- **Provides up to 8K-Address Matching System**
- **Provides Glueless External-Address Match (EAM) Interface to the TNETX3150/TNETX3150A/TNETX3100**
- **Uses Standard Off-the-Shelf 15-ns SRAMs**
- **EEPROM Interface for Auto-Configuration, No CPU Needed**
- **Automatic Aging Through User-Selectable Aging (AGE) Threshold**
- **Fabricated in 3.3-V Low-Voltage Technology**
- **Requires 3.3-V and 5-V Power Supplies**
- **5-V Tolerant I/Os**
- **Provides Direct Input/Output (DIO) Interface for Management Access and Control (MAC) of the Address-Lookup Table**
- **Address Lookups, Adds, and Deletes Are Automatically Performed in Hardware**
- **User-Selectable Interrupts Simplify the Management Operations**
- **Secure Addresses From Changing Ports**
- **MII Data I/O (MDIO) PHY Management Interface**
- **Management Access to Statistic Registers**
- **JTAG Compliant**
- **Packaged in 144-Terminal Plastic Quad Flatpack**

description

The TNETX15AE performs the address-lookup function for either the TNETX3150/TNETX3150A or the TNETX3100 Ethernet™ switch. The TNETX15AE device determines the addresses to use and match from the TNETX3150/TNETX3150A/TNETX3100 DRAM bus. The address-lookup table is maintained in external SRAM. The frame-matching and routing information is given to the TNETX3150/TNETX3150A/TNETX3100 through the external address-matching (EAM) interface.

The TNETX15AE is designed to work in either unmanaged or managed mode. Unmanaged operation is accomplished through EEPROM support. Startup options are auto-loaded into the TNETX15AE registers using the EEPROM interface. If the TNETX15AE is initialized by automatic loading from the EEPROM, it begins operation automatically.

All functions of this device are fully controllable by management through a direct input/output (DIO) interface. In addition, this device can interrupt the external management processor with user-selectable interrupts. This device also provides support for easy management control of IEEE Std 802.3u media-independent interface (MII) managed devices. A typical application is shown in Figure 1.

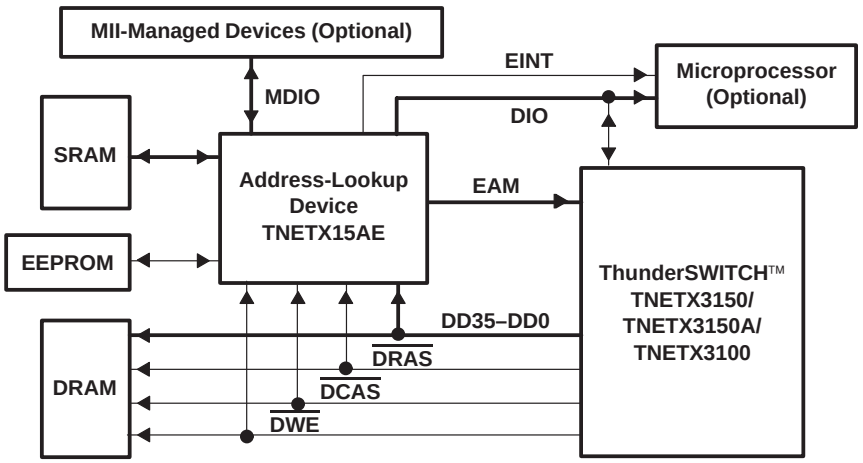


Figure 1. TNETX15AE Typical Application

PGE PACKAGE
(TOP VIEW)

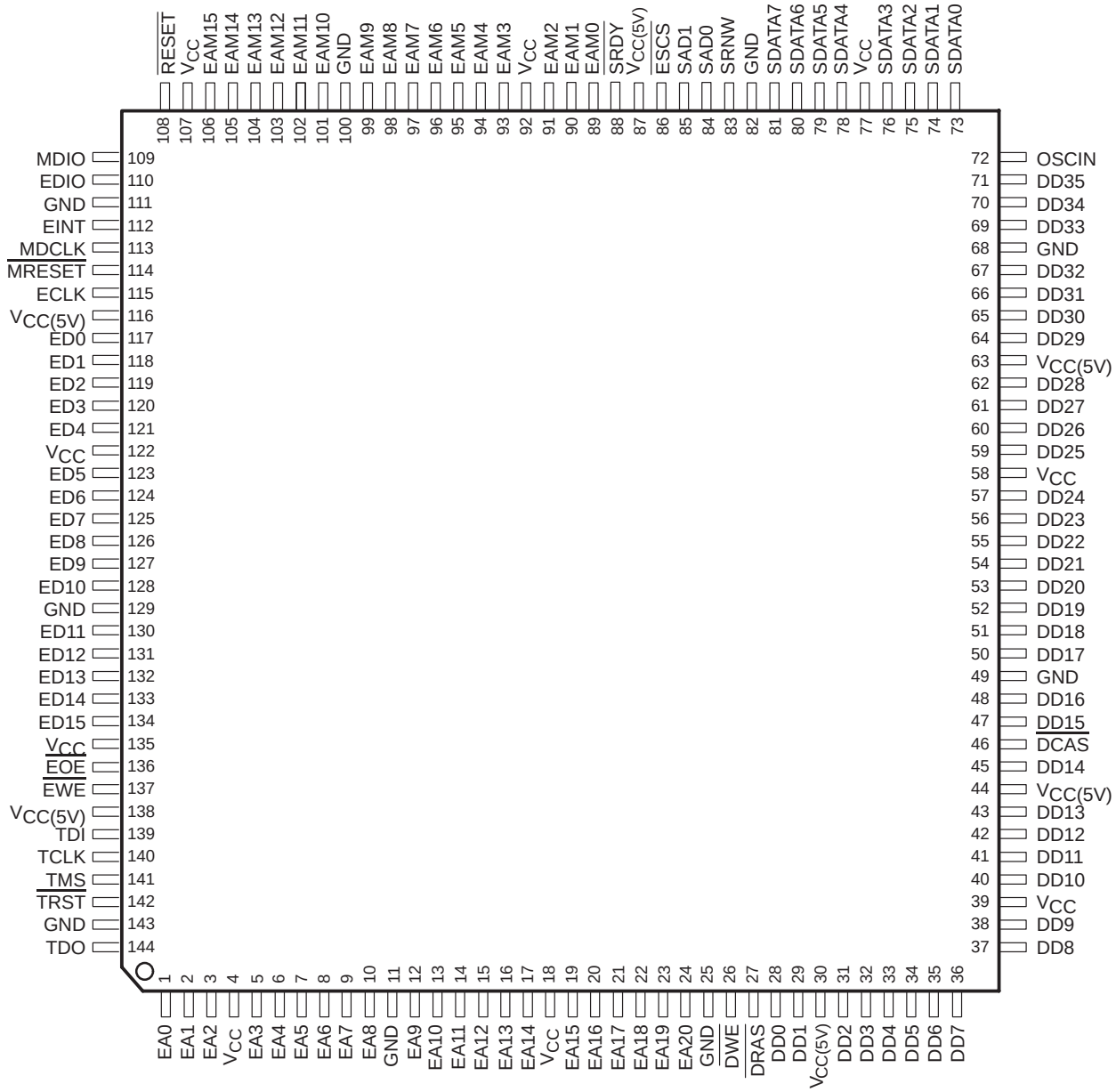
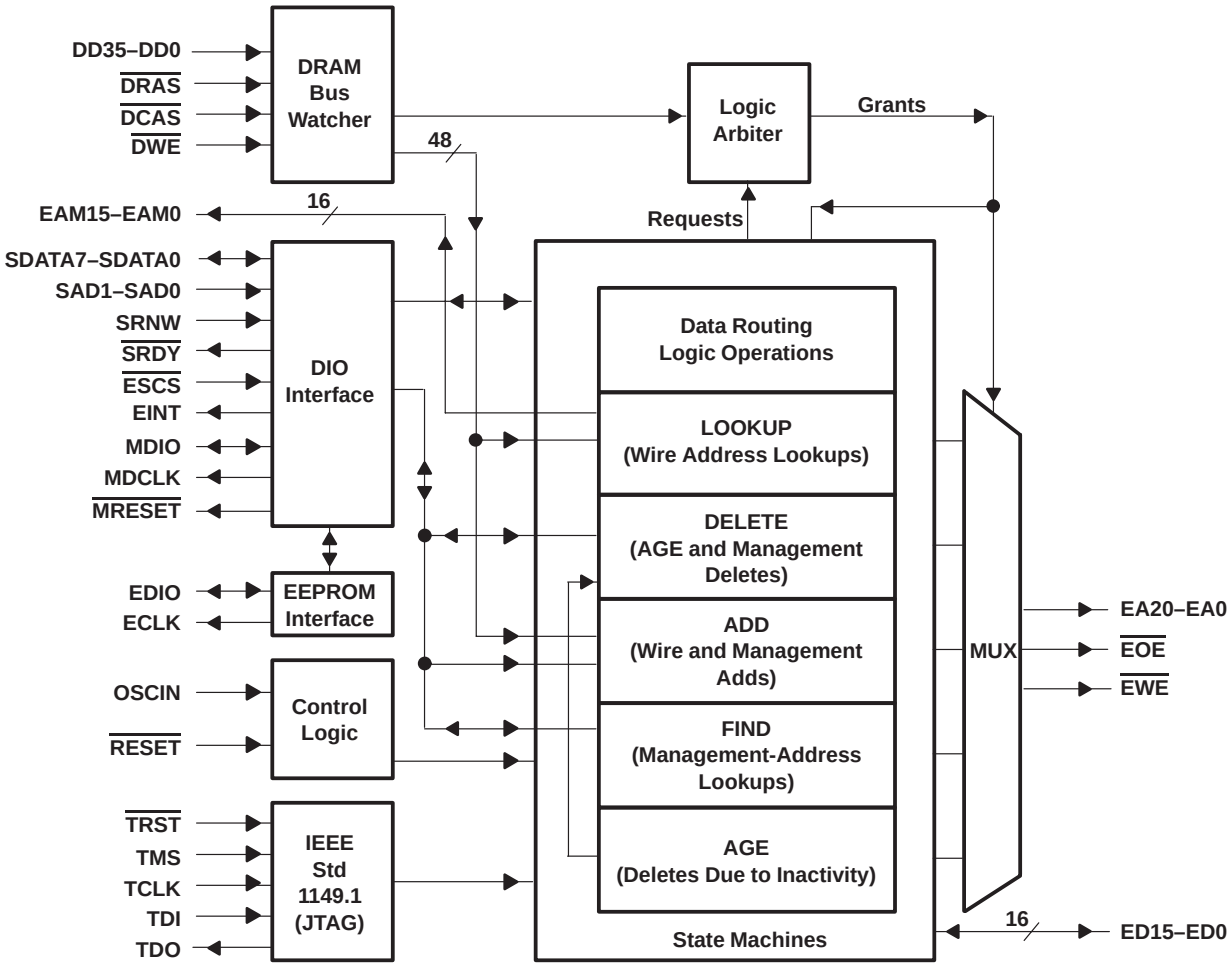


Table of Contents

Functional Block Diagram 4
General Description 5
Bus Watcher 5
DIO Interface 5
EEPROM Interface 5
Arbiter 5
SRAM Address/Data Multiplexer (MUX) 5
TNETX15AE Logic Operations 5
IEEE Std 1149.1 Test Port (JTAG) 5
Terminal Functions 6
Register Description 11
General Notation Notes 11
Node Address Format (Ethernet Native Data Format) 11
DIO Register Access 11
DIO Address Register DIOADR 12
DIO Data Register DIODATA 12
DIO Address Auto-Increment Register 12
Internal Register Map 13
Default Register Values at Reset 14
EEPROM Auto-Configuration
From an External x24C02 EEPROM 14
Internal Register Descriptions 17
Revision Register 17
SRAM Size-Select Register 17
Age-Deletion Time-Select Register 18
Unknown Unicast Port-Routing Register 18
Unknown Multicast Port-Routing Register 19
Mode Control Status Register 19
Serial Interface I/O Register 21
Management-Table Lookup Interface 22
Lookup-Node Address Register 22
Lookup Routing-Code Register 23
Node Age-Stamp Register 24
Lookup Table Search Control Register 24
Statistics Registers 25
SRAM Address Register 25
Manufacturing Test Register 26
SRAM Data Register 26
Interrupt Register 27
Interrupt Masking Register 28
New-Add Aged-Del Register Set 29
New-Node/Port-Change/Security-Violation
Interrupt Interface 29
Agednode Interrupt Interface Group 30
Management Add/Edit Address Interface Group 31
Management Delete Address Interface Group 33
Uplink Routing Register 34
Principles of Operation 34
Internal Operation 34
EAM Codings and In-Order Broadcasts (IOB) 34
TNETX15AE DRAM and EAM Interface 35
Decoding the Destination Address (DA)
and Source Address (SA) From the DRAM Bus 36
CRC Checking and Valid Frames 37
Operating Logic Arbitration 37
Lookup Algorithm 38
A Graphical Example of the Lookup Algorithm 40
SRAM Data Storage 42
Lookup Table SRAM Allocation 42
RAMsize and Number of Nodes Supported 44
Register Spaces/External Devices
Accessible Through the TNETX15AE 44
DIO Interface 45
DIO Write Cycle 46
DIO Read Cycle 47
Host (Access) Register Space 47
Internal Registers 49
Initialization 50
TNETX3150/TNETX3150A/TNETX3100 Initialization 50
Resetting the TNETX15AE 50
Hardware Reset 50
Software Reset 51
EEPROM Initialization/Automatic Loading (LOAD) 51
TNETX15AE Operational Modes 53
NAUTO mode 53
NCRC mode 53
NRLN0 mode 53
Lookup Table SRAM Initialization (START) 54
Frame-Forwarding LKUP Logic Operation 54
Management-Based Lookups (FIND) 54
FIND, FINDFIRST, and FINDNEXT Commands 55
Summary of FIND Operations Supported 56
Adding an Address 56
Interrupts Available to Support Both Wire ADDs
and Host ADDs 56
Management Address Adding 57
Adding Unicast and Multicast Addresses 58
Deleting an Address 58
Aging (AGE Logic Operation) 58
DEL Logic Operation (Management Address Deletions) . . 60
Interrupts 60
Masking Interrupts 61
Test Interrupts (INT) 61
ADD Interrupts 61
Aging Interrupts (AGE, AGEM) 62
Statistic Interrupt (STAT) 62
IEEE Std 1149.1 Test-Access Port (JTAG) 63
Absolute Maximum Ratings 64
Recommended Operating Conditions 64
Electrical Characteristics 65
Timing Requirements, External SRAM Read Cycle 65
Operating Characteristics, External SRAM Read Cycle 65
Timing Requirements, External SRAM Write Cycle 66
Operating Characteristics, External SRAM Write Cycle 66
Operating Characteristics, EAM Routing Code 67
Timing Requirements, DRAM Interface 68
Timing Requirements, DIO Read Cycle 69
Operating Characteristics, DIO Read Cycle 69
Timing Requirements, DIO Write Cycle 70
Operating Characteristics, DIO Write cycle 70
Timing Requirements, OSCIN clock 71
Timing Requirements, Power-On Reset 71
Timing Requirements, RESET (Software) Timing 72
Operating Characteristics 73
EEPROM and MDIO (MII) Interfaces 73
EEPROM Interface Timing 73
Parameter Measurement Information 74
Test Measurement 74
Application Information 75
TNETX15AE/TNETX3150/TNETX3150A
Stand-Alone Applications 75
Unmanaged Switch 75
Managed Switch 75
Mechanical Data 77

functional block diagram



general description

bus watcher

The bus watcher interfaces to the TNETX3150/TNETX3150A/TNETX3100 DRAM interface and extracts destination, source addresses, and the originating port number. The bus watcher is responsible for identifying the frame's start-of-frame and end-of-frame tags. It also interfaces to the arbiter and the TNETX15AE logic to perform off-the-wire address LOOKUPs and ADDs.

DIO interface

The DIO interface enables an attached microprocessor to access the TNETX15AE internal registers. The DIO interface is used to select control modes, read statistics, receive interrupts, read/write to attached MII devices, read/write to an attached EEPROM, and perform management LOOKUPs, ADDs, and DELETEs.

EEPROM interface

The EEPROM interface allows accesses to the EEPROM. It also interfaces with the EEPROM for automatic loading of selected registers from the EEPROM at startup or reset.

arbiter

The arbiter manages the SRAM access for the TNETX15AE logic operations by assigning priorities to the logic operations. Wire lookups have the highest priority, followed by DELETEs, ADDs, management LOOKUPs, and AGEs. The individual logic operations request the bus by asserting a request signal. The arbiter grants the SRAM bus by controlling the SRAM-bus address/data multiplexer.

SRAM address/data multiplexer (MUX)

The address/data MUX is controlled by the arbiter and selects the logic operation that has ownership of the SRAM bus.

TNETX15AE logic operations

The TNETX15AE logic operations consist of LOOKUP, DELETE, ADD, FIND, and AGE operations. Each logic operation is assigned a priority on the SRAM bus and is controlled by the arbiter. The LOOKUP operation has the highest priority and is responsible for wire lookups. The DELETE operation is responsible for either deletes from the AGE operation or for management-delete requests. The ADD operation is responsible for wire adds as well as for management-add requests. The FIND operation is responsible for management searches of the lookup table. The AGE operation is responsible for deleting addresses that have no activity in a fixed time period.

IEEE Std 1149.1 test port (JTAG)

The test-access port is composed of five terminals that are used to interface serially with the device and the board on which it is installed for boundary-scan testing. The TNETX15AE is fully JTAG compliant, with the exception of requiring external pullup resistors on terminals TDI, TMS, and $\overline{\text{TRST}}$.

