$	GND	Vdd
B	GND	Vdd	GND	I6	I3	Vth	LVdd	O4	LVdd	LVdd	OAH	LED	A05	A09	A13	A15	A18	A21	$\overline{\text{WE1}}$	$\overline{\text{WE4}}$	GND	Vdd	GND
C	$\overline{\text{ERST}}$	GND	Vdd	IB	I5	I2	O0	O2	BIAS	O7	OAL	A03	A06	A10	A14	A16	A20	$\overline{\text{WE0}}$	$\overline{\text{WE3}}$	$\overline{\text{WE7}}$	Vdd	GND	D02
D	SCLK	SDO	.	Vdd	ID	Vdd	I1	O1	Vdd	O5	OB	Vdd	A07	A12	Vdd	A19	LCLK1	Vdd	$\overline{\text{WE6}}$	Vdd	D00	D03	D05
E	P0	BCLK	SDI	$\overline{\text{RST}}$																D01	D04	D06	D08
F	GND	TCLK	P2	Vdd																Vdd	D07	D09	GND
G	INT	CLK	WIN	P1																D0P	D10	D12	D14
H	GND	$\overline{\text{A20}}$	.	RCLK																D11	D13	D15	GND
J	EP	WAKE	$\overline{\text{SW}}$	Vdd																Vdd	D1P	D16	D18
K	$\overline{\text{EWE5}}$	$\overline{\text{EWE6}}$	$\overline{\text{EWE7}}$	FRDY																D17	D19	D20	D21
L	$\overline{\text{EWE1}}$	$\overline{\text{EWE2}}$	$\overline{\text{EWE3}}$	$\overline{\text{EWE4}}$																D22	D23	D2P	D24
M	GND	ED63	$\overline{\text{EWE0}}$	Vdd								LANai 7								Vdd	D26	D25	GND
N	ED62	ED61	ED60	ED59																D30	D29	D28	D27
P	ED58	ED57	ED56	ED54																D34	D32	D3P	D31
R	ED55	ED53	ED52	Vdd																Vdd	D36	D35	D33
T	GND	ED51	ED49	ED47																D40	D38	D37	GND
U	ED50	ED48	ED46	ED43																D44	D41	D4P	D39
V	GND	ED45	ED42	Vdd																Vdd	D45	D42	GND
W	ED44	ED41	ED39	WMEM																D50	D5P	D46	D43
Y	ED40	DMA	$\overline{\text{EOE}}$	Vdd	RFIFO	Vdd	ED34	ED30	Vdd	ED23	ED18	Vdd	ED09	ED04	Vdd	D62	D58	Vdd	D52	Vdd	.	D48	D47
AA	RMEM	GND	Vdd	.	ED38	ED35	ED31	ED27	ED25	ED21	ED17	ED14	ED10	ED06	ED02	D7P	D61	D57	D55	D51	Vdd	GND	D49
AB	GND	Vdd	GND	LCLK	ED36	ED32	ED29	ED26	ED24	ED20	ED16	ED13	ED11	ED07	ED03	ED01	D63	D60	D56	D54	GND	Vdd	GND
AC	Vdd	GND	SPEC	ED37	ED33	GND	ED28	GND	ED22	ED19	ED15	GND	ED12	ED08	ED05	GND	ED00	GND	D59	D6P	D53	GND	Vdd

304-pin SuperBGA (top view)

CLOCKING

Pin	I/O	Description
CLK	I	**Clock:** The main clock input.
RCLK	I	**Reference Clock:** The additional clock-reference input.

CLK

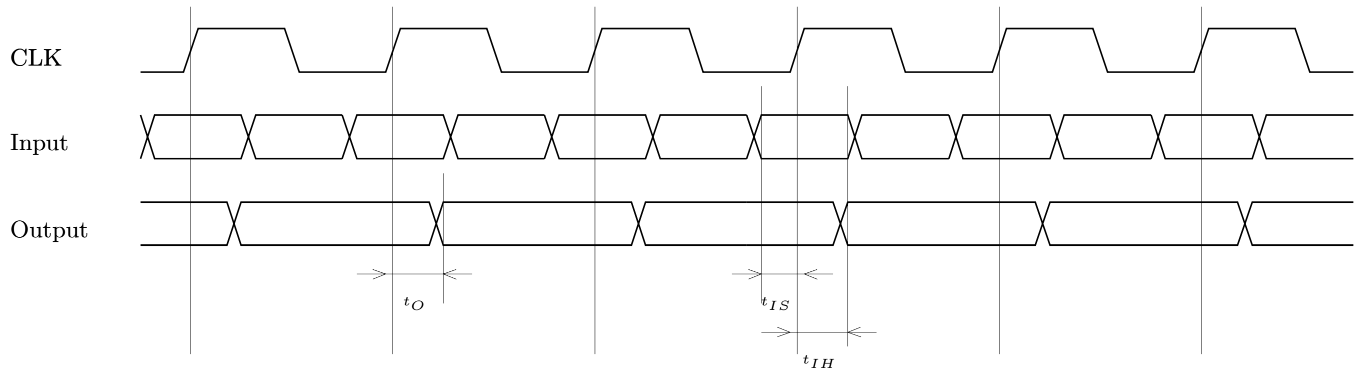


Symbol	Parameter	Min	Max
T_{CLK}	Clock period	15ns	20μs
T_{HCLK}	Clock half-period	6ns	10μs

CLK is the main clock input, and, except for the Myrinet-SAN interface (page 32), the entire chip works off of this clock. The internal chip clocks are derived from CLK, and the internal-clocking configuration is selected by the special register CLOCK (page 6):