Terminal Functions

EAM interface

<table>
<tr><th colspan="2">TERMINAL</th><th>I/O</th><th>DESCRIPTION</th></tr>
<tr><th>NAME</th><th>NO.</th><th></th><th></th></tr>
<tr><td>EAM15†
EAM14
EAM13
EAM12
EAM11
EAM10
EAM09
EAM08
EAM07
EAM06
EAM05
EAM04
EAM03
EAM02
EAM01
EAM00‡</td><td>106
105
104
103
102
101
99
98
97
96
95
94
93
91
90
89</td><td>O</td><td>
External address match single-/multiple-port routing code select. When EAM15 is high, the EAM-interface terminals contain a single-port routing code. When EAM15 is low, the EAM-interface terminals contain a multiple-port routing code.

External address match port routing code select. When EAM15 is high, the EAM14–EAM0 terminals are placed in the single-port mode. In this mode EAM14–EAM0 terminals encode a single port to which the frame is routed.

EAM14–EAM5 are don't-care bits and are set to 0. The single-port codes (EAM15–EAM0 terminals) are shown in the following:
<table>
<tr><th rowspan="2">TNETX3150/TNETX3100</th><th colspan="17">EAM15–EAM0</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th></th></tr>
<tr><td>Port 00 (uplink)</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>b</td></tr>
<tr><td>Port 01</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>b</td></tr>
<tr><td>Port 02</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>b</td></tr>
<tr><td>Port 03</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>b</td></tr>
<tr><td>Port 04</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>b</td></tr>
<tr><td>Port 05</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>b</td></tr>
<tr><td>Port 06</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>b</td></tr>
<tr><td>Port 07</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>b</td></tr>
<tr><td>Port 08</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>b</td></tr>
<tr><td>Port 09</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>b</td></tr>
<tr><td>Port 10</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>b</td></tr>
<tr><td>Port 11</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>b</td></tr>
<tr><td>Port 12</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>b</td></tr>
<tr><td>Port 13</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>b</td></tr>
<tr><td>Port 14</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>b</td></tr>
<tr><td>Reserved</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>b</td></tr>
<tr><td>No-operation code</td><td>1</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>1</td><td>x</td><td>x</td><td>x</td><td>x</td><td>b</td></tr>
</table>
The no-operation code signals that the EAM device does not take part in forwarding decisions on this packet. The TNETX15AE does not put out this code in normal operations. To successfully disable the TNETX15AE, it is reset and not started. This causes the EAM interface to float, and external pullup resistors on EAM15 and EAM04 (minimally) to supply the no-operation code to the TNETX3150/TNETX3150A/TNETX3100 device. Erratic operation occurs if the TNETX3150/TNETX3150A/TNETX3100 address matching is not enabled when the TNETX15AE is disabled. The TNETX15AE, when enabled, overrides the operations of the TNETX3150/TNETX3150A/TNETX3100 matching circuitry; but, it is strongly advised to disable the TNETX3150/TNETX3150A/TNETX3100 address matching if the TNETX15AE is active. This prevents two logic blocks from attempting to control port suspension and other functions.

When EAM15 is low, the EAM14–EAM0 terminals encode the multiple ports to which the data frame is routed. Each bit on the EAM14–EAM0 bus is coded as follows:
– 1 on the bit signifies that the data frame is routed to that port.
– 0 on the bit signifies that the data frame is not routed to that port.

Terminal number assignments have a one-to-one correspondence to the port numbers as shown in the following:
<table>
<tr><th>EAM Bus Bit</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>Port Number</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
</table>
For example, to multicast a frame to ports 08, 05, 02, and 00 a multiport code of 0000 0001 0010 0101 or 0x0125 is sent.

All multicast packets and unknown unicast packets are forwarded by multiport code (EAM15 = 0) even if only one bit is set.

To discard a frame, a code of 0x0000 is output (multiple routing code with no ports selected):
<table>
<tr><th>TNETX3150/TNETX3100</th><th>EAM14–EAM0</th></tr>
<tr><td>Discard frame</td><td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td></tr>
</table>
</td></tr>
</table>

† Most-significant bit
‡ Least-significant bit

Terminal Functions (Continued)

DRAM interface

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
DD35†	71	I	DRAM data bus. Data bus is sourced by the TNETX3150/TNETX3150A or the TNETX3100.
DD34	70		
DD33	69		
DD32	67		
DD31	66		
DD30	65		
DD29	64		
DD28	62		
DD27	61		
DD26	60		
DD25	59		
DD24	57		
DD23	56		
DD22	55		
DD21	54		
DD20	53		
DD19	52		
DD18	51		
DD17	50		
DD16	48		
DD15	47		
DD14	45		
DD13	43		
DD12	42		
DD11	41		
DD10	40		
DD9	38		
DD8	37		
DD7	36		
DD6	35		
DD5	34		
DD4	33		
DD3	32		
DD2	31		
DD1	29		
DD0‡	28		
$\overline{\text{DCAS}}$	46	I	DRAM column address select. $\overline{\text{DCAS}}$ is sourced by the TNETX3150/TNETX3150A or the TNETX3100.
$\overline{\text{DRAS}}$	27	I	DRAM row address select. $\overline{\text{DRAS}}$ is sourced by the TNETX3150/TNETX3150A or the TNETX3100.
$\overline{\text{DWE}}$	26	I	DRAM write enable. $\overline{\text{DWE}}$ is sourced by the TNETX3150/TNETX3150A or the TNETX3100.

† Most-significant bit
‡ Least-significant bit

Terminal Functions (Continued)

external SRAM interface

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
EA20†	24	O	External address bus. EA20–EA0 is the external SRAM address bus.
EA19	23		
EA18	22		
EA17	21		
EA16	20		
EA15	19		
EA14	17		
EA13	16		
EA12	15		
EA11	14		
EA10	13		
EA9	12		
EA8	10		
EA7	9		
EA6	8		
EA5	7		
EA4	6		
EA3	5		
EA2	3		
EA1	2		
EA0‡	1		
ED15†	134	I/O	External data bus. ED15–ED0 is the external SRAM data bus.
ED14	133		
ED13	132		
ED12	313		
ED11	130		
ED10	128		
ED9	127		
ED8	126		
ED7	125		
ED6	124		
ED5	123		
ED4	121		
ED3	120		
ED2	119		
ED1	118		
ED0‡	117		
$\overline{\text{EOE}}$	136	O	External output enable. $\overline{\text{EOE}}$ is the active-low external SRAM output-enable signal.
$\overline{\text{EWE}}$	137	O	External write enable. $\overline{\text{EWE}}$ is the active-low external SRAM write-enable signal.

† Most-significant bit
‡ Least-significant bit

Terminal Functions (Continued)

DIO interface-statistics

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
EINT	112	0	TNETX15AE interrupt. EINT is an interrupt request from the TNETX15AE to an optional attached microprocessor. Its purpose is to reduce the polling that is required for the processor to be current on the operational states of the TNETX15AE. The interrupt cause is determined by reading the interrupt register (0x28, 0x29). It is not required to acknowledge or act on the interrupts (unmanaged operation). EINT is active high and level sensitive; it is high when any of the interrupt triggers are active.
$\overline{\text{ESCS}}$	86	I	TNETX15AE chip select. When $\overline{\text{ESCS}}$ is active (low), a port access is valid for the TNETX15AE device. $\overline{\text{ESCS}}$ must not be tied to any other chip-select signal (such as the TNETX3150/TNETX3150A $\overline{\text{SCS}}$ or the TNETX3100 $\overline{\text{SCS}}$).
SAD1 SAD0	85 84	I	DIO address bus. SAD1–SAD0 selects the TNETX15AE host registers. **SAD1 SAD0 Description** 0 0 DIO address low 0 1 DIO address high 1 0 DIO data 1 1 DIO data auto increment
SDATA7† SDATA6 SDATA5 SDATA4 SDATA3 SDATA2 SDATA1 SDATA0‡	81 80 79 78 76 75 74 73	I/O	DIO data bus. SDATA7–SDATA0 is a byte-wide bidirectional DIO bus.
$\overline{\text{SRDY}}$	88	O	DIO ready. When $\overline{\text{SRDY}}$ is active low, the following conditions occur: – When SRNW = 1, the interface host is told that data is valid for reading by the host. – When SRNW = 0, the interface host is told that data has been received by the TNETX15AE. When $\overline{\text{ESCS}}$ is inactive high, $\overline{\text{SRDY}}$ is disabled [high-impedance (Z) state].
SRNW	83	I	DIO read/not write. SRNW is a read- or write-select signal. – When SRNW is high, the read operation is performed by the host. – When SRNW is low, the write operation is performed by the host.

† Most-significant bit
‡ Least-significant bit

DIO interface-MII

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
MDCLK	113	O	MII management data clock. MDCLK is a clock from the TNETX15AE for the serial MII management data MDIO. MDCLK can be disabled [high-impedance (Z) state] by MDIOEN [a bit in the SIO register at 0x0A (DIO)].
MDIO	109	I/O	MII management data input/output. MDIO is the data terminal for serial MII management data. MDIO requires an external pullup resistor for proper operation. MDIO can be disabled [high-impedance (Z) state] by MDIOEN [a bit in the SIO register at 0x0A (DIO)].
$\overline{\text{MRESET}}$	114	O	MII management reset. $\overline{\text{MRESET}}$ is a reset signal for the serial MII management interface. $\overline{\text{MRESET}}$ can be disabled [high-impedance (Z) state] by MDIOEN [a bit in the SIO register at 0x0A (DIO)].

Terminal Functions (Continued)

DIO interface-EEPROM

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
ECLK	115	O	EEPROM clock. ECLK is the serial EEPROM data clock and is implemented as a 3.3-V totem-pole output cell. Because of the active pullup, and since most EEPROMs use CMOS logic levels, it may be difficult to use 5-V EEPROMs.
EDIO	110	I/O	EEPROM data input/output. EDIO is the serial EEPROM data I/O signal. This terminal requires an external pullup (see EEPROM data sheet) for EEPROM operation.

control logic interface

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
OSCIN	72	I	Oscillator input. OSCIN is the clock input that drives the internal state machines. For no-glue interface to the TNETX3150/TNETX3150A/TNETX3100, OSCIN is connected to the TNETX3150/TNETX3150A/TNETX3100 OSCIN terminal.
$\overline{\text{RESET}}$	108	I	Reset. $\overline{\text{RESET}}$ (active low) resets the TNETX15AE. $\overline{\text{RESET}}$ must remain active for at least 25 ms after power and the clocks have stabilized on power up. After the power-up cycle, $\overline{\text{RESET}}$ must remain active for a minimum of 3 μs for recognition.

JTAG interface

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
TCLK	140	I	Test clock. TCLK clocks state information and test data into and out of the device during operation of the test port.
TDI	139	I	Test data input. TDI serially shifts test data and test instructions into the device during operation of the test port.
TDO	144	O	Test data out. TDO serially shifts test data and test instructions out of the device during operation of the test port.
TMS	141	I	Test mode select. TMS controls the state of the test-port controller.
$\overline{\text{TRST}}$	142	I	Test reset. $\overline{\text{TRST}}$ asynchronously resets the test-port controller when set low.

power interface

TERMINAL NAME	TERMINAL NO.	DESCRIPTION
GND	11, 25, 49, 68, 82, 100, 111, 129, 143	Ground. GND is the 0-V reference for the device.
V_{CC}	4, 18, 39, 58, 77, 92, 107, 122, 135,	Supply voltage. V_{CC} = 3.3 ± 0.3 V. V_{CC} is the main-logic power supply.
$V_{CC(5V)}$	30, 44, 63, 87, 116, 138	Supply voltage. $V_{CC(5V)}$ = 5 ± 0.5 V. $V_{CC(5V)}$ is the clamp rail voltage on I/O terminals that gives TTL capability.

register description

general notation notes

Hexadecimal notation: Leading zero, followed by lower-case x, then hex digits, with space or punctuation delimiters on both ends. A . . . F are equivalent to a . . . f (example: 0x12bc, 0x12BC0

Binary notation: Trailing lower-case b after string composed of only 1 or 0, with space or punctuation delimiters on both ends (example: 101011b).

Decimal notation: A number composed of 0 . . . 9 digits with no prefix or suffix, with a space or punctuation delimiters on both ends (example: 23965)

The least-significant bit in a register is the lowest number.

The least-significant byte of a multibyte value is the lowest number.

node address format (Ethernet native data format)

The node addresses are kept inside the TNETX15AE in Ethernet native data format. This is the same format used when the address appears in the frame transmitted on the wire. Ethernet first transmits the least-significant bit of a data byte on the wire. This makes the group/specific address appear in bit 40 (least-significant bit of most-significant byte). The TNETX15AE address storage format is shown in the following:

BIT							
BYTE (most-significant byte)		BYTE	BYTE	BYTE	BYTE	BYTE	BYTE
47 42	41	40	39 32	31 24	23 16	15 8	7 0
	Local/Universal Address	Group/Specific Address					

DIO register access

There are only four byte-wide DIO registers that are hardware interfaced to the DIO interface. These registers are treated like a pointer structure to locate internal registers, some of which are interfaced to external terminals of the device (see Figure 2). The small number of registers actually on the DIO interface give the TNETX15AE a small I/O map requirement when designing systems; also, it allows additional registers to be defined without requiring the hardware interface to change.

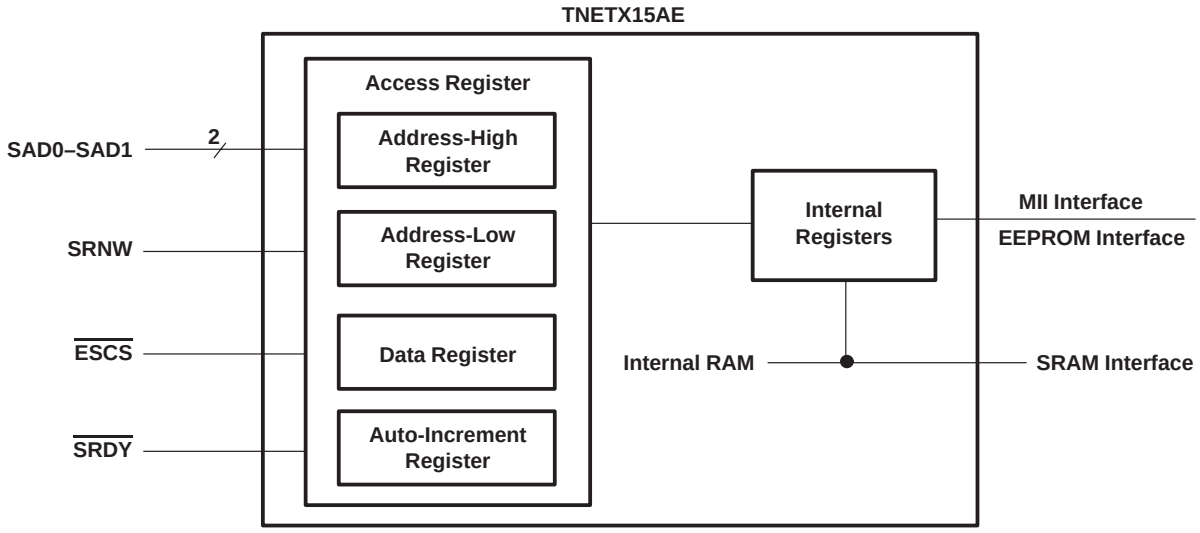


Figure 2. DIO Access (Pointer) Registers

DIO register access (continued)

To access an internal register, post both halves of the address to the DIO address-low and DIO address-high registers and read or write the DIO data register. The DIO access register code is given in Table 1. If the host performs a read, the internal state machine enables the internal register at the posted address to drive the bus when the read cycle is active. If the host performs a write to a TNETX15AE register, the value written to the data register address is transferred to the posted address in the DIO address-low and DIO address-high registers. $\overline{\text{SRDY}}$ (on a host read) indicates that the internal fetch was accomplished. A delay of up to 60 ns after cycle start is encountered if the resource being requested is currently being used by an internal state machine. There are dedicated cycles for DIO to prevent lockout, but they are allocated as 2 of 5, and DIO is asynchronous to internal processes. $\overline{\text{SRDY}}$ (on a host write) indicates that the value written to the data register is captured and is written to the resource addressed. For maximum throughput, $\overline{\text{SRDY}}$ should be part of the interface; if not, add 60 ns to the minimum DIO cycle times.

If a range of consecutive (rising in order) accesses is needed, reading or writing to the DIO data auto-increment register causes a post increment of the DIO address low and DIO address high registers after each access. Reads and writes can be mixed in a string of accesses; the post increment is by one after each cycle.

All registers are set to their default values on a hardware reset (setting the $\overline{\text{RESET}}$ terminal low, then high). All registers, except the control register, also are set to their default values on a software reset (setting the RESET bit high in the control register).

Table 1. DIO Access Register Code

SAD1	SAD0	DESCRIPTION
0	0	DIO address low
0	1	DIO address high
1	0	DIO Data
1	1	DIO data auto-increment

DIO address register DIOADR at SAD1–SAD0 = 00b and 01b (host)

DIO ADDRESS HIGH								DIO ADDRESS LOW							
15†	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0‡
DIOADR															

† Most-significant bit
‡ Least-significant bit

BIT	NAME	FUNCTION
15–0	DIOADR	DIO address. DIOADR contains the internal DIO address used on subsequent accesses to the DIO data or DIO data auto-increment registers. This field auto-increments (by one) on all accesses to the DIO data auto-increment register.

DIO data register DIODATA at SAD1–SAD0 = 10b (DIO)

The DIO data register address allows indirect access to internal TNETX15AE registers. This is a temporary staging register for data being written through to an internal register (write) or fetched from an internal register (read).