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>7</td><td>PLL</td><td>This bit must be set to 1, to instruct the on-chip PLL to produce a symmetric on-chip clock.</td></tr>
<tr><td>6-4</td><td>MULT</td><td>Frequency of the internal chip clock (ICLK) is a multiple of the frequency of the CLK signal. The three MULT bits determine the frequency-multiplication factor:
<table>
<tr><th>Value</th><th>Internal Clock</th></tr>
<tr><td>0</td><td>1 x CLK</td></tr>
<tr><td>1</td><td>1.5 x CLK</td></tr>
<tr><td>2</td><td>2 x CLK</td></tr>
<tr><td>3</td><td>2.5 x CLK</td></tr>
<tr><td>4</td><td>3 x CLK</td></tr>
<tr><td>6</td><td>4 x CLK</td></tr>
</table></td></tr>
<tr><td>3-2</td><td>-</td><td>Reserved.</td></tr>
<tr><td>1</td><td>SELR</td><td>In systems where the CLK input is connected to a clock that may be stopped for arbitrarily long time periods (such as the PCI bus clock), the on-chip PLL requires an additional clock reference on the RCLK input. This clock reference must have duty cycle of 45-55%, maximum frequency of 80MHz, and its period must be smaller than the smallest possible period of CLK on that system.
The maximum clock period and half-period requirements remain 20μs and 10μs, respectively.</td></tr>
<tr><td>0</td><td>DBL</td><td>If this bit is 1, there are two LBUS memory accesses per clock cycle. If this bit is 0, there is one LBUS memory access per clock cycle. The LANai 7.0 and 7.1 versions of the chip do not support single-clocked LBUS, so the DBL bit must be set to 1.</td></tr>
</table>

SYNCHRONOUS INPUTS AND OUTPUTS

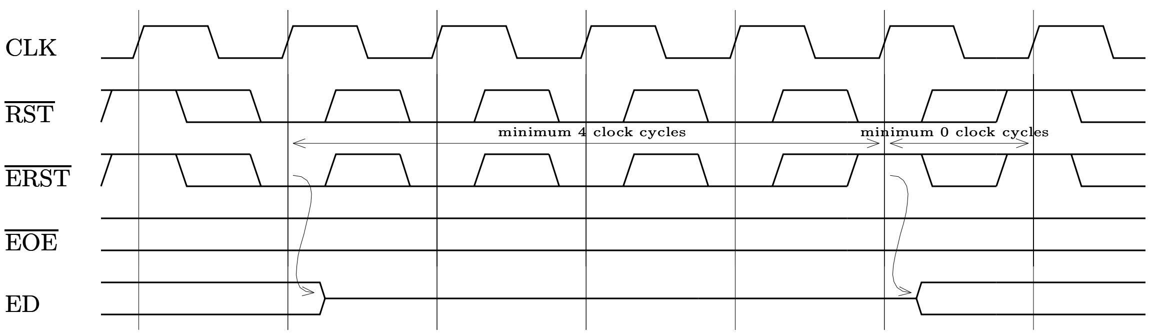


Symbol	Parameter	Min	Max
t_{IS}	Synchronous-input setup time	3ns	
t_{IH}	Synchronous-input hold time	0ns	
t_O	Synchronous-output delay (ICLK = CLK)	1ns	7ns
t_O	Synchronous-output delay (ICLK = 1.5 x CLK)	1ns	$\frac{2}{3}T_{CLK} + 7$ns
t_O	Synchronous-output delay (ICLK = 2 x CLK)	1ns	$\frac{1}{2}T_{CLK} + 7$ns
t_O	Synchronous-output delay (ICLK = 2.5 x CLK)	1ns	$\frac{4}{5}T_{CLK} + 7$ns
t_O	Synchronous-output delay (ICLK = 3 x CLK)	1ns	$\frac{2}{3}T_{CLK} + 7$ns
t_O	Synchronous-output delay (ICLK = 4 x CLK)	1ns	$\frac{3}{4}T_{CLK} + 7$ns

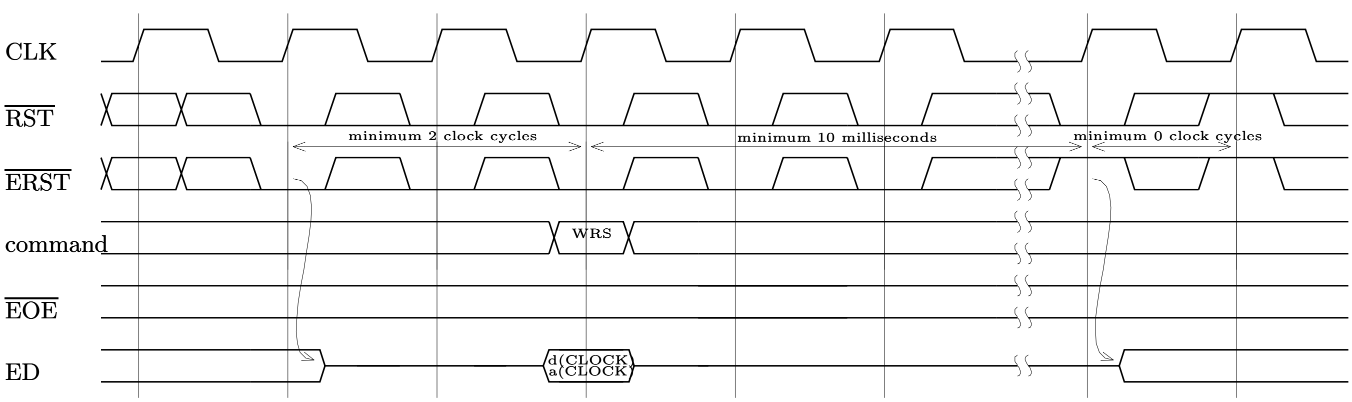
RESET SEQUENCE

Pin	I/O	Description
$\overline{\text{RST}}$	synch. I	**Reset:** The main reset input. During the power-on reset, the special register CLOCK, which controls the on-chip clock-generation, must be initialized. Note that, unlike with other special registers, both $\overline{\text{RST}}$ and $\overline{\text{ERST}}$ must be asserted while the CLOCK is written. This register need not be initialized during a non-power-on reset.
$\overline{\text{ERST}}$	synch. I	**EBUS Reset:** This additional reset input resets the LANai circuitry associated with EBUS access. When the $\overline{\text{RST}}$ input is asserted and $\overline{\text{ERST}}$ is not, the EBUS access is enabled so that, for example, code for the LANai processor can be loaded into the LBUS memory.