DIO address auto-increment register DIOADDRINC at SAD1–SAD0 = 11b (DIO)

The DIO address auto-increment register address allows indirect access to internal TNETX15AE registers. This is a temporary staging register for data being written through to an internal register (write) or fetched from an internal register (read). Accesses to this register cause a post increment of the DIO address register.

internal register map

The DIO addresses for the internal registers are shown in Table 2.

Table 2. DIO Internal Register Address Map

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Agingtimer (see Note 1)		RAMsize (see Note 1)	Revision	0x00
Unkmultiports (see Note 1)		Unkuniports (see Note 1)		0x04
	SIO	Control (see Note 1)		0x08
Findnode23–Findnode16	Findnode31–Findnode24	Findnode39–Findnode32	Findnode47–Findnode40	0x0C
Findport		Findnode7–Findnode0	Findnode15–Findnode8	0x10
SECVIOctr	Findcontrol	Findnodeage		0x14
Unkmultictr		Unkunictr		0x18
		Numnodes		0x1C
MANtest	RAMaddr			0x20
		RAMdata		0x24
Intmask		Int		0x28
			Adddelcontrol	0x2C
Newnode23–Newnode16	Newnode31–Newnode24	Newnode39–Newnode32	Newnode47–Newnode40	0x30
Newport		Newnode7–Newnode0	Newnode15–Newnode8	0x34
Addnode23–Addnode16	Addnode31–Addnode24	Addnode39–Addnode32	Addnode47–Addnode40	0x38
Addport		Addnode7–Addnode0	Addnode15–Addnode8	0x3C
Agednode23–Agednode16	Agednode31–Agednode24	Agednode39–Agednode32	Agednode47–Agednode40	0x40
	Agedport	Agednode7–Agednode0	Agednode15–Agednode8	0x44
Delnode23–Delnode16	Delnode31–Delnode24	Delnode39–Delnode32	Delnode47–Delnode40	0x48
		Delnode7–Delnode0	Delnode15–Delnode8	0x4C
Manufacturer test (see Note 1)				0x50
Manufacturer test (see Note 1)				0x54
Manufacturer test (see Note 1)				0x58
Manufacturer test (see Note 1)				0x5C
Manufacturer test (see Note 1)				0x60
Manufacturer test (see Note 1)				0x64
Manufacturer test (see Note 1)				0x68
	Uplinkport	Manufacturer test (see Note 1)		0x6C

NOTE 1: These registers are automatically loaded from the attached EEPROM when the LOAD bit in the control register is set, or when the TNETX15AE is hardware reset by deasserting the $\overline{\text{RESET}}$ terminal and an external EEPROM is detected.

default register values at reset

The reset values for the default registers are shown in Table 3.

Table 3. Default Registers Reset Values

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
0x0000		0x80	0x10	0x00
0x7FFF		0x7FFF		0x04
0x00	MDIO*(0x10) + EDIO*(0x01)	0x0000		0x08
0x00	0x00	0x00	0x00	0x0C
0x0000		0x00	0x00	0x10
0x0000		0x0000		0x14
0x0000		0x0000		0x18
0x0000		0x0000		0x1C
0x00	0x000000			0x20
0x0000		SRAM data at address 0		0x24
0x0000		0x0000		0x28
0x0000		0x00	0x00	0x2C
0x00	0x00	0x00	0x00	0x30
0x0000		0x00	0x00	0x34
0x00	0x00	0x00	0x00	0x38
0x0000		0x00	0x00	0x3C
0x00	0x00	0x00	0x00	0x40
0x00	0x00	0x00	0x00	0x44
0x00	0x00	0x00	0x00	0x48
0x00	0x00	0x00	0x00	0x4C
0x7FFD		0x7FFE		0x50
0x7FF7		0x7FFB		0x54
0x7FDF		0x7FEF		0x58
0x7F7F		0x7FBF		0x5C
0x7DFF		0x7EFF		0x60
0x77FF		0x7BFF		0x64
0x5FFF		0x6FFF		0x68
0x00	0x00	0x3FFF		0x6C

EEPROM auto-configuration from an external x24C02 EEPROM

The flash EEPROM interface is provided so the system-level manufacturers can provide an optional preconfigured system to their customers. Customers can change or reconfigure their system and retain their preferences between system power downs.

The flash EEPROM contains configuration and initialization information, which is accessed infrequently (typically at power up and reset).

The TNETX15AE uses the 24C02 serial EEPROM device (2048 bits organized as 256×8). This device uses a two-wire serial interface for communications.

EEPROM programming can be done while in the system through the TNETX15AE EDIO port using suitable driver software.

EEPROM auto-configuration from an external x24C02 EEPROM (continued)

The organization of the EEPROM data is shown in Table 4. The last register loaded is the control register. This allows a complete initialization by downloading the contents of the EEPROM into the TNETX15AE. During the download, no DIO operations are permitted. The LOAD and RESET bits in the control register cannot be set during a download, preventing a download loop.

The TNETX15AE has reserved space in the register map with corresponding reserved space in the EEPROM. This is to support additional features in future revisions of the device with minimal impact on software drivers written for this device. Although the TNETX15AE does not use the values in the reserved EEPROM locations to load the reserved locations in the register map, the auto-loader reads every address from the beginning of the EEPROM to the CRC to calculate the CRC. That is, changing a loaded value or a reserved value requires a new CRC for the EEPROM.

The TNETX15AE detects the presence/absence of the EEPROM. If it is not installed, the EDIO terminal should be tied low. For EEPROM operation, the terminal requires an external pullup (see EEPROM data sheet). When no EEPROM is detected, the TNETX15AE assumes default register values at power up and is halted. Downloading a configuration from the EEPROM terminals is disabled when no EEPROM is present.

The first bit written to or read from the EEPROM is the most-significant bit of the byte, i.e., data (7). Therefore, writing the address 0xC0 is accomplished by writing a 1 and then 1, 0, 0, 0, 0, 0, 0. For details of reading, writing, or programming a 24C02 device, refer to the device data sheet.

The TNETX15AE causes data to be stored in the EEPROM in a specific format. The range from 0x00 to 0x2A in the EEPROM is reserved for use by the adapter. The contents of the remaining bytes are undefined. The EEPROM can be read/written by driver software through the SIO register.

A 32-bit CRC value must be calculated from the EEPROM data and placed in the EEPROM in the location following the bytes loaded into the internal registers. The TNETX15AE uses this 32-bit CRC to validate the EEPROM data. If the CRC fails, the TNETX15AE registers are set to their default (hardwired) values. This is the same state the TNETX15AE is in if no EEPROM is present. Without an EEPROM, a management CPU is required to load all the registers and set the start bit to 1 in the upper one-half of the control/status register (0x08). The CRC calculation on the EEPROM bytes is the same as the IEEE Std 802.3u for the packet CRC calculation on a byte-by-byte basis. For reference, the equation is:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (1)$$

The TNETX15AE EEPROM register assignments are shown in Table 4.

Table 4. TNETX15AE EEPROM Register Assignments

DIO REGISTER ADDRESS	EEPROM VALUE	EEPROM ADDRESS
0x01	RAMsize	0x00h
0x02	Agingtimer LSB	0x01h
0x03	Agingtimer MSB	0x02h
0x04	Unkuniports LSB	0x03h
0x05	Unkuniports MSB	0x04h
0x06	Unkmultiports LSB	0x05h
0x07	Unkmultiports MSB	0x06h
0x50	Manufacturing test (0xFE)	0x07h
0x51	Manufacturing test (0x7F)	0x08h
0x52	Manufacturing test (0xFD)	0x09h
0x53	Manufacturing test (0x7F)	0x0Ah
0x54	Manufacturing test (0xFB)	0x0Bh
0x55	Manufacturing test (0x7F)	0x0Ch
0x56	Manufacturing test (0xF7)	0x0Dh
0x57	Manufacturing test (0x7F)	0x0Eh
0x58	Manufacturing test (0xEF)	0x0Fh
0x59	Manufacturing test (0x7F)	0x10h
0x5A	Manufacturing test (0xDF)	0x11h
0x5B	Manufacturing test (0x7F)	0x12h
0x5C	Manufacturing test (0xBF)	0x13h
0x5D	Manufacturing test (0x7F)	0x14h
0x5E	Manufacturing test (0x7F)	0x15h
0x5F	Manufacturing test (0x7F)	0x16h
0x60	Manufacturing test (0xFF)	0x17h
0x61	Manufacturing test (0x7E)	0x18h
0x62	Manufacturing test (0xFF)	0x19h
0x63	Manufacturing test (0x7D)	0x1Ah
0x64	Manufacturing test (0xFF)	0x1Bh
0x65	Manufacturing test (0x7B)	0x1Ch
0x66	Manufacturing test (0xFF)	0x1Dh
0x67	Manufacturing test (0x77)	0x1Eh
0x68	Manufacturing test (0xFF)	0x1Fh
0x69	Manufacturing test (0x6F)	0x20h
0x6A	Manufacturing test (0xFF)	0x21h
0x6B	Manufacturing test (0x5F)	0x22h
0x6C	Manufacturing test (0xFF)	0x23h
0x6D	Manufacturing test (0x3F)	0x24h
0x08	Control LSB	0x25h
0x09	Control MSB	0x26h
	CRC byte 3 MSB	0x27h
	CRC byte 2	0x28h
	CRC byte 1	0x29h
	CRC byte 0 LSB	0x2Ah

internal register descriptions

revision register, revision at 0x00 (DIO)

BIT							
7 (MOST-SIGNIFICANT BIT)	6	5	4	3	2	1	0 (LEAST-SIGNIFICANT BIT)
Product code				Revision			
Initial values after reset							
0	0	0	1	0	0	0	0

This register contains the revision code for the device. The initial revision code is hardwired to 0x10. This register is read only, and writes to it are ignored.

SRAM size-select register, RAMsize at 0x01 (DIO)

The RAMsize register can be written to only when the START bit in the control register is set to 0. This register is automatically loaded from the EEPROM when the $\overline{\text{RESET}}$ terminal is asserted low, or when LOAD in the control register is set.

BIT							
7	6	5	4	3	2	1	0
Reserved				RSIZE			
Initial values after reset							
1	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
7–4	Reserved	Writes to this location are ignored and are read as 0.
3–0	RSIZE	Ram size select. This field indicates the size of the external SRAM, and has a bearing on the number of addresses that the TNETX15AE supports. The TNETX15AE uses this field to determine how many tables to initialize as part of the reset sequence by indicating how many address lines to use. The TNETX15AE uses external SRAM for tables of pointers used in address decoding and for table end pointers (the two-word address descriptors). This field can be written only when the START bit is 0 and reset is not active. This field also is loaded from the EEPROM interface during auto load. Code values are as follows: **RAMsize REGISTER** / **RAM SIZE** / **WORST CASE** 0x00 / 640 × 8 / 2 0x01 / 832 × 8 / 2 0x02 / 1K × 8 / 3 0x03 / 2K × 8 / 7 0x04 / 4K × 8 / 14 0x05 / 8K × 8 / 28 0x06 / 16K × 9 / 59 0x07 / 32K × 10 / 123 0x08 / 64K × 11 / 251 0x09 / 128K × 12 / 507 0x0A / 256K × 13 / 1019 0x0B / 512K × 14 / 2189 0x0C / 1M × 15 / 4530 0x0D to 0x0F / 2M × 16 / 9211

age-deletion time-select register, agingtimer at 0x02–0x03 (DIO)

BYTE 1								BYTE 0							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Agingtimer															
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The agingtimer register is 16 bits wide and is used to control the aging process. There are two aging modes, and the modes are selected according to the value of this register:

- When agingtimer is 0 or 0xFFFF, the TNETX15AE performs table-full aging. The TNETX15AE ages out the oldest address when the lookup table becomes full.
- When agingtimer is not 0 or 0xFFFF, the TNETX15AE performs threshold aging. The value in the agingtimer is the time threshold in seconds. All addresses that are older than this time are aged out.

Aging does not delete addresses that have been secured, and multicast addresses also are not aged. Aging is disabled when the NAUTO bit in the control register is set. It is management's responsibility in NAUTO mode to manage the lookup table.

All bits in this register are read/writeable and default to 0x0 during reset. This field also is automatically loaded from the EEPROM when the $\overline{\text{RESET}}$ terminal is asserted low or when LOAD in the control register is set to 1.

unknown unicast port-routing register, unkuniports at 0x04–0x05 (DIO)

BYTE 1								BYTE 0							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Unkuniports														
Initial values after reset															
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The unkuniports register is used to route unicast frames whose destination address is not found within the lookup table. Normally these frames are flooded to all ports except to the port that originated the frame. The TNETX15AE uses the unkuniports register to route these frames to only selected ports. When the TNETX15AE uses the unkuniports register for unicast flooding, it increments the unkunictr counter. The TNETX15AE masks out the originating port when using this register. This prevents the attached switch device from forwarding the frame to its originating port.

The bit numbers in this register have a one-to-one correspondence with the attached switch-device port number. This register is readable/writeable and is auto-loaded from the EEPROM when the $\overline{\text{RESET}}$ terminal is asserted low or when LOAD in the control register is set.

unknown multicast port-routing register, unkmultiports at 0x06–0x07 (DIO)

BYTE 1								BYTE 0							
BIT															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	Unkmultiports														
Initial values after reset															
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The unkmultiports register is used to route multicast frames whose multicast address is not found within the lookup table. Normally these frames are flooded to all ports except to the port that originated the frame. The TNETX15AE uses the unkmultiports register to route these frames to only selected ports. When the TNETX15AE uses the unkmultiports register for multicast flooding, it increments the unkmultictr counter. The TNETX15AE masks out the originating port when using this register. This prevents the attached switch from forwarding the frame to its originating port.

The bit numbers in this register have a one-to-one correspondence with the attached switch-device port number. This register is readable/writeable and is auto-loaded from the EEPROM when the $\overline{\text{RESET}}$ terminal is asserted low or when LOAD in the control register is set.

mode control status register, control at 0x08–0x09 (DIO)

BYTE 1								BYTE 0							
BIT															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESET	LOAD	START	INITD	NEEPM	NAUTO	Reserved			NLRN0	NCRC	Reserved				
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
No EEPROM detected															
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Auto loading fails															
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

The control register is automatically loaded from the EEPROM when the $\overline{\text{RESET}}$ terminal is asserted low, or when the LOAD bit is set. Only selected bits in this register are loaded from the EEPROM. RESET and LOAD bits are not loaded to prevent automatic loading loops. The two status bits, INITD and NEEPM, also are not loadable. If automatic loading fails due to EEPROM malfunction or CRC error, the control register RESET bit is set.

mode control status register, control at 0x08–0x09 (DIO) (continued)

BIT	NAME	FUNCTION
15	RESET	Reset. Writing a 1 to RESET places the TNETX15AE in a hardware reset state. This function sets all internal logic operations to a known state, and clears all registers (except for control). (All data from the lookup table is lost.) This bit is not automatically loaded from the EEPROM. If EEPROM automatic loading fails, then the RESET bit is set to 1.
14	LOAD	Load system. Writing a 1 to LOAD starts the automatic loading of registers from the attached EEPROM. This bit is not automatically loaded from the EEPROM. The EEPROM automatic loader clears this bit to 0 (writing a 0 to this bit has no effect). This is another way to initialize registers besides the auto-load (if EEPROM present) after reset.
13	START	Start system. Writing a 1 to START causes the TNETX15AE to begin operation. While the SRAM tables are initialized, no address checking is performed. (Writing a 0 to this bit has no effect.)
12	INITD	RAM-initialization-done signal. INITD becomes high when the lookup table SRAM is initialized. The TNETX15AE begins learning/matching addresses after this signal goes high. This is a read-only bit. Expect this to take 40 ns $\times$ RAMsize.
11	NEEPM	No external EEPROM. NEEPM indicates when an internal EEPROM is detected. If this bit is 1, no EEPROM is present or the TNETX15AE is unable to detect it, or the CRC is wrong. If this bit is set to 0, a properly formatted EEPROM was detected. This is a read-only bit.
10	NAUTO	Not automatically add-address mode select. NAUTO selects the manner in which addresses are added to the lookup table. In NAUTO mode, the aging logic operation is disabled. It is management's responsibility to manage the lookup table in this mode. – When set to 1, the TNETX15AE adds addresses only to the lookup table when a DIO ADD command is given to it. The bus watcher/LKUP state machines are still watching source addresses, and the NEW int is issued if a new address is observed on the wire. – When set to 0, the TNETX15AE automatically adds unknown addresses to its lookup table.
9–7	Reserved	Cleared to 0 on reset. Must always be written with a 0, even via EEPROM load.
6	NLRN0	Not-learn addresses from port 0. When set, the TNETX15AE does not learn SRC addresses on port 0 (uplink).
5	NCRC	No CRC check. NCRC enables/disables the add-on-only-good CRC function. – When set to a 1, the TNETX15AE adds frames immediately after the source address is found on the DRAM bus. A good CRC is not required to add the source address to the table. – When set to a 0, the TNETX15AE waits until the EOB/EOF and a good CRC indication before adding addresses. Only one source address add can be pending at a time. Waiting on a good CRC can force the TNETX15AE to miss adds on other ports.
4–0	Reserved	Writes to these locations are ignored and read as 0.

serial interface I/O register, SIO at 0x0A (DIO)

BIT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NMRST	MCLK	MTXEN	MDATA	MDIOEN	ECLOK	ETXEN	EDATA
Initial values after reset							
0	0	0	MDIO	0	0	0	EDIO

The SIO register contains the control bits for two serial interfaces to the TNETX15AE. Setting the outbound signals changes the state of the external terminal. Sampling the inbound signals gives the state of the external terminal. The MII to the PHY devices has a serial management portion. This interface can have multiple masters, and allows for two-way data between master and slave. The interface to the EEPROM is simpler because it is a dedicated path between one TNETX15AE and its EEPROM. Although it is also serial and two way, there is only one master.