Non-Power-On Reset Sequence



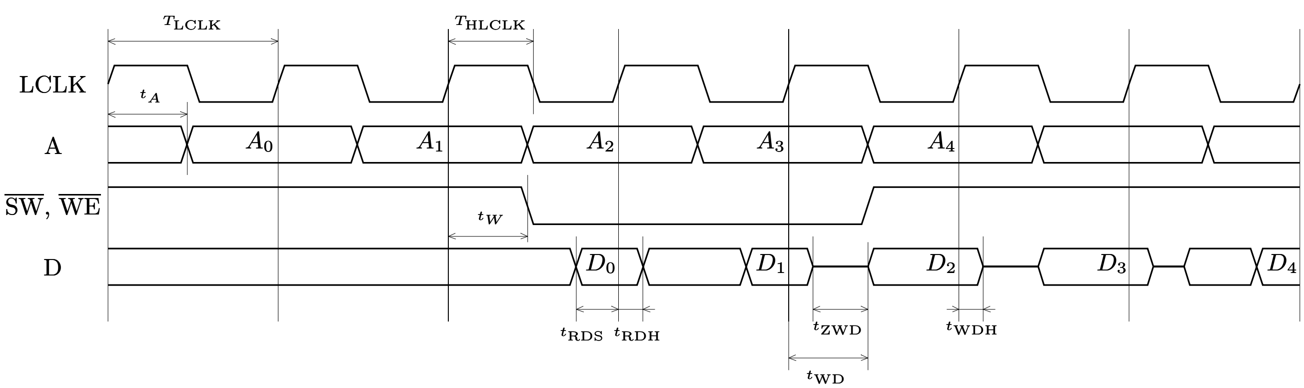
Power-On Reset Sequence



LBUS INTERFACE

The LBUS operates with pipelined zero-bus-turnaround SRAM. The SRAM is clocked at twice the internal LANai clock rate (twice the ICLK). See page 20 for definition of ICLK.

Pin	I/O	Description
LCLK, LCLK1	O	**LBUS Clock:** The SRAM-chip clock. There are two separate outputs that may be connected together, if a stronger clock driver is required (for example, when each SRAM chip's clock is driven by a separate, source-terminated line).
D00 - D63	I/O	**LBUS Data:** 64-bit bi-directional data bus (bit 63 is the most significant).
D0P - D7P	I/O	**LBUS Parity:** 8 data-parity bits, one for each byte in the 64-bit word. Bit D0P corresponds to bits D00 - D07, bit D1P to bits D08 - D15, *etc.*
A03 - A22	O	**LBUS Address:** The LBUS address pins. The internals of the LANai 7 chip support 32-bit addresses, but pin count limits the LBUS address space to 8 megabytes.
$\overline{\text{A20}}$	O	**LBUS Address Complement:** This signal is provided for address decoding when using multiple banks of SRAM chips.
$\overline{\text{WE0}}$ - $\overline{\text{WE7}}$	O	**LBUS Write Enable:** 8 write enable signals, one for each byte in the 64-bit word. The LANai is a big-endian machine, and $\overline{\text{WE0}}$ corresponds to the smallest byte address, *i.e.,* to the most-significant byte.
$\overline{\text{SW}}$	O	**LBUS Synchronous Write:** This signal is asserted if at least one of $\overline{\text{WE0}}$ - $\overline{\text{WE7}}$ is asserted.



Symbol	Parameter	Min	Max
T_{ICLK}	Period of the internal LANai clock (page 20).		
T_{LCLK}	Period of the LBUS SRAM clock.	$T_{ICLK}/2 - 0.5$ns	$T_{ICLK}/2 + 0.5$ns
T_{HLCLK}	LCLK high time	$T_{ICLK}/4 - 1$ns	$T_{ICLK}/4 + 1$ns
T_{SCLK}	CLOCK-register-controlled delay which ensures that the SRAM timing requirements are satisfied.	0ns	10ns
t_A	Address delay	T_{SCLK}	$T_{SCLK} + out_L + 0.5$ns
t_W	Write delay	T_{SCLK}	$T_{SCLK} + out_L + 0.5$ns
t_{RDS}	Read-data setup time	1ns	
t_{RDH}	Read-data hold time	0ns	
t_{WD}	Write-data delay (LANai limited)	T_{SCLK}	$T_{SCLK} + out + 1$ns
t_{ZWD}	Write-data delay (SRAM limited)		out
t_{WDH}	Write-data hold time	0.5ns	1ns

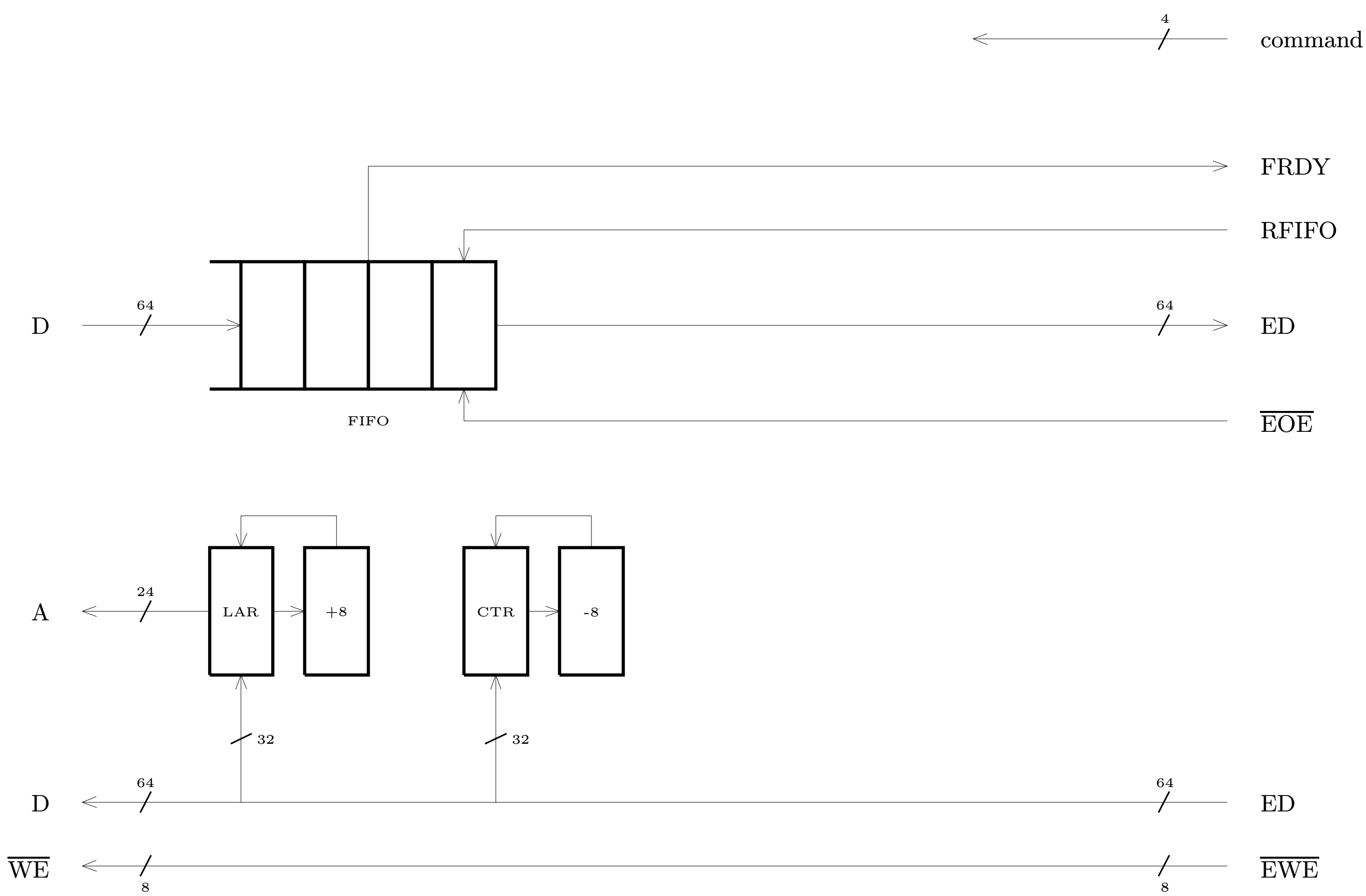
EBUS INTERFACE

The LANai 7 EBUS is a 64-bit wide, synchronous, pipelined interface, optimized for burst memory access.

The simplest EBUS operation is writing a value into a LANai special register. Since the LANai special registers have at most 32 bits, and the LANai address space is also 32 bits, the address and data are simultaneously presented on the 64-bit ED bus.

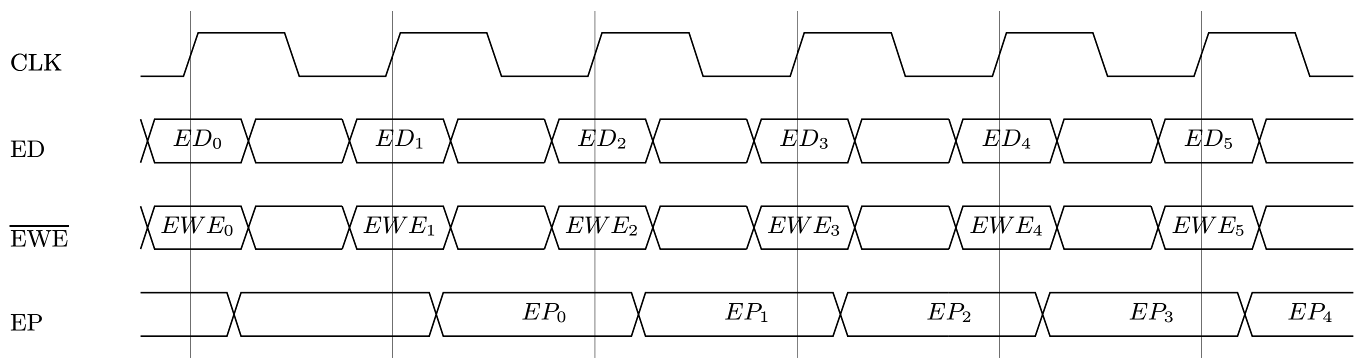
Writing into the LANai LBUS memory is a two-stage operation: first the address of the memory access is written into the special register LAR (LBUS Address Register), and later the 64 data bits are presented on the ED bus, along with the request that they be written into the memory at the address specified by the LAR. A side effect of the memory-write request is that the LAR is incremented by 8, so that consecutive memory locations can be written without having to write the LAR again.

The read accesses of special registers and LBUS memory are done in a similar fashion, but the read data becomes available with the latency of four clock cycles. To be able to sustain the full EBUS bandwidth, the read data is placed into the four-double-words-deep FIFO. The most distinguishing characteristic of the EBUS interface is that the consumption of data from the FIFO is independent of the read requests, as long as the FIFO is kept from overflowing. For one particular, often-used case of reading a block of data from the LBUS memory, the LANai provides a mechanism that can be used to keep the FIFO from overflowing (page 28). The LBUS Access Counter (CTR) is used to specify the size of the data block.