BIT	NAME	FUNCTION
7	NMRST	MII not reset. The state of MNRST directly controls the state of the $\overline{\text{MRESET}}$ line (MII reset) – If NMRST is set to 0, the $\overline{\text{MRESET}}$ line is set to 0. – If NMRST is set to 1, the $\overline{\text{MRESET}}$ line is set to 1. This bit is not self-clearing and must be manually deasserted. It can be set low and then immediately set high. Since every PHY attached to the MII may not have a reset terminal, it may be necessary to toggle NMRST and also individually reset each PHY through MII commands. The default state of this bit is 0 (MII is in reset). The MDIOEN bit must be set to drive $\overline{\text{MRESET}}$.
6	MCLK	MII SIO clock. MCLK controls the state of the MDCLK terminal. The MDIOEN bit must be set to drive MDCLK. – When MCLK is set to 1, MDCLK is set to 1. – When MCLK is set to 0, MDCLK is set to 0.
5	MTXEN	MII SIO transmit enable. MTXEN is used with the MDATA bit to read/write information from/to the MDIO terminal. The MDIOEN bit must be set to drive MDIO. – When MTXEN is set to 1, the MDIO terminal is driven with the value in the MDATA bit. – When MTXEN is set to 0, MDATA is loaded with the value in the MDIO terminal.
4	MDATA	MII SIO data. MDATA is used with MTXEN to read/write information from/to the MDIO terminal. – When MTXEN is set to 1, MDIO is driven with the value in this bit. – When MTXEN is set to 0, this bit is loaded with the value on MDIO. The MDIOEN bit must be set to drive MDIO. The default value of this bit is the value driven on the MDIO terminal.
3	MDIOEN	MII SIO data terminal enable. MDIOEN enables the high-Z control of the MDIO, MDCLK, and $\overline{\text{MRESET}}$ terminals. – Setting MDIOEN to 1 enables MII outputs. – Setting MDIOEN to 0 places MII outputs in a high-Z state. The default state of this bit is 0 (MII disabled)
2	ECLOK	EEPROM SIO clock. ECLOK controls the state of the ECLK terminal. – When ECLOK is set to 1, ECLK is set to 1. – When ECLOK is set to 0, ECLK is set to 0.
1	ETXEN	EEPROM SIO transmit enable. ETXEN controls the direction of the EDIO terminal. – When ETXEN is set to 1, EDIO is driven with the value in the EDATA bit. – When ETXEN is set to 0, EDATA is loaded with the value on the EDIO terminal.
0	EDATA	EEPROM SIO data. EDATA is used to read or write the state of the EDIO terminal. – When EXTEN is set to 1, EDIO is driven with the value in this bit. – When EXTEN is set to 0, EDATA is loaded with the value on EDIO. The default value of this bit is the value driven on the EDIO terminal.

management-table lookup interface at 0x0B–0x16 (DIO)

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Findnode23–Findnode16	Findnode31–Findnode24	Findnode39–Findnode32	Findnode47–Findnode40	0x0C
Findport		Findnode7–Findnode0	Findnode15–Findnode8	0x10
	Findcontrol	Findnodeage		0x14

The management-table lookup registers allow the management entity to find information about the node addresses contained in the table.

These registers are part of the address lookup/inspection mechanism supported through the DIO (host) interface. An address is fully described by the six bytes of findnode (0xC to 0x11). Data about an address is returned in findport (0x12). FIND commands are issued through the findcontrol (0x16) register. They are discussed later in more detail in numerical order.

To cause a FIND operation to execute, the host writes to the findcontrol register (0x16). To find the first address in the table, execute a FIRST command. The target MAC address is returned in six bytes of findnode registers. If a successive FIND command is executed (findnext), the starting point to continue the search is the six bytes of findnode. The results are returned in the same registers, overwriting the original values. These registers can be read/write between FINDS; a find next picks up with the current values of findnode. If a lookup on an address to check parameters is executed, the data returned corresponds to the value in the six bytes of findnode when the lookup command was given; the address parameters are posted in the findport register (0x12, 0x13). The FIND operation completion is detected when the command bit is written back to 0 (poll). The find register block is discussed later in further detail in numerical order.

lookup-node address register, findnode at 0x0C–0x11 (DIO)

The findnode registers are used to exchange addresses between the TNETX15AE and the management CPU. The node address in findnode is kept in Ethernet native data format; bit 40 is the group/specific bit. The function of findnode depends on the bits set in findcontrol:

- On FIRST operations, this register shows the first address in the lookup table. The data in findnode is valid only when the FOUND bit in findcontrol is a 1.
- On NEXT operations, this register shows the next address in the lookup table. The data in findnode is valid only when the FOUND bit in findcontrol is a 1.
- On LKUP operations, the lookup logic operation looks up the address stored in this register. If found, the FOUND bit in findcontrol is set to a 1.

This register is readable/writeable. While any of the FIND command bits are set, writes to these addresses are ignored.

lookup routing-code register, findport at 0x12–0x13 (DIO)

The findport register is a read-only register, which returns port assignment information for the node address contained in the findnode register after a lookup. The data structure for the findport register depends on the type of address stored in the findnode register.

If findnode contains a unicast address, findnode (40) = 0 (bit 40 of the Ethernet adddress is 0), then findport is defined as:

BYTE 3								BYTE 2							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALID	SECURE	LOCKED	CUPLNK	PORTCODE				Reserved							
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
15	VALID	Valid address indication.
14	SECURE	Secured address indication. SECURE shows the security level for the address contained in the findnode register. Secure addresses are not aged out and cannot move ports. If a station with a secured address moves ports, a security violation interrupt is given to the host, and the address is locked. Since the lookup of the destination address causes the queueing of the packet for transmit, and this happens before the source address is inspected, the packet that causes a security violation is processed through the switch.
13	LOCKED	Locked address indication. LOCKED shows the lock status for the address contained in the findnode register. Locked addresses output a discard code on the EAM interface; frames destined for LOCKED addresses are not delivered. Since the destination address lookup, which causes the queueing of the packet for transmission, is independent of checking the source address, locked addresses cannot receive packets (but still can transmit) into the switch attached to TNETX15AE.
12	CUPLNK	Copy frames to uplink indication. CUPLINK shows the copy uplink status for the address contained in the findnode register. The address tagged by this bit adds the port specified in the uplinkports register to the routing code output on the EAM bus. This can affect performance of the TNETX3150/TNETX3150A/TNETX3100, as each packet must be linked to two transmit queues.
11–8	PORTCODE	Current port for node. PORTCODE holds the current port for the unicast address shown in the findnode register.
7–0	Reserved	Writes to this location are ignored and read as 0.

If findnode is a multicast address, findnode (40) = 1 (bit 40 of the Ethernet address is 1), then findport is defined as:

BYTE 3								BYTE 2							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALID	PORTFLAG														
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
15	VALID	Valid address indication.
14–0	PORTFLAG	Port-bit vector for multicast. PORTFLAG shows the port-bit vector for the multicast address contained in the findnode register. The bit values in this field correspond one-to-one with the TNETX3150/TNETX3150A/TNETX3100 port assignment.

node age-stamp register, findnodeage at 0x14–0x15 (DIO)

BYTE 1								BYTE 0							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NODEAGE															
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The findnodeage register is a read-only register, which holds the current 16-bit age-time stamp of the address contained in the findnode register after a find command executes. Age units are seconds. The time stamp indicates when this address was last seen.

lookup table search control register, findcontrol at 0x16 (DIO)

The management logic uses the findcontrol register to scan the lookup table for addresses. Three commands are available: findfirst (FIRST), findnext (NEXT), find (LKUP). Only one command is valid at one time. Example: a findfirst and a findnext command cannot be issued at the same time (0x06). The TNETX15AE ignores all multiple commands. This register is readable/writeable. Further accesses of this register are delayed until the operation is complete; that is, SRDY does not go active until the FIND state machine is done. This can produce a 6.4-µs read or write cycle if this register is accessed right after a FINDNEXT is started on a sparsely loaded table.

BIT							
7	6	5	4	3	2	1	0
FOUND	Reserved				FIRST	NEXT	LKUP
Initial values after reset							
0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
7	FOUND	Address found. If the address contained in the findnode register is found in the table, FOUND is asserted. The data contained in the find interface (findnode, findport, and findnodeage) is valid only when this bit is a 1. This is a read-only bit.
6–3	Reserved	Reserved
2	FIRST	Lookup first address. When FIRST is 1, the TNETX15AE scans the address table for the first valid address. If FOUND is set, the address parts are placed in findnode, findport, and findnodeage.
1	NEXT	Lookup next addresses. When asserted, the TNETX15AE scans the address table for the next available address after the address currently in the findnode registers (0xC–0x11). If FOUND is set, the address parts are placed in findnode, findport, and findnodeage.
0	LKUP	Address lookup. When asserted, the TNETX15AE scans the address table for the address contained in the findnode register. If found, the FOUND bit reads a 1 and findport and findnodeage are valid; if not found, it reads a 0.

statistics registers at 0x17–0x1D (DIO)

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Secvioctr				0x14
Unkmultictr		Unkunictr		0x18
		Numnodes		0x1C

All registers in this field are read only and their default value after reset is 0. The 16-bit counters in this section must be read by bytes. It is recommended that the bytes be read least-significant byte through most-significant byte (last). An additional stimulus can cause a carry between these two reads.

security violation counter, secvioctr at 0x17 (DIO)

The secvioctr field contains the number of times that a secured address attempts to move ports. This register generates a stat interrupt (statistics overflow interrupt) when it is one-half full (most-significant bit in the field is a 1). Reading this register automatically clears it and the default value of this register is 0x00.

unknown unicasts counter, unkunictr at 0x18–0x19 (DIO)

The unkunictr register counts the number of times that the TNETX15AE device floods a frame that has a unicast destination address. These frames are used when the TNETX15AE cannot find the destination address in its lookup table. This register generates a stat interrupt (statistic overflow interrupt) when it is one-half full (most-significant bit in the field is a 1). Reading the most-significant byte of this register automatically clears it and the default value of this register is 0x0000.

unknown multicasts counter, unkmultictr at 0x1A–0x1B (DIO)

The unkmultictr register counts the number of times that the TNETX15AE device floods a frame that has a multicast destination address. Multicast destination addresses are flooded when the TNETX15AE cannot find the destination address in its lookup table. This register generates a stat interrupt (statistics overflow interrupt) when it is one-half full (most-significant bit in the field is a 1). Reading the most-significant byte of this register automatically clears it and the default value of this register is 0x0000.

lookup table node count, numnodes at 0x1C–0x1D (DIO)

The numnodes counter register contains the number of addresses currently in the lookup table. This register is read only and its value at reset is 0x0000.

SRAM address register, RAMaddr at 0x20–0x22 (DIO)

The RAMaddr register is used with the RAMdata (at 0x24, 0x25) register to access the TNETX15AE external SRAM.

BYTE 2								BYTE 1								BYTE 0							
BIT																							
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INC	Reserved		RAM ADD																				
Initial values after reset																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
23	INC	Address auto-increment. When INC is 1, this increments the RAM ADD field to access the next location in the SRAM. The address is incremented by one after every read or write is performed on the most-significant byte of the RAMdata register.
22–21	Reserved	Writes to this location are ignored and read as 0.
20–0	RAM ADD	RAM address. This 21-bit field holds the address of the SRAM location, which is read or written. The data that is to be read or written is placed in the RAMdata register. This interface is read/write at all times except during reset.

manufacturing test register, MANtest at 0x23 (DIO)

This register is reserved for Texas Instruments (TI™) manufacturing test. It is written to 0x0 for normal operation. All reset mechanisms leave 0x0 in this register, and it is not loaded from the EEPROM. It is written to only when the START bit is 0.

It contains mechanisms, for example, that increment all the counters at the same time (or run the aging algorithm quicker than usual to decrease the time required to test the part at manufacture). Setting any of these bits results in abnormal operation. To allow improvement to the testing, these functions are subject to change.

SRAM data register, RAMdata at 0x24–0x25 (DIO)

BYTE 1								BYTE 0							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAMdata after reset															
Data at SRAM (0x000000)															

The RAMdata register is used to access the SRAM address held in the RAM ADD field of the RAMaddr register. This field is 16-bits wide.

- Writes to the RAM are executed when the MS byte of the RAMdata register is written to by the host. When depositing values for transfer to SRAM, write the least-significant byte first.
- Reads are accomplished by reading the data from either byte in the RAMdata register.
- The SRAM address to be accessed should be placed in the RAMaddr register. If the INC bit in the RAMaddr register is a 1, the address to be accessed is increased when the most-significant byte of RAMdata is accessed.

interrupt register, int at 0x28–0x29 (DIO)

The int register is used with the intmask register to provide interrupts to the attached CPU. When the TNETX15AE EINT terminal is 1, this register gives the reason for the interrupt. Specific interrupts can be masked out by setting the appropriate bit in the intmask register. This register is read only, with the exception of the int bit. All bits in a byte are automatically cleared when the byte is read, even if more then one was set.

BYTE 1								BYTE 0							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEW	NEWM	CHNG	CHNGM	SECVIO	SECVIOM	AGE	AGEM	INT	Reserved					STAT	FULL
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
15	NEW	New-node interrupt. NEW indicates that a new node has been added to the lookup table. The node address is given in the newnode register and the node's port is given in the newport register.
14	NEWM	Missed-new-node interrupt indication. NEWM indicates that a new node interrupt was given, but the information previously placed in the newnode registers has not been read by the host CPU.
13	CHNG	Node-port change interrupt. CHNG indicates that there has been a change in port assignment for a node that exists in the lookup table. The node address is given in the newnode register and the node's new port is given in the newport register.
12	CHNGM	Missed-node-port change interrupt indication. CHNGM indicates that a node-port change interrupt was issued, but the information previously placed in the newnode registers has not been read by the host CPU.
11	SECVIO	Security-violation interrupt. SECVIO indicates that a node (which has been secured) has attempted to move port assignments. The node address is given in the newnode registers. The location where the node attempted to move is contained in the newport register.
10	SECVIOM	Missed-security-violation interrupt indication. SECVIOM indicates that a node-port change interrupt was issued, but the information previously placed in the newnode registers has not been read by the host CPU.
9	AGE	Age-out interrupt. AGE indicates that a node has been aged-out (deleted from the lookup table). The node address is issued in the agednode register. The node's assigned port is given in the agedport register.
8	AGEM	Missed-age-out interrupt indication. AGEM indicates that an age-out interrupt was issued, but the information previously placed in the agednode registers has not been read by the host CPU.
7	INT	Test interrupt request. Asserting INT gives a test interrupt to the attached CPU.
6–2	Reserved	Writes to this location are ignored and read as 0.
1	STAT	Statistics overflow interrupt. STAT indicates that a counter in the statistics is one-half full (most-significant bit in the counter is a 1). This is an indication to the CPU to read the statistic counters (thereby clearing them).
0	FULL	SRAM full interrupt. FULL indicates that a condition existed (or exists) that will prevent the addition of another address to the TNETX15AE address-lookup table. During the next ADD, space for either a pointer table (32 word) or a station descriptor (four bytes) will not be available. If NAUTO (bit 10, control 0x9) is set, the host is responsible for adding addresses. The host-initiated ADD operation did not complete, and the FULL interrupt was asserted. The host must delete addresses until the ADD completes. If NAUTO is not set, the internal state machines have deleted the necessary oldest addresses until the ADD operation completes, even if the ADD was initiated by the host. In this case, the interrupt indicates that the table is operating near peak capacity.

interrupt masking register, intmask at 0x2A–0x2B (DIO)

The intmask register is used with the int register to select the type of interrupts that should be enabled to the attached CPU. Bit definitions in the intmask register agree one-to-one with the bit definitions in the int register. Only those fields with the bit set to 1 generate an interrupt to the CPU. Clearing the bit for a condition prevents only the external terminal from moving to a logic 1 when the appropriate condition is true. The appropriate bit in the interrupt register is always a 1 if the condition is true, and the bit has not been cleared by being read. This register is read/writeable.

BYTE 1								BYTE 0							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEW	NEWM	CHNG	CHNGM	SECVIO	SECVIOM	AGE	AGEM	INT	Reserved					STAT	FULL
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
15	NEW	New-node interrupt mask. When NEW is set, a new-node interrupt is posted if the NEW bit in the int register is set.
14	NEWM	Missed-new-node interrupt mask. When NEWM is set, a missed new-node interrupt is posted if the NEWM bit in the int register is set.
13	CHNG	Node-port change interrupt mask. When CHNG is set, a node-port change interrupt is posted if the CHNG bit in the int register is set.
12	CHNGM	Missed-node-port change interrupt mask. When CHNGM is set, a missed-node-port interrupt is posted if the CHNGM bit in the int register is set.
11	SECVIO	Security-violation interrupt mask. When SECVIO is set, a security-violation interrupt is posted if the SECVIO bit in the int register is set.
10	SECVIOM	Missed-security-violation interrupt mask. When SECVIOM is set, a missed-security-violation interrupt is posted if the SECVIOM bit in the int register is set.
9	AGE	Age-out interrupt mask. When AGE is set, an age-out interrupt is posted if the AGE bit in the int register is set.
8	AGEM	Missed-age-out interrupt mask. When AGEM is set, a missed age-out interrupt is posted if the AGEM bit in the int register is set.
7	INT	Test interrupt mask. When INT is set, a test interrupt is posted if the INT bit in the int register is set.
6–2	Reserved	Writes to this location are ignored and read as 0.
1	STAT	Statistics overflow interrupt mask. When STAT is set, a statistics interrupt is posted if the STATS bit in the int register is set.
0	FULL	SRAM full interrupt mask. When FULL is set, a memory-full interrupt is posted if the FULL bit in the int register is set.

new-add aged-del register set

The add/delete register set is used by the state machines (logic operations) to report on stations' addresses that have been added, deleted, or involved in port security violations as executed by the internal state machines, or to request such actions be taken on the specified address on behalf of the host.

The newnode and agednode set of registers send status information from the state machines to the host following an interrupt. Addnode and delnode registers are host information/actions to be applied to the TNETX15AE table. The status groups (new, aged) are covered in detail first, followed by the host command groups (add, delete).

REGISTER SET AND ADDRESSES				
BYTE 3	**BYTE 2**	**BYTE 1**	**BYTE 0**	**DIO ADDRESS**
			Adddelcontrol	0x2C
Newnode23–Newnode16	Newnode31–Newnode24	Newnode39–Newnode32	Newnode47–Newnode40	0x30
Newport		Newnode7–Newnode0	Newnode15–Newnode8	0x34
Addnode23–Addnode16	Addnode31–Addnode24	Addnode39–Addnode32	Addnode47–Addnode40	0x38
Addport		Addnode7–Addnode0	Addnode15–Addnode8	0x3C
Agednode23–Agednode16	Agednode31–Agednode24	Agednode39–Agednode32	Agednode47–Agednode40	0x40
	Agedport	Agednode7–Agednode0	Agednode15–Agednode8	0x44
Delnode23–Delnode16	Delnode31–Delnode24	Delnode39–Delnode32	Delnode47–Delnode40	0x48
		Delnode7–Delnode0	Delnode15–Delnode8	0x4C

new-node/port-change/security-violation interrupt interface at 0x30–0x37 (DIO)

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Newnode23–Newnode16	Newnode31–Newnode24	Newnode39–Newnode32	Newnode47–Newnode40	0x30
Newport		Newnode7–Newnode0	Newnode15–Newnode8	0x34

The new-node/port-change/security-violation interrupt interface is used with the int and intmask registers to exchange information relating to new addresses being added or modified in the lookup table following a wire event. This means a new source address was seen, a source address was seen on a new port, or a source address was seen on a port other than the one where it was secured, regardless of the state of the NAUTO bit in the control register. These registers are valid on a NEW, CHNG, or SECVIO interrupt. These registers are read only and default to 0 on reset. With NAUTO = 0, a new address is already added to the table. With NAUTO = 1, the host CPU gets notification of a new address via these registers and must perform the add.

To avoid the data in these registers from being changed if another interrupt occurs while reading the current data, the data in these registers is locked until the most-significant byte of the newport register is read.

The bit definition follows:

new-address register, newnode at 0x30–0x35 (DIO)

The newnode registers contain the node address (in Ethernet native data format) for which the interrupt was given. The default value of this register after reset is 0x00.00.00.00.00.00.

new-address port register, newport at 0x36–0x37 (DIO)

BYTE 3								BYTE 2							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALID	Reserved			PORTCODE				Reserved				OLDPORT			
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
15	VALID	Valid address indication.
14–12	Reserved	Writes to this location are ignored and read as 0.
11–8	PORTCODE	Current port for node. The PORTCODE field holds the assigned port number for the address contained in the newnode register.
7–4	Reserved	Writes to this location are ignored and read as 0.
3–0	OLDPORT	Old port for address. When an address moves port locations, OLDPORT contains the old port location for the address. When a security-violation interrupt is asserted by the TNETX15AE, the SECVIO bit is set in the int register, and OLDPORT shows the port where the node attempted to move.

agednode interrupt interface group at 0x40–0x46 (DIO)

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Agednode23–Agednode16	Agednode31–Agednode24	Agednode39–Agednode32	Agednode47–Agednode40	0x40
	Agedport	Agednode7–Agednode0	Agednode15–Agednode8	0x44

The agednode interrupt interface is used with the int and intmask registers to pass information to the management agent about addresses that have been deleted from the lookup table due to the internal aging process. The information placed in these registers (written in Ethernet native data format) is valid only when the AGE bit in the int register is set to a 1. These registers are read only and default to 0 after reset.

To prevent the data in these registers from being changed if another interrupt occurs while reading the current data, the data in these registers is locked until the agedport register is read.

aged-address register, agednode at 0x40–0x45 (DIO)

On an age interrupt, the agednode registers contain the address of the node that has been aged out from the lookup table. This is a read-only register and defaults to 0x00.00.00.00.00.00 after reset. A different interrupt is issued if this information remains unread long enough that another node is aged out before the agednode registers are released.

aged-address port assignment, agedport at 0x46 (DIO), byte 2

BIT							
7	6	5	4	3	2	1	0
Reserved				PORTCODE			
Initial values after reset							
0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
7–4	Reserved	Writes to this location are ignored and read as 0.
3–0	PORTCODE	Agednode port assignment. PORTCODE displays the assigned port for the aged-out address contained in the agednode register.

management add/edit address interface group at 0x2C and 0x38–0x3F (DIO)

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
			Adddelcontrol	0x2C
				0x30
				0x34
Addnode23–Addnode16	Addnode31–Addnode24	Addnode39–Addnode32	Addnode47–Addnode40	0x38
Addport		Addnode7–Addnode0	Addnode15–Addnode8	0x3C

The management add/edit address registers are used with the ADD bit in the adddelcontrol register to perform CPU adds and edits to the lookup table. These registers are normally readable/writeable. As long as the ADD bit in the addelcontrol register is high, writes to these bytes are ignored.

lookup table add/delete command register, adddelcontrol at 0x2C (DIO), byte 0

BYTE 0							
BIT							
7	6	5	4	3	2	1	0
Reserved						ADD	DEL
Initial values after reset							
0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
7–2	Reserved	Writes to this location are ignored and are read as 0.
1	ADD	Address add. When ADD is asserted, the TNETX15AE uses the information contained in the management add/edit address interface to add or edit an address in the lookup table. ADD remains asserted until the add process is complete.
0	DEL	See the delete command bit description and the description of the delnode registers (0x48–0x4f).

The adddelcontrol register can be read at any time. If the ADD or DEL bit is being written and that bit is already set, the DIO interface will stall and SRDY will be delayed until the current cycle completes. If neither of these two bits is already set during a write, a normal cycle results.

add address register, addnode at 0x38–0x3D (DIO)

The unicast or multicast address in this register (written in Ethernet native data format) is added to the lookup table when the ADD bit in the adddelcontrol register is set to 1, along with the data in the addport register. The default value of this register after reset is 0x00.00.00.00.00.00.

add routing code register, addport at 0x3E–0x3F (DIO)

The addport register is used to set or change port and flag assignments for the node address contained in the addnode register. The definition for the addport register depends on whether the address stored in the addnode register is a unicast or multicast address.

This interface is used by the host to enter broadcast, multicast, and unicast addresses to the TNETX15AE table. Wire ADDs operate on packet-source addresses, and all of these are forced to unicast addresses.

If addnode is a unicast address, (addnode (40) = 0), addport is defined as follows:

BYTE 3								BYTE 2							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SECURE	LOCKED	CUPLNK	PORTCODE				Reserved							
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
15	Reserved	Writes are ignored and read as 0.
14	SECURE	Secured address flag. SECURE is used to change the security level for the address contained in the addnode register.
13	LOCKED	Locked address flag. LOCKED locks/unlocks the address contained in the addnode register on an add operation. Locked addresses output a discard code on the EAM interface.
12	CUPLINK	Copy frames to uplink flag. CUPLINK sets the copy uplink status for the address contained in the addnode register. Addresses tagged by this bit add the port (specified in the uplinkports register) to the routing code.
11–8	PORTCODE	Current port code for address. The value in PORTCODE becomes the port to which packets (sent to this address) are forwarded. In addresses that are added automatically to the table, the source port goes to this field. This field is writeable, and can be used to redirect traffic.
7–0	Reserved	Writes to this location are ignored and read as 0.

If addnode is a multicast address, (addnode (40) = 1), addport is defined as follows:

BYTE 3								BYTE 2							
BIT															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PORTFLAG														
Initial values after reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
15	Reserved	Writes are ignored and read as 0.
14–0	PORTFLAG	PORTFLAG for multicast. PORTFLAG is the port assignment for the multicast address contained in the addnode register. The bit values in this field correspond one-to-one with the TNETX3150/TNETX3150A/TNETX3100 port assignment.

management delete address interface group at 0x2C, 0x48–0x4F (DIO)

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
			Adddelcontrol	0x2C
				0x30
				0x34
				0x38
				0x3C
				0x40
				0x44
Delnode23–Delnode16	Delnode31–Delnode24	Delnode39–Delnode32	Delnode47–Delnode40	0x48
		Delnode7–Delnode0	Delnode15–Delnode8	0x4C

The management delete address interface is used with the DEL bits in the adddelcontrol register to perform management deletions from the lookup table.

lookup table add/delete command register, adddelcontrol at 0x2C (DIO), byte 0

BYTE 0							
BIT							
7	6	5	4	3	2	1	0
Reserved						ADD	DEL
Initial values after reset							
0	0	0	0	0	0	0	0

BIT	NAME	FUNCTION
7–2	Reserved	Writes to this location are ignored and are read as 0.
1	ADD	See the add command-bit description and the description of the addnode registers (0x38–0x3f).
0	DEL	Address delete. When DEL is asserted, the TNETX15AE uses the information contained in the management delete address interface to delete an address from the lookup table. DEL remains asserted until the delete process is complete.

The adddelcontrol register can be read at any time. If the ADD or DEL bit is being written and that bit is already set, the DIO interface will stall and SRDY will be delayed until the current cycle completes. If neither of these two bits is already set during a write, a normal cycle results.

delete-node address register, delnode at 0x48–0x4D (DIO)

The delnode registers are used with the DEL bit in the adddelcontrol register to allow for management deletion of an address in the lookup table. To delete an address, the address to be deleted is placed in Ethernet native data format in this register, and the DEL bit in the adddelcontrol register is asserted. This register defaults to 0x00.00.00.00.00.00 after reset. These registers are normally read/write except when the DEL bit in the adddelcontrol register is high; then, writes are ignored.

uplink routing register, uplinkport at 0x6E (DIO)

BIT							
7	6	5	4	3	2	1	0
Reserved				Uplink			
Initial values after reset							
0	0	0	0	0	0	0	0

The uplinkport register routes selected packets to a user-selected port. This register is valid only when the destination address being looked up has the CUPLNK bit set. Then the TNETX15AE forwards frames to the port specified in the lookup table and the port specified in this register. The TNETX15AE masks (does not send frames to) the port that originated the frame.

PRINCIPLES OF OPERATION

The purpose of this section is to assist in the development of management operational code for the TNETX15AE. Management code is drastically simplified when using the TNETX15AE since it is designed to operate without the need for a management CPU. Many of the functions that were performed previously by a CPU are now integrated on the TNETX15AE.

This section discusses the following topics:

- A background on the TNETX15AE architecture, including the data-storage algorithm and the arbitration between logic operations
- How to access the TNETX15AE registers through the DIO interface, and how these registers can be used to access the internal/external SRAM, the EEPROM, and any MII-managed devices
- How to access the TNETX15AE through the DIO interface from a hardware reset, and do the steps needed to bring the TNETX15AE to an operational state
- How to use the logic operations to perform management-based lookups, adds, and deletions.
- How to use the TNETX15AE interrupt support to create event-driven management code.

internal operation

This section describes the TNETX15AE hardware and how it affects the programmer and the TNETX3150/TNETX3150A/TNETX3100 performance. The functionality described in this section is transparent to the user, but it is included to help the user become familiar with the TNETX15AE architecture and how it interfaces to the TNETX3150/TNETX3150A/TNETX3100 device. Refer to the latest TNETX3150/TNETX3150A/TNETX3100 documentation for additional information.

EAM codings and in-order broadcasts (IOB)

The TNETX15AE uses two styles of EAM codings – single-port and multiple-port codings. The TNETX3150/TNETX3150A/TNETX3100 treats these two types of coding differently.

Single-port codings forward frames to a single port. The TNETX3150/TNETX3150A/TNETX3100 queues these frames to the port queue.

Multiple-port codings forward frames to multiple ports. The TNETX3150/TNETX3150A/TNETX3100 creates an IOB list structure to queue this frame to multiple-port queues. IOB lists use more memory bandwidth than a regular list because IOB lists require the use of an extra 64-byte buffer to contain all queue pointers for the ports. If many small frames are sent to multiple ports, IOB structures then can use significant TNETX3150/TNETX3150A/TNETX3100 memory bandwidth.

PRINCIPLES OF OPERATION

EAM codings and in-order broadcasts (IOB) (continued)

The TNETX15AE uses single-port codings when possible, to maximize performance. The TNETX15AE inspects all multiple-port codings. If, after removing all ineligible receivers (e.g., source port) only one destination port is left, the TNETX15AE switches to a single-port coding on the EAM bus.

For a more complete description of IOB lists, refer to the latest TNETX3150/TNETX3150A/TNETX3100 specification.

TNETX15AE DRAM and EAM interface

The TNETX15AE takes its frame input through the TNETX3150/TNETX3150A DRAM bus. It must recognize a start-of-frame (SOF) indication on the first flag byte of the frame. After the SOF is found, the TNETX15AE latches the first 32 bits of the destination address on the next DRAM cycle. At this point, it must complete a lookup cycle, choose the appropriate EAM code, and output this code in 440 ns, or less. Figure 3 shows the lookup timing. Check the data set for the TNETX3150/TNETX3150A/TNETX3100 for field definitions beyond those used by the TNETX15AE. The data on each cycle has the following format.

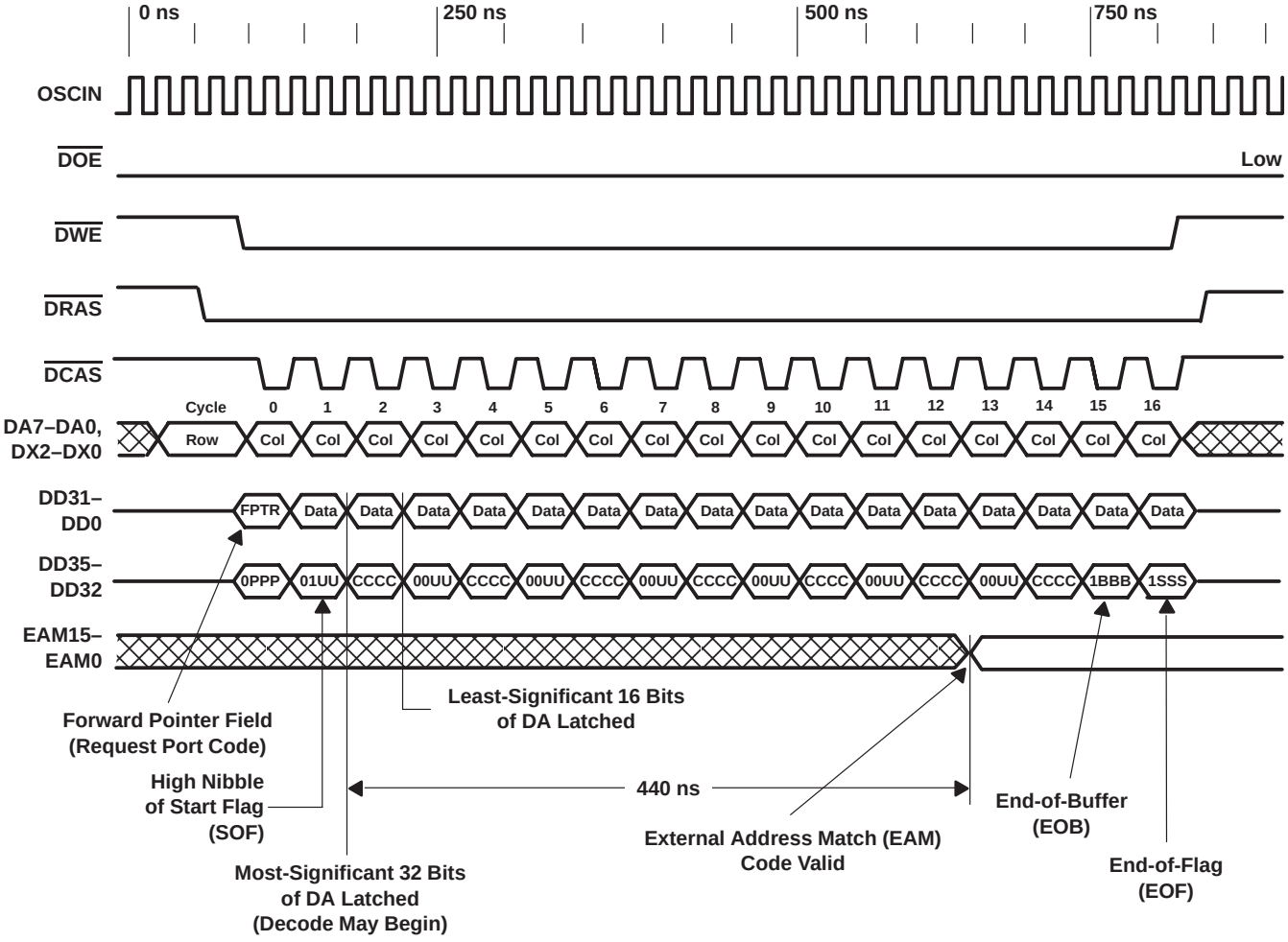


Figure 3. Lookup Cycle Timing

PRINCIPLES OF OPERATION

TNETX15AE DRAM and EAM interface (continued)

The TNETX15AE must determine that the data being written to memory is the first buffer of a data frame and not an IOB index buffer. The buffer contains packet data if the IOB bit is 0. The channel/source port code is latched from bits DD27–DD24 during the forward pointer (FPTR) transfer when TNETX3150/TNETX3150A/TNETX3100 is writing packet data to DRAM.