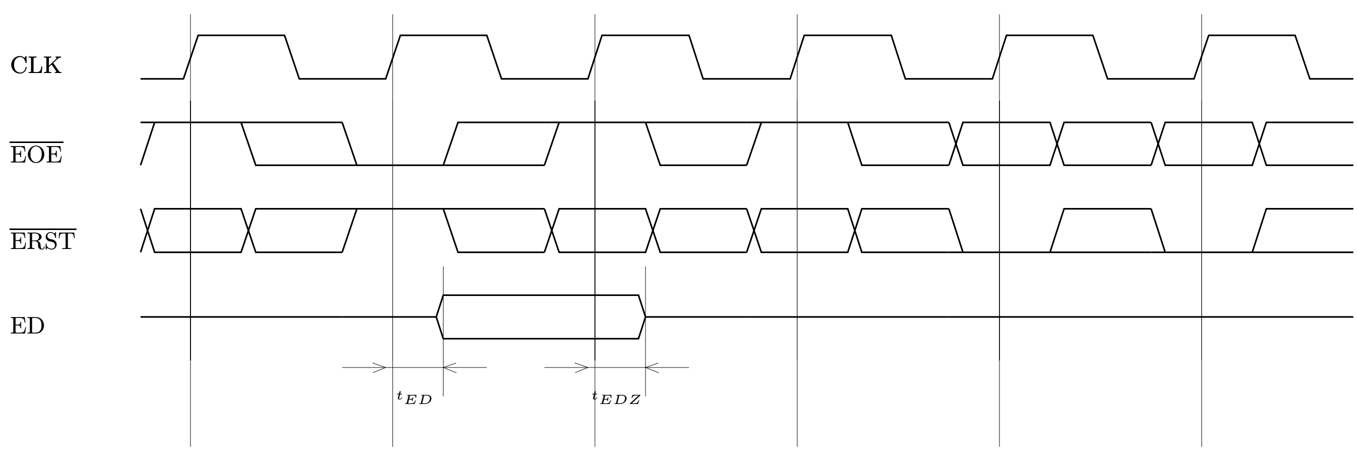


Pin	I/O	Description
$\overline{\text{EOE}}$	synch. I	**EBUS Output Enable:** This signal controls the direction of the ED bus. If, at a rising CLK edge, the $\overline{\text{EOE}}$ is asserted and the $\overline{\text{ERST}}$ is not, the ED pins are outputs during the immediately following clock cycle.
ED00 - ED63	synch. I/O	**EBUS Data:** 64-bit bi-directional address and data bus (bit 63 is the most significant).
$\overline{\text{EWE0}}$ - $\overline{\text{EWE7}}$	synch. I	**EBUS Write Enable:** 8 write enable signals, one for each byte in the 64-bit word. The LANai is a big-endian machine, and $\overline{\text{EWE0}}$ corresponds to the smallest byte address, *i.e.,* to the most-significant byte. The values of these signals affect only the Write Memory EBUS command.
EP	synch. O	**EBUS Parity:** This output is computed as the exclusive-or of the 64 EBUS data pins and 8 EBUS write enable pins.

EP Timing



ED Timing

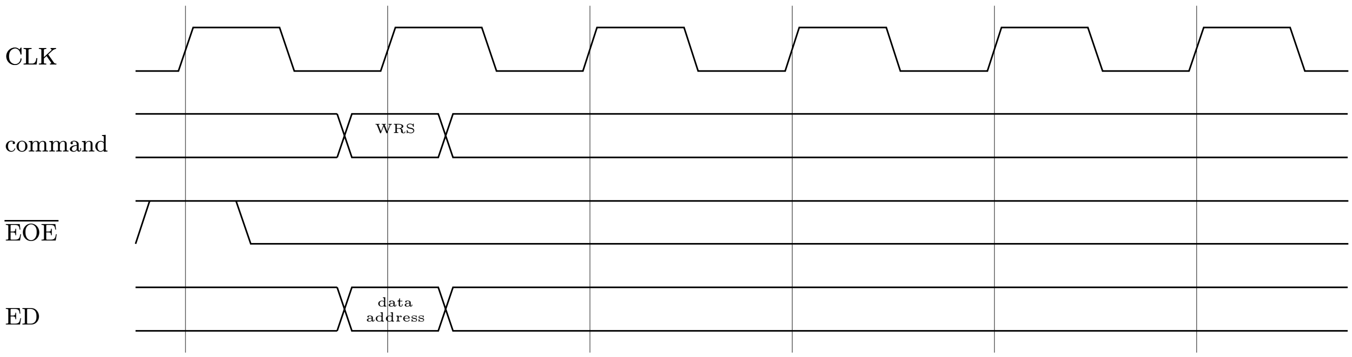


Symbol	Parameter	Min	Max
t_{EDZ}	Output release time	1ns	3ns
t_{ED}	Output delay	1ns	7ns

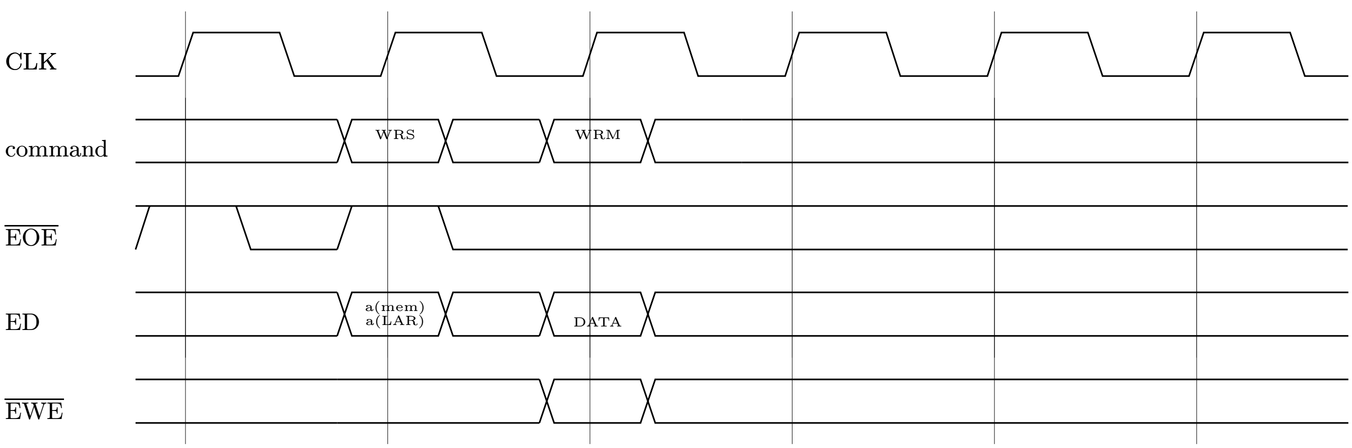
Pin	I/O	Description
WIN	O	**Chip Window:** This signal should be connected to a test point that can be used for timing observation of several internal chip signals. The special register DEBUG selects which internal signal is to be output on the WIN pin.
LED	synch. O	**LED Output:** This general-purpose output is controlled by the least-significant bit of the special register LED.
P0 P1 P2	synch. O	**PULSE Outputs:** These three outputs are controlled by the least-significant bits of the special register PULSE. When a value of 1 is written into such a bit, a one-clock-cycle-long pulse is generated on the corresponding output pin.
INT	synch. O	**Interrupt Request:** This output is asserted if a bit in the special register ISR (Interface Status Register) and the corresponding bit in the special register EIMR (External Interrupt Mask Register) are both equal to 1.
WAKE	synch. I	**Wakeup:** When this input is asserted, the wake_int bit in the special register ISR (Interface Status Register) is set.
FRDY	synch. O	**FIFO Ready:** This signal indicates that the LANai EBUS FIFO has at least one word in it.
RFIFO	synch. I	**Read FIFO:** When both RFIFO and FRDY are asserted at a rising CLK edge, the LANai EBUS FIFO is advanced.
SPEC DMA WMEM RMEM	synch. I	**Special, DMA, Write Memory, Read Memory:** These 4 signals together specify the EBUS command, as specified below.