The TNETX15AE must determine the start of frame by looking at the flag in cycle 1. The flag is given in the DD35–DD32 terminals. The SOF is shown in the following table in cycle 1 at bit (35–34) = 01b.

CYCLE	BIT															
	35	34	33	32	31	30	29	28	27	26	25	24	23 – 16		15 – 0	
0	IOB	Parity†			Tag†				Channel code				Forward pointer†			
1	0	1	Reserved										Most-significant 32 bits of DA			
2	Channel code†												Most-significant 16 bits SA		Least-significant 16 bits DA	
3	0	0	Reserved										Least-significant 32 bits of SA			
• • •	Flags†												Data†			
N–1	EOB	Valid bytes†											Data†			
N	EOF	Frame status†											CRC			

† These bits are ignored by the TNETX15AE.

The TNETX15AE destination lookup cycle begins when it latches the partial destination address to begin the table lookup in cycle 1, and ends when it outputs an EAM code within the allocated 440 ns after the SOF condition is met. It must pick up the 16 bits at the destination in cycle 2 to complete the decode.

The channel-code field in the body of the frame is exactly the same as that in the forward pointer. The TNETX15AE uses the channel code from the forward pointer.

decoding the destination address (DA) and source address (SA) from the DRAM bus

The TNETX3150/TNETX3150A/TNETX3100 writes data to the DRAM buffers in a specific format (refer to the TNETX3150/TNETX3150A/TNETX3100 data sheet for additional information on this format). The TNETX15AE recognizes this format to determine the correct destination and source addresses. The format for the DA and the SA is shown in the following:

CYCLE	BIT			
	31 – 25	24 – 16	15 – 8	7 – 0
1	DA23–DA16	DA31–DA24	DA39–32	DA47–DA40
2	SA39–SA32	SA47–SA40	DA7–DA0	DA15–DA8
3	SA7–SA0	SA15–SA8	SA23–SA16	SA31–SA24

The TNETX15AE will latch and save the source address, cycles 2 and 3, if it is not already saving one. A source-address lookup will take place when there is no activity (TNETX3150/TNETX3150A/TNETX3100 to DRAM), when the transfer to memory is not the first 64 bytes of a packet, or when the transfer is from memory to an output FIFO.

PRINCIPLES OF OPERATION

decoding the destination address (DA) and source address (SA) from the DRAM bus (continued)

The node addresses are transmitted on the DRAM bus in the Ethernet native data format. This is the same format in which the address appears in the frame. Ethernet first transmits the least-significant bit of a data byte on the wire. This makes the specific/group bit appear in bit 40 (least-significant bit of most-significant byte). For example, a destination address of 0x12.34.56.78.9A. BC and a source address of 0xDE.F1.23.45.67.89 is seen on the DD terminals as follows:

CYCLE	BIT			
	31 25	24 16	15 8	7 0
1	0x78	0x56	0x34	0x12
2	0xF1	0xDE	0xBC	0x9A
3	0x89	0x67	0x45	0x23

CRC checking and valid frames

The TNETX15AE determines the status of the frame when the EOB, followed by an EOF, is detected. CRC checking is determined from the frame-status field. The code for a good CRC is frame status = 000b. All other frame-status codings indicate that the TNETX3150/TNETX3150A/TNETX3100 aborts the frame due to either a CRC error, a FIFO overflow, or a network error. The frame status is checked only when the option of adding addresses to TNETX15AE tables on a good CRC is indicated rather than immediately adding a new address. If a good CRC is required to add an address, the ADD state machine simply waits from source address time to CRC time. It can be interrupted by a lookup for another port, but no ADD/DEL or other lower-priority activity on any other port can take place. This might cause TNETX15AE to miss adding addresses when expected.

operating logic arbitration

The lookup table is contained in the external SRAM. All of the TNETX15AE logic operations share access to this SRAM. An arbitration scheme is implemented to give all logic operations access to the SRAM while meeting the lookup timing requirements.

The TNETX15AE contains seven logic operations that require the use of the SRAM bus. They are the RAM initialization logic operation (INIT), the lookup logic operation (LKUP), the delete logic operation (DEL), the add logic operation (ADD), the management address-lookup logic operation (FIND), the RAM registers RAMaddr and RAMdata (REG), and the aging logic operation (AGE).

The arbiter assigns a priority to each logic operation. The highest priority is assigned to the INIT logic operation to initialize the TNETX15AE after a reset. After initialization, LKUP becomes the logic operation with the highest priority because it is the logic operation that is the most time critical. The next priority level is shared by ADD and DEL. Register-based accesses (REG) are followed by the FIND logic operation. AGE becomes the lowest priority. Figure 4 shows the priorities of the TNETX15AE logic operations.

The arbiter grants the bus to the logic operation with the highest priority that is currently requesting the bus. Each logic operation requests the bus by asserting its request signal. The arbiter assigns the bus to that logic operation by asserting its grant signal. If no logic operation is requesting the bus, the arbiter grants the bus to AGE for background aging operations.

PRINCIPLES OF OPERATION

operating logic arbitration (continued)

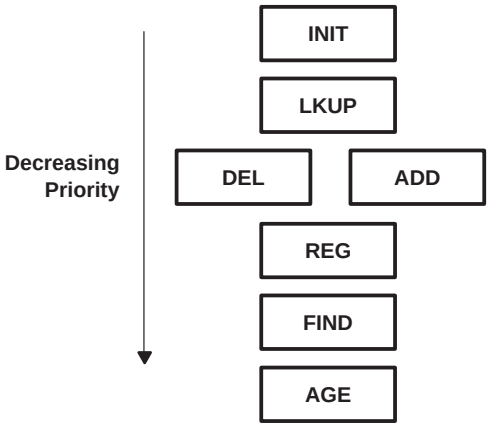


Figure 4. Logic-Operation Priorities

One logic operation can interrupt a lower-priority logic operation to acquire the bus. For example, an LKUP operation interrupts an ADD operation.

For the case of ADD and DEL (with the same priority), the arbiter grants the bus to the first logic operation that requests it. It then grants an uninterruptable bus (unless by a LKUP) to that logic operation until that logic operation is completed. If both ADD and DEL request the bus at the same time, the bus is granted to ADD. This ensures that ADD is not interrupted by a DEL operation and vice versa. This hierarchy explains why a host request for access to RAM completes with some variability — there may be higher-priority operations accessing RAM at that instant.

lookup algorithm

The TNETX15AE device uses a table-based lookup algorithm to provide deterministic lookups with less than 2^{48} memory words. The tables are hierarchical and are linked to the lower tables by threads. Each table can thread to several different tables in the hierarchy. The lowest table in the hierarchy (leaf) does not point to anything and contains information about the address to be matched.

Each level in the hierarchy is assigned to a specific range of bits in the address. The bits in the range are used as an offset within the table at each level. If a thread exists at that offset, the TNETX15AE follows that thread. The TNETX15AE matches an address when it finds a complete thread to a leaf. The thread structure is shown in Figure 5.

The first level (root level) has only one table from which it can branch out to 2^N possible tables. N is the number of bits compared at a given time. Each additional table in the hierarchy branches down to 2^N other possible tables. The second level contains 2^N table and 2^{2N} threads. The third level contains 2^{2N} tables and 2^{3N} threads, and so on.

Because of this exponential growth, the threads and the amount of possible paths at each level soon overtake the number of addresses required. If this growth were to go unchecked with an N of 5, the third level would contain 1,024 tables and 32,768 threads. If only 1024 addresses are required, there are more tables allocated than needed, and most contain NULL pointers.

PRINCIPLES OF OPERATION

lookup algorithm (continued)

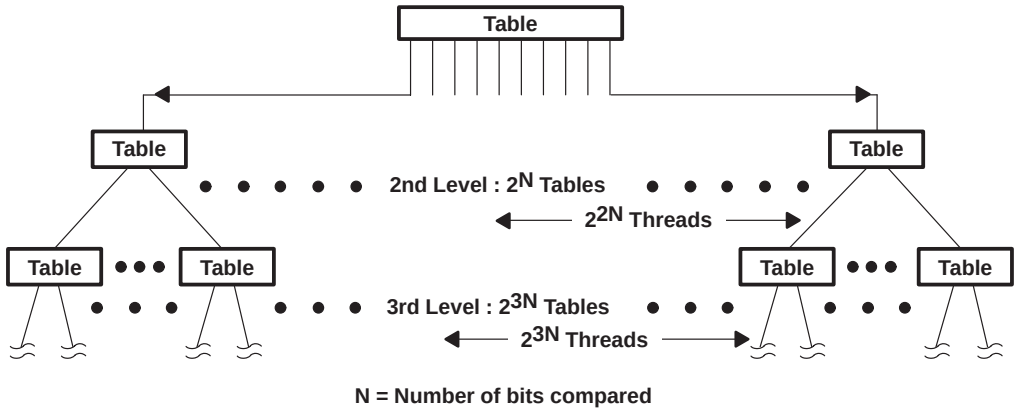


Figure 5. Lookup-Algorithm Table Hierarchy

For bit groups of five (gives table sizes of 32), and with five levels of table hierarchy, preassigning all possible tables at initialization requires more RAM than is possible to attach to the TNETX15AE. Therefore, table allocation occurs only when a new table is required as a result of discovering a bit pattern at any of the levels previously unseen. This has several side effects – given the same address set, entering them into the table in a different order yields different SRAM contents as the tables required are allocated from free space in a different order; the pointer values written to the table entries to point to the next level in the hierarchy are dynamically allocated. An address is in the table if there are valid entries at each level in the hierarchy (as each n-bit part of the address is inspected) that continue all the way down to a leaf where the characteristics of that address are stored. An address is not present in the table when there is a null pointer at any level of the decode. Addresses that are sequential share all the intermediate pointers down to the address of the leaf. Addresses with at least one bit different in each N-bit decode group occupy completely different tables. The worst-case rating on the storage capacity of the TNETX15AE SRAM assumes that each address is different from all the others by at least one bit in each decode group. This implies that each card is from a different manufacturer, and the serial number of any card is not within 2^N of any other card.

Since each table needs to compare 2^N possible combinations, it requires 2^N pointers. Each table has the format shown in Table 5.

Table 5. Lookup-Algorithm Table Format

OFFSET	N (BITS TO BE COMPARED)	POINTERS
0	00000	To table x at next level
1	00001	To table y at next level
2	00010	To table z at next level
•	•	•
•	•	•
•	•	•
$2^N - 1$	$2^N - 1$	

Only the pointers column of Table 5 occupies memory locations.

PRINCIPLES OF OPERATION

lookup algorithm (continued)

Each pointer points to a table in the next level. The TNETX15AE uses N bits in the address as an offset to this table and, if a valid pointer is found, decode proceeds to the next level. A null pointer indicates that the entry was not found (in this case the search fails). The width of the memory word (W) must increase as the memory gets deeper, such that $2^{N+W} - 1 \geq$ memory size; the memory address of the next pointer is (contents of current table entry):(next N bits to decode).

a graphical example of the lookup algorithm

This example uses the following method to look up the number 0xB2 (10.11.00.10b), two bits at a time (N = 2). Graphically, this number is represented in Figure 6.

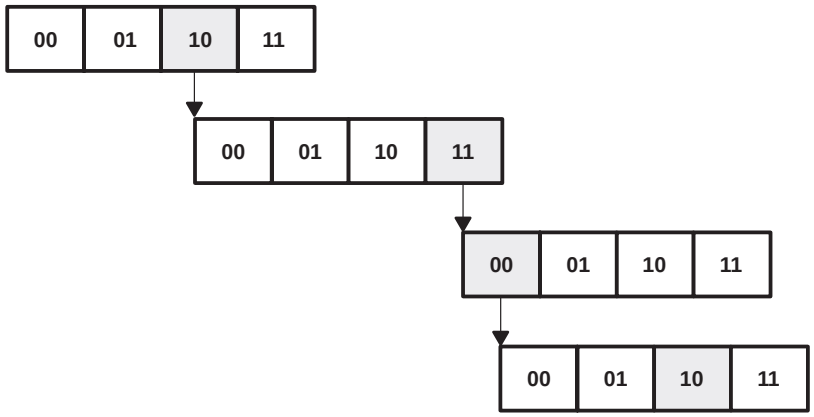


Figure 6. Example of Lookup-Number Algorithm (Matching 0xB2)

The first table at offset 0x10b points to the second level (see Figure 6). The pointer at the second level (offset by the second set of bits (0x11b)) points to the third table. This process continues until the last two bits are matched. Matching 0xB2, two bits at a time, uses four tables, with each containing four possible pointers. Not all locations in the tables are used, which potentially can lead to unused memory. Now, add 0xB0 (10.11.00.00b) to the table. Figure 7 illustrates the results.

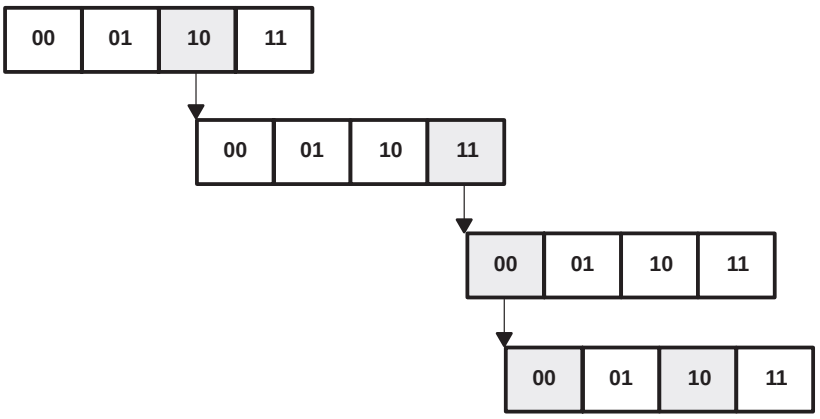


Figure 7. Example of Lookup-Number Algorithm (Adding 0xB0)

0xB0 follows exactly the same thread as 0xB2 and the only difference between the two is in the last table. 0xB0 matches offset 00b while 0xB2 matches offset 10b. There are now two numbers being represented, but there are still four tables allocated.

PRINCIPLES OF OPERATION

a graphical example of the lookup algorithm (continued)

Continuing this example, we can add 0xB1 and 0xB2 with the same number of tables allocated. This is called the best-case scenario since the maximum amount of addresses can be stored in the minimum amount of memory. Now add 0x22 (00.10.00.10b) to a lookup table containing only 0xB2. Figure 8 shows the results.

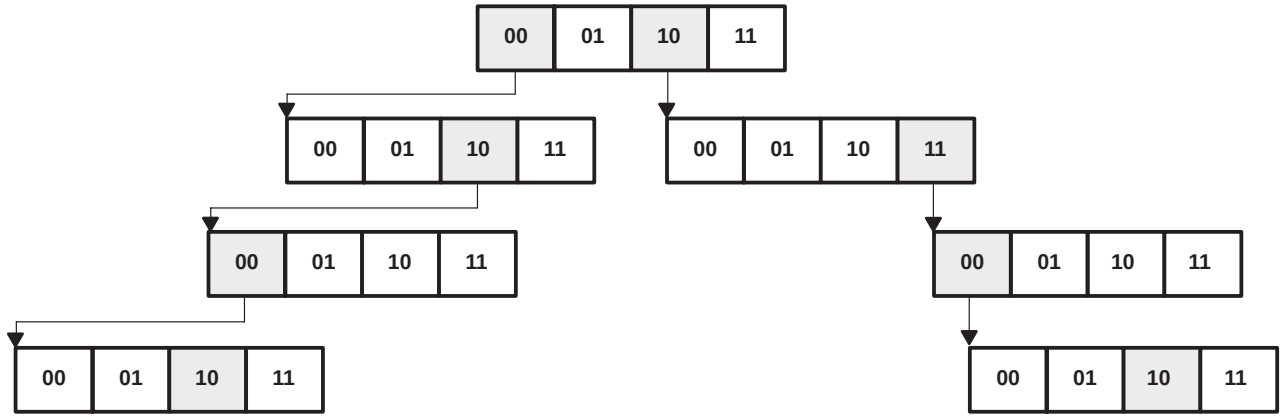


Figure 8. Example of Lookup-Number Algorithm (Adding 0x22)

Adding 0x22 requires allocating three additional tables. Seven tables are now required to hold two addresses. Compared to numbers that differ in their least-significant bits, numbers that differ in their most-significant bits require more tables. Continuing the example, adding 0x2A requires an additional three tables, as would 0xE2. This is the worst-case scenario of storing addresses with this technique.

The TNETX15AE is designed to handle the worst-case address distribution. The worst-case address distribution requires a separate thread per address. A purely random distribution creates multiple threads at the top levels. The most-significant bits of a network card address are the vendor bits. However (in most networks) there are only a couple of vendors used. These cards do not have a purely random address distribution, and they all share a common set of bits that identifies the vendor. This configuration requires fewer pointers for the same number of addresses. In such a network, the tables look more like those in Figure 9.

There is a need to allocate for worst case, but since the worst case is not likely to happen in a real system, the opportunity exists to include more addresses than indicated by the worst-case rating. The actual number of addresses supported depends on the nature of the nodes in the network. TNETX15AE devices in networks with nodes from one or a small number of manufacturers can recognize more addresses than those in a purely random-address network.

This algorithm has the primary advantage that the lookup time is independent of the number of addresses stored in the lookup table. Whether the number of addresses present is one or a half million, the lookup time depends on the number of table levels required to match the address.