Command	SPEC	DMA	WMEM	RMEM	Description
NOP	0	0	0	0	Null Operation.
WRS	1	0	1	0	**Write Special Register:** The 10 least-significant bits of the ED bus specify the address of the LANai special register to write. The 32 most-significant bits of the ED bus specify the data to be stored into the register (special registers have at most 32 data bits).
WRM	0	0	1	0	**Write Memory:** The 64 ED bits specify the data to be stored into the memory location pointed to by the LANai LBUS Address Register (LAR). The $\overline{\text{EWE}}$ bits enable writes into their corresponding bytes within the 64-bit word. The LAR is incremented by 8, and the LBUS Access Counter (CTR) is decremented by 8.
RDS	1	0	0	1	**Read Special Register:** The 10 least-significant bits of the ED bus specify the address of the LANai special register to read. A 64-bit word, with its most-significant 32 bits equal to the value of the special register, and its least-significant 32 bit undefined, is written into the FIFO. If the FIFO is full, the value written into it is lost.
RDM	0	0	0	1	**Read Memory:** The 64-bit value of the memory location pointed to by the LAR is written into the FIFO. The LAR is incremented by 8 and the CTR is decremented by 8. If the FIFO is full, the value written into it is lost.
START	0	1	0	1	**Start Read DMA:** This command instructs the LANai to fetch the block of data starting at the memory location pointed to by the LAR and of size specified by CTR, and write it into the FIFO. The LAR is incremented by 8 and the CTR is decremented by 8 as each 64-bit word is read from the memory. The LANai guarantees that the FIFO is not overwritten. No EBUS command (except for NOP and STOP) may be issued while the read DMA is going on.
STOP	0	1	0	0	**Stop Read DMA:** This command instructs the LANai to stop any pending Read DMA and flush the FIFO.
INIT	1	1	1	1	**CLOCK register initialization:** While the chip is held in reset (both $\overline{\text{RST}}$ and $\overline{\text{ERST}}$ asserted), issuing this command (typically, during power-on) initializes the CLOCK register to a value that, although it does not guarantee correct chip operation, does guarantee that the chip is properly reset. Writing the correct, system-specific value into the CLOCK register is still required before deasserting $\overline{\text{RST}}$ and/or $\overline{\text{ERST}}$. The INIT command is useful for ensuring that the LANai chip does not interfere with other circuitry until the system-specific value is available.