PRINCIPLES OF OPERATION

a graphical example of the lookup algorithm (continued)

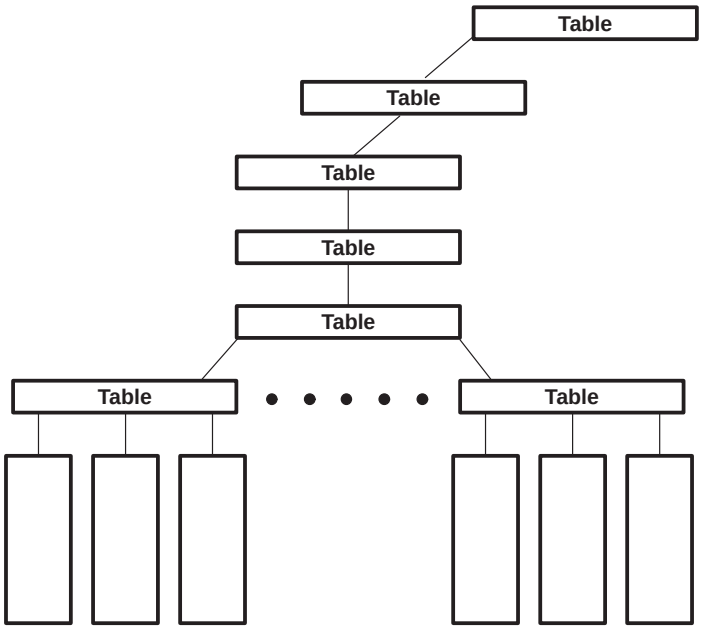


Figure 9. Address Distribution Table Hierarchy (Worst Case)

Another side effect of this algorithm is that the table order is already sorted (smallest to largest).

The choice of N (the number of bits to consider each cycle) must be an integer and is a balance of:

1. The size of each table = 2^N words; larger means more wasted space per lightly used table, and smaller means less wasted space per lightly used table.
2. The clocks required to process a whole address (in this case 48 IEEE addresses) = 48 ÷ N rounded up to the next integer; considering more bits at a time decodes 48 bits in less time (considering fewer at a time takes more time). The upper limit is 10 or 11 clocks.

SRAM data storage

lookup table SRAM allocation

The TNETX15AE uses a 5-bit version of the lookup algorithm described in the previous section.

When a 5-bit version of the lookup algorithm is used, each address requires ten tables to store a 48-bit value. Tables 0 through 9 are used to match the 48 bits involved in the lookup. Table 9 (shown in Figure 10) contains the actual data describing an address.

The external SRAM must be a minimum of eight bits wide, so address descriptor information (see *findport* and f*indnodeage* register definitions) can be stored in four locations as bytes. Starting at 16K, the external SRAM must get wider so that a pointer of installed word width (when five bits are appended) gives a pointer with enough bits to address any memory location. Both address descriptor blocks and pointer tables must be contiguous, and both are allocated dynamically. The table is full when the next structure being allocated does not fit. Several addresses may have to be deleted on a table-full condition to make room for a new address.

An allocated (but unused) table entry is set to 0 since zero cannot be a valid address for a table leaf.

PRINCIPLES OF OPERATION

lookup table SRAM allocation (continued)

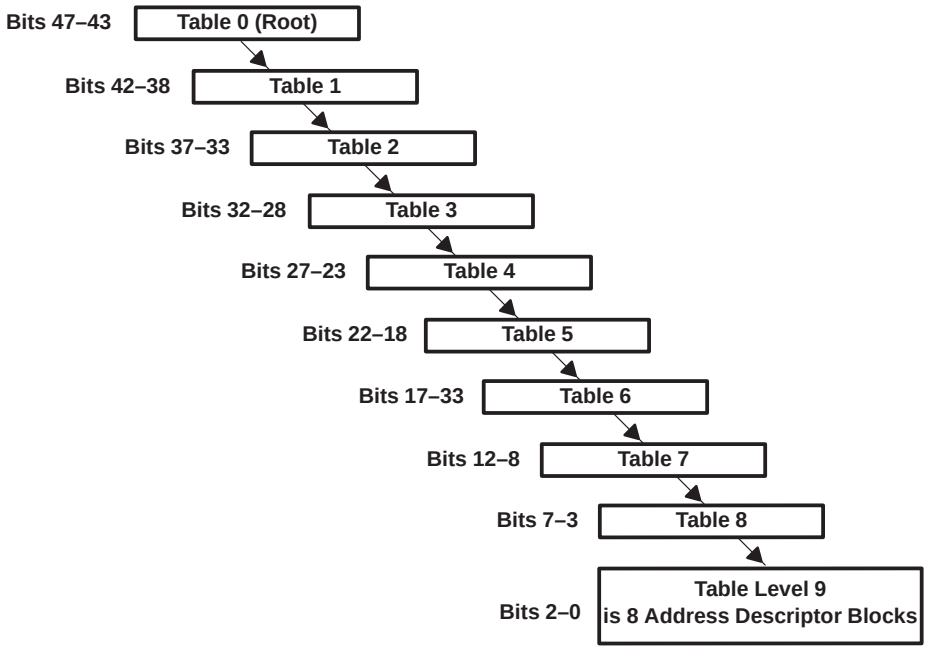


Figure 10. Address Tables

The last level decodes only bits 2–0 of the address, therefore, only eight unique locations are required. Since tables are allocated 32 words at a time, the last table is divided into eight four-word units. The lower eight bits of each word hold one byte of the address descriptor block as shown in Figure 11.

BYTE 1	BYTE 2	BYTE 3	BYTE 4
Flags/port code	Reserved	MSG age stamp	LSB age stamp

Figure 11. Address Descriptor Block Format

Since only the first byte of the address descriptor block is required to make a forwarding decision, an address lookup requires 40 ns to access each of 10 levels for a total 400-ns lookup time plus an internal setup cycle. The total time required (400 ns + 20 ns) gives some margin to generate the EAM code in time for the TNETX3150/TNETX3150A/TNETX3100.

PRINCIPLES OF OPERATION

RAMsize and number of nodes supported

The address capability for the various RAM sizes is given in Table 6 (repeated from RAMsize register definition).

Table 6. Address Capability for RAM Sizes

RAMsize REGISTER	RAM SIZE	WORST CASE	BEST CASE
0x00	640 × 8	2	88
0x01	832 × 8	2	136
0x02	1K × 8	3	184
0x03	2K × 8	7	432
0x04	4K × 8	14	928
0x05	8K × 8	28	1920
0x06	16K × 9	59	3904
0x07	32K × 10	123	7880
0x08	64K × 11	251	15816
0x09	128K × 12	507	31688
0x0A	256K × 13	1019	63432
0x0B	512K × 14	2189	134 040
0x0C	1M × 15	4530	277 408
0x0D–0x0F	2M × 16	9211	564 144

The data in Table 6 shows a large range between the worst-case performance and the best-case performance. For a 500-address system, 128K × 12 is needed. A 1000 address system requires 256K × 13. A 2000 address system requires 512K × 14, where K = 1024 for binary-sized memories.

The difference between the best and worst case at each row depends on the similarity (when viewed in 5-bit groups) of the addresses entered into the table. If addresses are sequential, there is maximal table sharing, and the most addresses fit into the smallest space. If addresses are widely spaced with respect to the 5-bit groups, there is minimal table sharing, because each 32-entry table has only one entry at each level. This requires the most SRAM per address.

register spaces/external devices accessible through the TNETX15AE

Although the TNETX15AE is designed to work in a CPU-less environment, access to the internal registers is useful for the following reasons:

- Setting/changing port assignments
- Setting operational modes (startup options)
- Resetting the device (software reset)
- Management-based access and control of the lookup table
- Statistics gathering
- Diagnostic operations
- Communicating with attached PHYs through the MII
- Reading/writing to an external EEPROM

PRINCIPLES OF OPERATION

register spaces/external devices accessible through the TNETX15AE (continued)

Figure 12 shows the various register spaces provided by and accessed through the TNETX15AE.

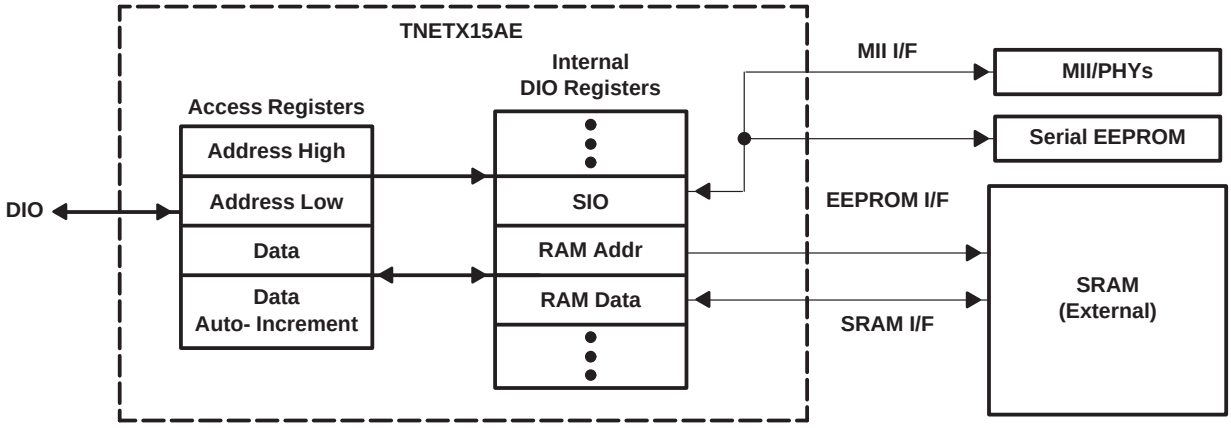


Figure 12. Register Spaces Provided and Accessed Through the TNETX15AE

DIO interface

The DIO interface is asynchronous to allow easy adaptation to a range of microprocessor devices and computer system interfaces. The DIO interface is designed to operate from the same bus as the TNETX3150/TNETX3150A/TNETX3100 DIO interface. Thus, both devices are accessed using the same DIO read and write routines. Each device is selected for DIO reads and writes through independent chip-select signals. Chip select for the TNETX3150/TNETX3150A/TNETX3100 is named $\overline{\text{SCS}}$, while chip select for the TNETX15AE is named $\overline{\text{ESCS}}$. Figure 13 shows how the TNETX15AE and the TNETX3150/TNETX3150A/TNETX3100 share the DIO interface.

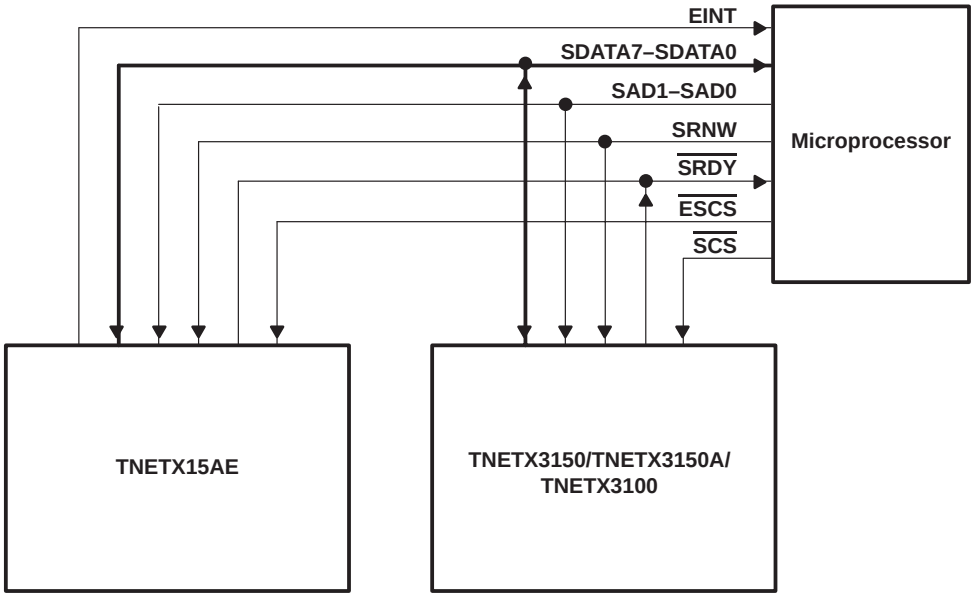


Figure 13. DIO Interface Bus

PRINCIPLES OF OPERATION

DIO write cycle

The write cycle waveforms are shown in Figure 14 and described in the following:

- The TNETX15AE host register address (SAD1–SAD0 and SDATA7–SDATA0) is asserted and SRNW goes low.
- After a setup time, $\overline{\text{ESCS}}$ goes low, initiating a write cycle.
- The TNETX15AE pulls $\overline{\text{SRDY}}$ low as the data is accepted.
- SDATA7–SDATA0, SAD1–SAD0, and SRNW signals are deasserted after the hold time has expired.
- $\overline{\text{ESCS}}$ goes high (by the host) to complete the cycle, causing $\overline{\text{SRDY}}$ to deassert and to go high for one cycle before going into the high-impedance (Z) state.

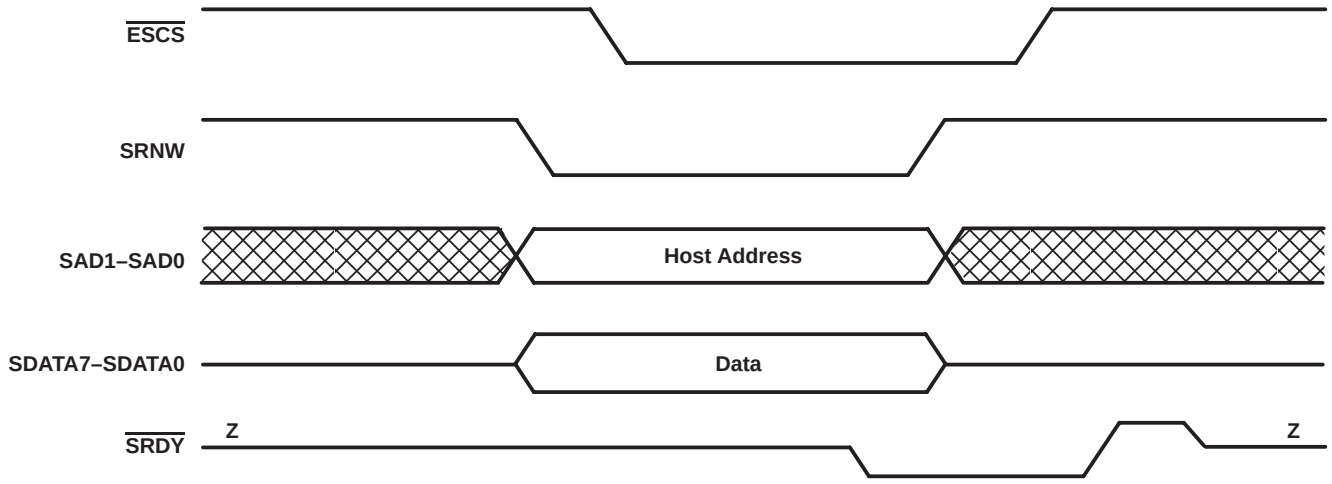


Figure 14. DIO Write-Cycle Waveforms

PRINCIPLES OF OPERATION

DIO read cycle

The read-cycle waveforms are shown in Figure 15 and described in the following:

- The TNETX15AE host register address SAD1–SAD0 is asserted while SRNW is held high.
- After a setup time, $\overline{\text{ESCS}}$ goes low, initiating the read cycle.
- After a delay time, with $\overline{\text{ESCS}}$ low, SDATA7–SDATA0 is released from the high-impedance (Z) state. SDATA7–SDATA0 is driven with valid data and $\overline{\text{SRDY}}$ goes low. The host can access the data.
- $\overline{\text{ESCS}}$ goes high (by the host) to signal completion of the cycle, causing $\overline{\text{SRDY}}$ to deassert. $\overline{\text{SRDY}}$ goes high for one clock cycle before going into the Z state. SDATA7–SDATA0 also goes to the Z state.

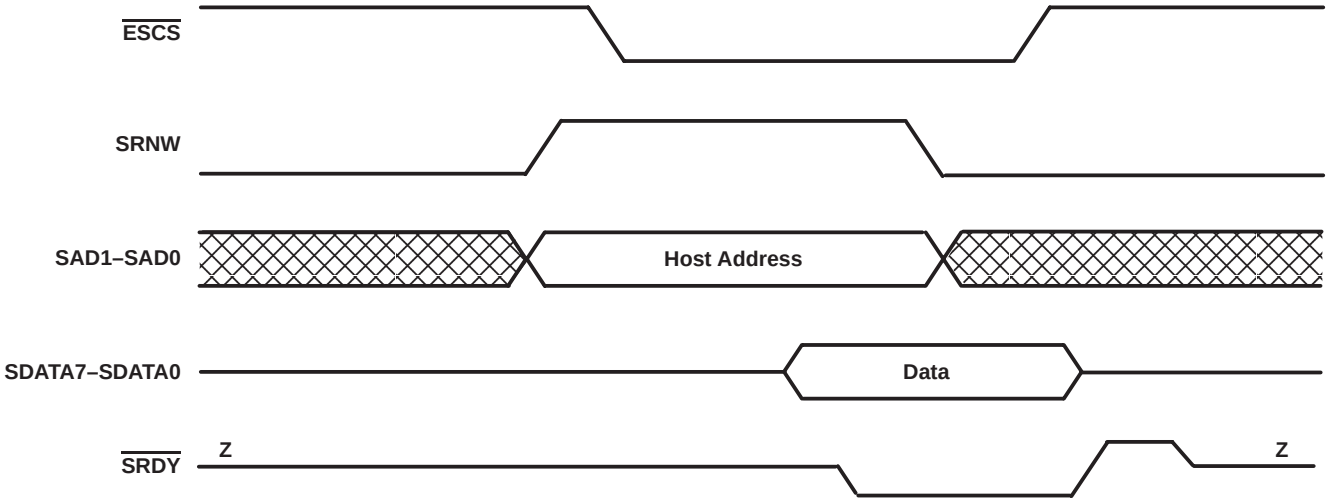


Figure 15. DIO Read-Cycle Waveforms

host (access) register space

The TNETX15AE registers, SRAM (internal and external), and EEPROM are indirectly accessed through the access registers. The access registers are written/read to/through the DIO interface. There are four byte-wide access registers. They are individually selected through the SAD bus, as shown in Table 7, and the registers are read/written through the SDATA bus.