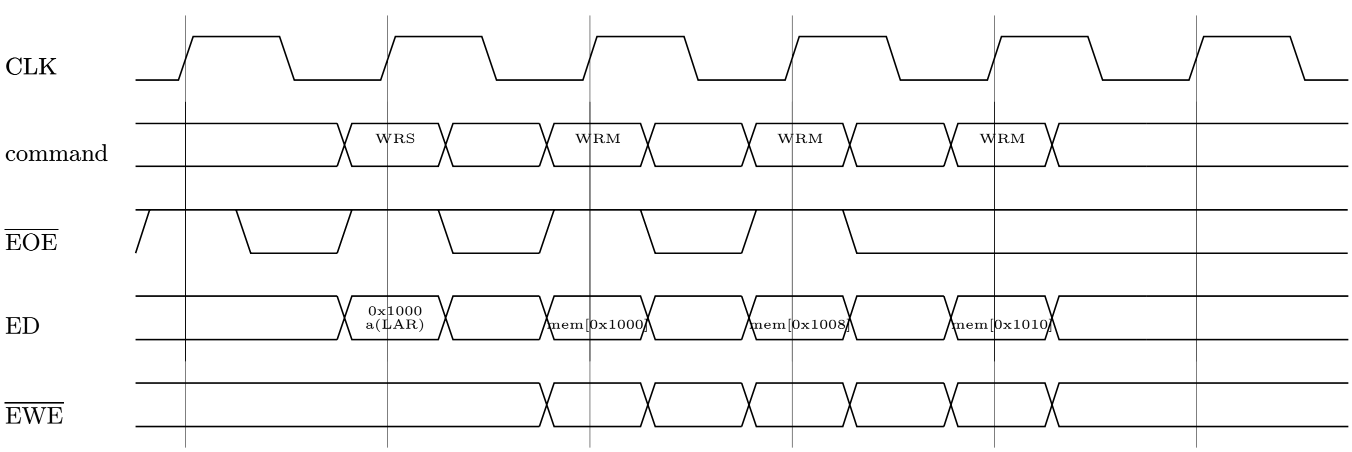
Write Special Register



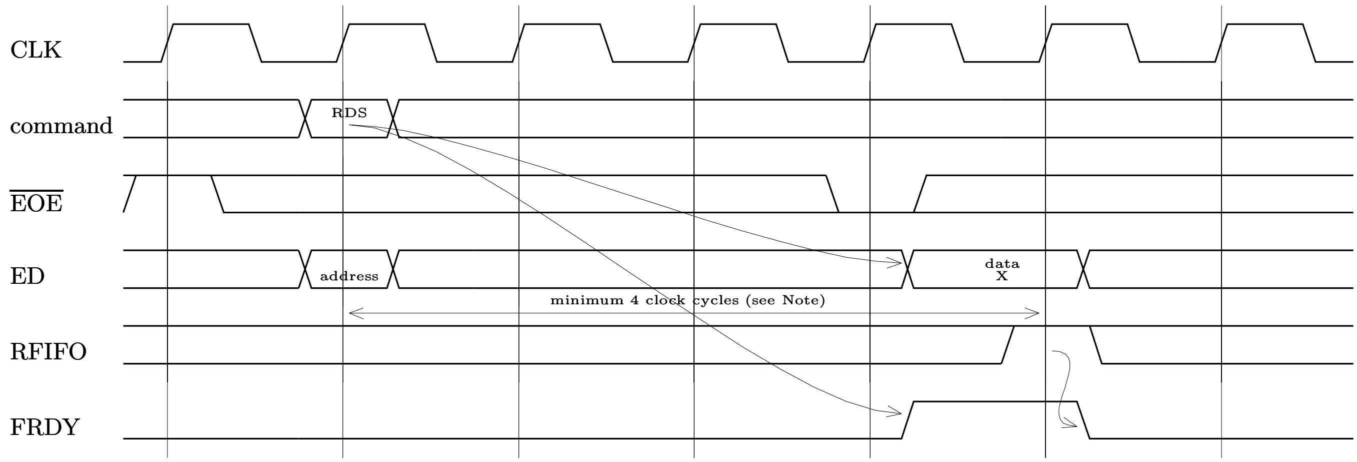
Write Single Word into the Memory



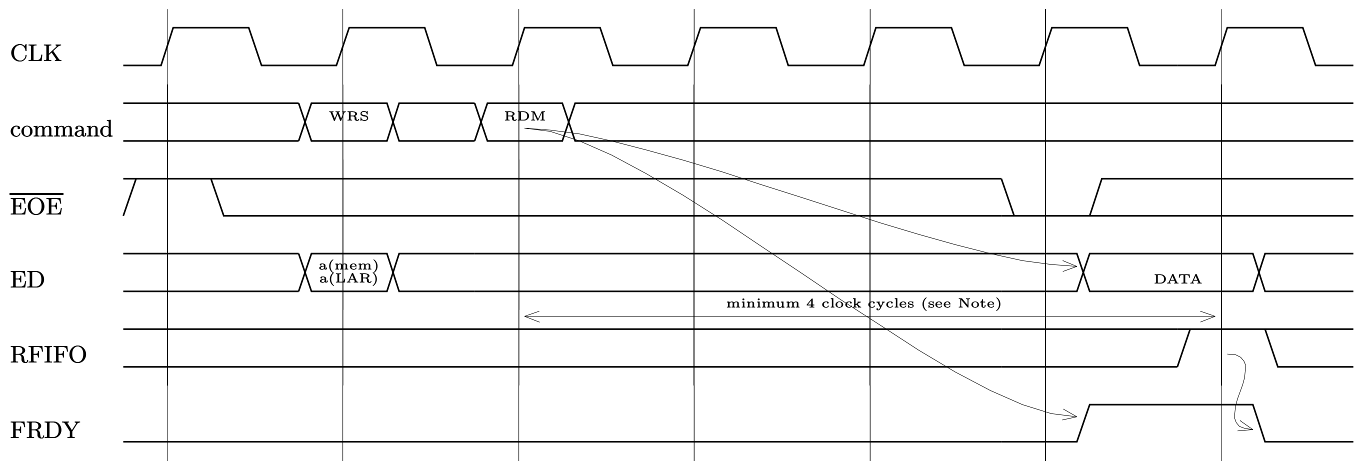
Write Successive Words into the Memory



Read Special Register

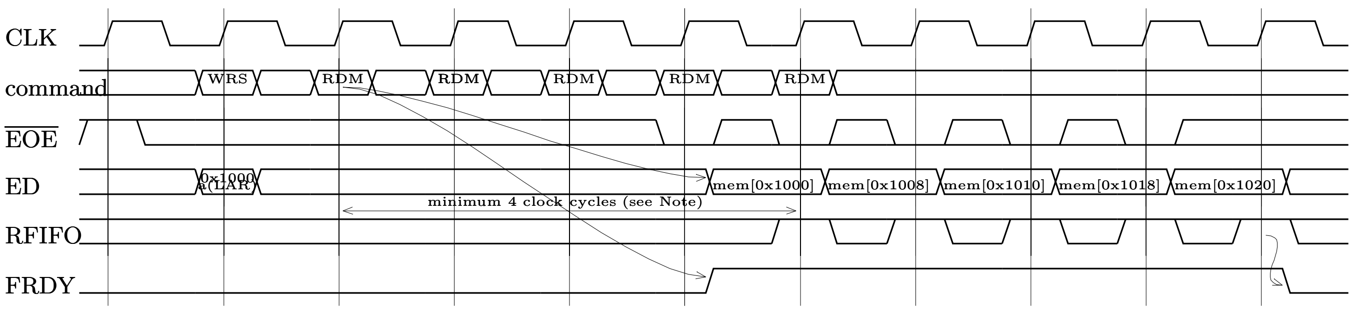


Read Single Word from the Memory

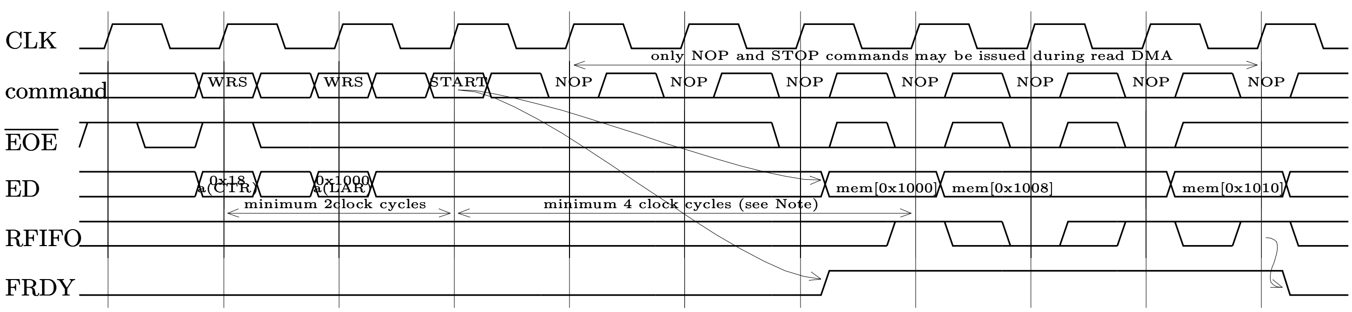


Note: When the internal-chip clock is 1.5 times the CLK, the read-access delay is equal to 3 CLK cycles. When the internal-chip clock is 2, 2.5, or 3 times the CLK, the read-access delay is equal to 2 CLK cycles. When the internal-chip clock is 4 times the CLK, the read-access delay is equal to 1 CLK cycle.

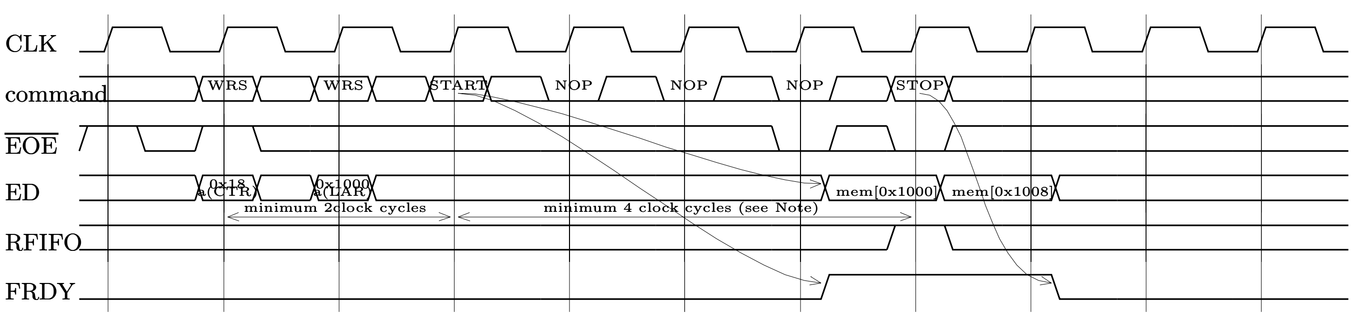
Read Successive Words from the Memory



Read DMA



Stop DMA



Note: When the internal-chip clock is 1.5 times the CLK, the read-access delay is equal to 3 CLK cycles. When the internal-chip clock is 2, 2.5, or 3 times the CLK, the read-access delay is equal to 2 CLK cycles. When the internal-chip clock is 4 times the CLK, the read-access delay is equal to 1 CLK cycle.

MYRINET SAN INTERFACE

	Pin	I/O	Description
Configuration	TCLK	I	**SAN Transmit Clock:** This clock input sets the data rate of the SAN output link. A byte is sent on every transition, so, for the nominal Myrinet SAN link rate of 160 megabytes per second, this is an 80 MHz clock. The allowed range for duty cycle is 45-55%. There is no restriction on the phase of TCLK with respect to any other clock (including TCLK on the other end of the SAN link). The frequency of TCLK must be within 1000ppm (0.1%) of 80MHz for compatibility with other Myrinet products.
	LVDD	power	**SAN Low-Voltage-Driver-Supply Voltage:** This pin should be connected to a 1.25V +/- 2% supply if it is to be compatible with other Myrinet SAN links. Power consumption varies with channel usage, 20mA minimum, 90mA maximum. If the SAN output drivers are shorted to GND, current draw can exceed 240mA.
	VTH	I	**SAN Input-Threshold Reference:** This reference input should be LVDD/2 $\pm$ 1%. Current draw is $1\mu A$ max.
	BIAS	I	**SAN Bias Reference:** For 3.3V operation, this pin should be fed 1.0mA. At that current level, the voltage on the pin will be approximately 1.4V. A 1.87KΩ resistor to 3.3V supply will achieve this.
	OAH	SAN O	**SAN Impedence Reference, High:** This pin should be connected to GND through a 50 ohm resistor.
	OAL	SAN O	**SAN Impedence Reference, Low:** This pin should be connected to LVDD through a 50-ohm resistor.
Input Channel	I0 - I7 ID	SAN I	**SAN Input:** The 8 data bits (I0-I7, I7 most significant) and the control bit (ID) are transition encoded. A transition on a data bit corresponds to the value of 1, no transition to the value of 0. A transition on the control bit corresponds to the data byte, no transition to the control symbol (see Myrinet SAN Link Specification). Each of these pins has a built-in 20KΩ pulldown resistor.
	OB	SAN O	**SAN Output Block:** This output is asserted to notify the connecting output SAN link channel that it must stop transmitting (see Myrinet SAN Link Specification).
Output Channel	O0 - O7 OD	SAN O	**SAN Output:** The 8 data bits (O0-O7, O7 most significant) and the control bit (OD) are transition encoded. A transition on a data bit corresponds to the value of 1, no transition to the value of 0. A transition on the control bit corresponds to the data byte, no transition to the control symbol (see Myrinet SAN Link Specification).
	IB	SAN I	**SAN Input Block:** When this input is asserted, the output channel stops transmitting (see Myrinet SAN Link Specification). This pin has a built-in 20KΩ pulldown resistor.
Monitoring	BCLK SCLK SDI SDO $\overline{\text{SLAT}}$	I I I O I	**SAN-Link-Monitor Interface:** The SAN interface on the LANai 7 chip includes several features that enable it to detect and report channel failure. These features require off-chip assistance, typically provided by a microcontroller with some associated circuitry (available from Myricom). These five pins provide the interface between the LANai 7 chip and the SAN-monitoring off-chip circuitry.

BUG LIST

Chip Version	Description
LANai7.0, 7.1	**EBUS Read DMA:** This mode of operation is not functional. The single-word read and the successive-word read operations are not affected.
LANai7.0, 7.1	**ISR _SIG BITS:** Under certain conditions outside programmer's control, if 1 is written into an ISR _sig bit by the LANai processor, the bit may behave as if it was written from the EBUS.