Table 7. DIO Access Register Code

SAD1	SAD0	ACCESS REGISTERS
0	0	DIO address low
0	1	DIO address high
1	0	DIO data
1	1	DIO data auto-increment

Two bytes, DIO address low and DIO address high, are used as a pointer to the address (DIOADR) of the internal register being selected. DIO address high is the most-significant byte of DIOADR and DIO address low is the least-significant byte. The DIO address register is byte writeable. The user does not have to write to both DIO address locations for each access to the internal registers. Up to 2^{16} possible internal locations can be accessed through the DIO address registers. Some of these first-level indirect registers are themselves address registers (RAMaddr) for second-level indirect access to even bigger address spaces.

PRINCIPLES OF OPERATION

host (access) register space (continued)

The next two bytes, DIO data and DIO data auto-increment, are used to read and write data to the byte-wide internal register selected in the DIOADR. Both DIO data and DIO data auto-increment can be used effectively to read and write the data, but the DIO data auto-increment register provides additional functionality over DIO data. Access to the DIO data auto-increment register provides a post-increment (by one) to the DIO address register. This is useful for reading/writing to a block of registers.

As an example, to access a single-byte-wide register such as the SIO register (DIO address = 0x0A) the operations needed are:

- Write a 0x00 to DIO address high
- Write a 0x0A to DIO address low to select DIO address 0x0A
- Read the SIO register by reading DIO data, or write to the SIO register by writing to DIO data.

Multiple-byte registers are accessed by reading/writing to their individual bytes. The control register (DIO address 0x08–0x09) is accessed in the following manner:

- Write a 0x00 to DIO address high
- Write a 0x08 to DIO address low to select DIO address 0x08
- Read the least-significant byte of the control register by reading DIO data, or write to the least-significant byte of the control register by writing to DIO data.
- Write a 0x00 to DIO address high
- Write a 0x09 to DIO address low to select DIO address 0x09
- Read the most-significant byte of the control register by reading DIO data, or write to the most-significant byte of the control register by writing to DIO data.

Improvement on the preceding steps is obtained by writing a 0x00 to DIO address high and changing only DIO address low. More steps can be eliminated by using the DIO data auto-increment register to read or write to contiguous register bytes. The following shows how to use the auto-incrementing function to access the control register.

- Write a 0x00 to DIO address high
- Write a 0x08 to DIO address low to select DIO address 0x08
- Read the least-significant byte of the control register by reading DIO data auto-increment or write to the least-significant byte of the control register by writing to DIO data auto-increment. The address in DIO address auto-increments to 0x0009.
- Read the most-significant byte of the control register by reading DIO data auto-increment, or write to the most-significant byte of the control register by writing to DIO data auto-increment.

The auto-incrementing function is useful when reading or writing to a large number of adjacent registers, such as the 48-bit address registers or when reading the statistics block.

PRINCIPLES OF OPERATION

internal registers

The internal registers are used to initialize and/or software-reset the TNETX15AE, select the TNETX15AE startup and routing options, check the number of nodes within the TNETX15AE and statistics, enable management-based operations on the lookup table, and interface with the on-chip or external SRAM, the EEPROM, and any MII-managed devices.

The internal registers are described in the detailed *internal register* section of this document.

The following sections provide additional information on the use of internal registers to access the SRAM, MII devices, and EEPROM.

lookup-table SRAM access

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
		RAMsize		0x00
	RAMaddr			0x20
		RAMdata		0x24

The TNETX15AE SRAM can be accessed with the RAMaddr and RAMdata registers. The algorithm for reading and writing to the RAM is similar to that for reading and writing to the internal registers; the address of the location to access is placed in the RAMaddr register and the data is read from or written to the RAMdata register.

This interface also has an auto-increment function, which is selected from the INC bit in the RAMaddr register. The auto-incrementer increments by one when the most-significant byte of the RAMdata register is read or written to.

The DIO state machine must request access to SRAM via the SRAM bus scheduler. A write to SRAM is queued when the most-significant byte of the RAMdata register is written. A read of SRAM is queued when either byte of the RAMdata register is read through the DIO port.

serial interface – MII-managed devices

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
	SIO			0x08

The TNETX15AE gives the programmer an easy way to implement a software-controlled bit-serial interface. This interface is most appropriate in implementing an MII management serial interface.

MII devices, which implement the management interface, consisting of MDIO and MDCLK, are accessed in this way though the SIO register. In addition (for PHYs that support this) the TNETX15AE implements a MII-management signal, $\overline{\text{MRESET}}$, to hardware-reset MII PHYs.

MDIO requires an external pullup for operation. The I/O direction is controlled by the MTXEN bit, and the external terminal state is set from or read in MDATA. The complete serial interface (MDIO, MDCLK, $\overline{\text{MRESET}}$) can be placed in a high-Z state through the MDIOEN bit in SIO.

The TNETX15AE does not implement any timing or data structure on its serial interface. Appropriate timing and frame format must be assured by the management software by setting or clearing bits at the right time and in the right order. Refer to the IEEE Std 802.3u specification and the data sheet for the MII-managed device for the nature and timing of the MII waveforms.

PRINCIPLES OF OPERATION

x24C02 EEPROM interface

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
	SIO	Control		0x08

The flash EEPROM interface is provided so the system-level manufacturers can provide a preconfigured system to their customers. Customers can change or reconfigure the system and retain preferences between system power downs. The flash EEPROM contains configuration and initialization information that is accessed infrequently (typically at power up and reset). The host can write to the EEPROM through this SIO register interface.

The TNETX15AE implements a two-wire serial interface, consisting of EDIO and EDCLK that communicates with the EEPROM. Similar to the MII interface, the TNETX15AE does not implement any timing or data structure on its serial interface. Appropriate timing and frame format must be ensured by the management software by setting or clearing bits at the right times. There are no limits to the size or organization of the EEPROM when the host is driving this interface, but these same terminals are used by an internal state machine to load the contents of the EEPROM into internal registers. This state machine expects a 24C02 (as device 0) on the serial bus. Refer to the manufacturer's data sheet for a description of the EEPROM waveforms.

If an EEPROM is not installed, EDIO should be tied low. For EEPROM operation, EDIO and EDCLK require an external pullup (see EEPROM data sheet). The TNETX15AE detects the presence or absence of the EEPROM and indicates this in the NEEPM bit of the control register.

initialization

The TNETX15AE can be designed for stand-alone use, with no need for a management CPU. It can be reset and initialized with or without a host CPU. This section discusses the steps necessary to bring the TNETX15AE up to an operating level.

TNETX3150/TNETX3150A/TNETX3100 initialization

If multiple-port EAM codings are used, then TNETX3150/TNETX3150A/TNETX3100 IOBMOD bit in the systcl register must be set to a 1. If reversed (IOBMOD in TNETX3150/TNETX3150A/TNETX3100 systcl register = 0), the ability to send frames to multiple ports, but not all ports, is lost. Multicasts become broadcasts to all ports, regardless of EAM code. Also, the broadcast is not in order with other traffic on each transmit queue.

The user also must disable the TNETX3150/TNETX3150A/TNETX3100 internal-address matching when using the TNETX15AE. This is accomplished by writing a 1 to the ADRDIS bit in the port control register of each port.

resetting the TNETX15AE

hardware reset

The TNETX15AE is hardware reset by asserting $\overline{\text{RESET}}$ low. The TNETX15AE comes out of reset when $\overline{\text{RESET}}$ goes high.

During a hardware reset, no access to the internal registers is allowed. A read or write stalls, that is, $\overline{\text{SRDY}}$ does not come true until the hardware state machine is done. All access registers and internal registers are initialized to default values. Hardware reset should complete in a few tens of clock cycles.

If an EEPROM is detected (after a hardware reset), the TNETX15AE begins the EEPROM auto-loading process. No DIO operations are allowed during auto-loading.

PRINCIPLES OF OPERATION

software reset

The TNETX15AE is software reset by setting the RESET bit to a 1 in the control register (0x00,0x09 DIO). The TNETX15AE remains in the reset state until this bit is cleared. All internal registers are initialized to default values during a software reset, except for the control register, which keeps its current value. Reading the internal registers is allowed during a software reset, but the user cannot write to any register (except for the control register).

The EEPROM auto-loading process does not start during a software reset. The user must assert the LOAD bit in the control register for auto-loading to start.

EEPROM initialization/automatic loading (LOAD)

The TNETX15AE automatically loads selected registers from an attached 24C02 EEPROM after a hardware reset or when the LOAD bit in the control register is set. Up to eight 24C02 EEPROMs can be connected across the same serial interface. They are distinguished by separate addresses that are selectable by pulling address terminals up or down on each EEPROM. The TNETX15AE requires the automatic loading information in device number 0x00, starting at location 0x00.

The TNETX15AE determines if the EEPROM device is present. Several conditions cause the TNETX15AE to determine that a device is not present. If EDIO is pulled down, the EEPROM is assumed to be not present. If the EEPROM fails to acknowledge (ACK) on data writes, the EEPROM is not present. If the cyclic redundancy check (CRC) in the EEPROM does not match the internally calculated CRC, the EEPROM is assumed to be not present.

When no EEPROM is detected, The TNETX15AE sets the NEEPM bit in the control register to a 1, and the TNETX15AE is placed in a reset state (RESET and NEEPM are set in the control register) while the registers assume their default values.

The automatic loader reads the register values from the EEPROM and programs the TNETX15AE accordingly. The last register written is the control register.

The automatic loader can initialize and start up the TNETX15AE if the START bit in the control register is programmed in the EEPROM. This allows for initialization and startup without a host CPU.

During the automatic loading, no DIO operations are permitted. Both reads and writes stall, that is, $\overline{\text{SRDY}}$ does not come true on a pending operation until the EEPROM load cycle is over. The load cycle should take (bits per word access on the EEPROM) $\times$ (words read) $\times$ 100 kHz, or 27 $\times$ 43 $\times$ 100 kHz, or 12 ms. To sense the end, wait for a sample register read to complete, or wait 15 ms.

If load is followed by start, then the INITD bit in the control register can be used to indicate when all the setup is done. The download bit, LOAD bit, RESET bit, and any other read only or reserved bits cannot be set during automatic loading. The CRC for the EEPROM is calculated using the information written in the EEPROM although information may not be written to the TNETX15AE. For example; a value of 0x8F or 0xFF in the EEPROM for RAMsize is written as 0x0F in the TNETX15AE since bits 7, 6, 5, and 4 are reserved, but the calculated CRC for the EEPROM for each case is different.

PRINCIPLES OF OPERATION

EEPROM initialization/automatic loading (LOAD) (continued)

The last four bytes read by the automatic loader correspond to a 32-bit CRC value for the information stored in the EEPROM. The CRC value is calculated by using the following callable C routine, which assumes that the data to be examined is loaded into an array eeprom_data [ ]:

```
void ifexeeprom (char *cs)
   {
   long crc;
   int k;

   crc = 0xffffffffl;
   for (k=0;k<=38;k++)
      {
      crcbyt (eeprom_data[k], &crc);
      }

   crc ^= 0xffffffffl;
   eeprom_data[k++]=(int) ((crc >> 24) & 0x0ffl);
   eeprom_data[k++]=(int) ((crc >> 16) & 0x0ffl);
   eeprom_data[k++]=(int) ((crc >>  8) & 0x0ffl);
   eeprom_data[k++]=(int) ((crc      ) & 0x0ffl);
   }

crcbyt (dat,crc)
int dat;
long *crc;
   {
   int i;

   for (i=0;i<8;i++)
      {
      crcbit(dat>>7,crc);
      dat = dat <<1;
      }
   }

crcbit (dat,crc)
int dat;
long *crc;
   {

   if ( (((*crc>>31) & 11)^((long)dat & 11)) ==1)
      {
      *crc ^= 0x02608edbl;
      *crc = *crc <<1;
      *crc |= 0x00000001l;
      }
   else
      {
      *crc = *crc <<1;
      *crc &= 0xfffffffel;
      }
}
```

In this example, the values for which the CRC is calculated are placed in the `eeprom_data[]` array from offset 0 to 38 (0x26). The routine `crcbyt` is called for each byte. The C routine then places the calculated CRC values in the `eeprom_data[]` array at locations 39 to 42 (0x27 to 0x2A).

PRINCIPLES OF OPERATION

TNETX15AE operational modes

The TNETX15AE operational modes are selected through the control register. They are used to control decision points in the logic operations. The modes available through the control register are NAUTO, NCRC, and NLRN0. All three of these modes can be changed after the START bit is set high.

NAUTO mode

The not automatically add-address (NAUTO) mode is implemented to give the management CPU complete control of the lookup table. It does so by disabling the two automatic processes that can affect the lookup table, wire additions, and aging.

NAUTO to 1 disables adding addresses to the TNETX15VE tables learned from packets passing through the TNETX3150/TNETX3150A. All addresses in the TNETX15VE must be added through the DIO interface (see ADD in internal register section). No interrupt is generated on addresses added through the DIO interface.

NEW interrupts in this mode signal that the address-checking state machine has seen an address on the wire that is not in the table.

NAUTO also affects the AGE logic operation by disabling it. AGE does not delete nodes from the table even if the table is full. It is the management's responsibility in this mode to maintain the addresses in the lookup table. Table-full conditions are determined through a FULL interrupt, or when a host-initiated ADD stalls.

NCRC mode

The no-CRC (NCRC) check mode enables the TNETX15AE to add addresses learned from packets before the CRC is checked. It is a performance-boosting feature for the ADD state machine that allows it to perform more ADD logic operations in the same amount of time. This allows the TNETX15AE to add more efficiently and keep the aging time stamp current on nodes that do not talk frequently on the network, avoiding unnecessary aging.

The tradeoff in this mode is the possibility that corrupt addresses could be added to the lookup table. This condition should not become critical unless aging is turned off.

NRLN0 mode

When NRLN0 is set to 1, no new source address from a packet entering the switch on port 0 is added to the table. This feature, combined with unkuniport and unkmultiport registers, allows the TNETX15AE to be at the edge of a large network and only learn direct-attached station addresses, not the whole network above the uplinked port.

aging modes (agingtimer register)

The TNETX15AE implements an autonomous aging logic operation. There are two modes of aging and these modes are controlled through the agingtimer register. The two modes are time-threshold aging and table-full aging.

Time-threshold aging is set by writing a value between 0x0001 and 0xFFFE to this register. In this mode, the register value gives the time in seconds that is used by the AGE logic operation to delete addresses. This provides an aging range of 1 second to 18 hours. Addresses older than this are deleted.

Table-full aging is set by writing 0x0000 or 0xFFFF to the register. In this mode, the AGE logic operation ages out addresses only when the table is full and the ADD logic operation needs to add an address. AGE deletes the oldest address in the table. To identify the oldest address, the time stamp stored in each address descriptor is incremented with each address added instead of each second of time.

PRINCIPLES OF OPERATION

lookup table SRAM initialization (START)

The lookup table is automatically initialized by the TNETX15AE, with no need for an external processor. The steps for initializing are:

- Write to RAMsize the size of the external SRAM. Writing to RAMsize can be performed by a CPU before START is set or via the EEPROM auto-load operation.
- Assert the START bit in the control register. This is accomplished either by the CPU or EEPROM.
- The TNETX15AE indicates the completion of the lookup-table initialization by setting the INITD bit in the control register to 1.

The TNETX15AE clears the lookup table by writing 0x0000 to all available locations and then initializes the memory data structures. After these operations are done, the TNETX15AE performs lookups, adding, and aging operations as directed by mode bits.

frame-forwarding LKUP logic operation

The lookup (LKUP) logic operation is designed for two important tasks: 1) perform time-critical lookups of the wire within the TNETX3150/TNETX3150A/TNETX3100's allotted time, and 2) forward the frame to the right ports. The LKUP logic operation works independently of all other logic operations. To meet the timing requirements, this logic operation occupies the highest priority on the SRAM bus after SRAM is initialized for looking up destination addresses. LKUP performs a lookup on the destination address found from the DRAM interface.

If an address is found, the TNETX15AE uses the information contained in the lookup table for routing. If the address is not found, the TNETX15AE floods the frame to the ports indicated in the unkmultiports or unkuniports register. The LKUP logic is also used to check for the presence of the source address in the table.

management-based lookups (FIND)

The LKUP logic operation also is called when the DIO interface sets the FIND (LKUP) bit in the findcontrol register. The FIND logic operation is designed to give the programmer a simple way to find addresses within the lookup table. FIND is controlled from the following internal registers:

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Findnode23–Findnode16	Findnode31–Findnode24	Findnode39–Findnode32	Findnode47–Findnode40	0x0C
Findport		Findnode7–Findnode0	Findnode15–Findnode8	0x10
	Findcontrol	Findnodeage		0x14

The interface provides a 48-bit read or writeable register findnode in which the address is placed; a 16-bit register findport in which routing information is placed; and a 16-bit register findnodeage, which contains the age time stamp of the node being looked up.

PRINCIPLES OF OPERATION

FIND, FINDFIRST, and FINDNEXT commands

There are three find commands supported by the TNETX15AE: FINDFIRST, FINDNEXT, and FIND(LKUP). These are used to interrogate the table that the TNETX15AE built as a result of all ADDS. FIND works on all addresses.

FIND indicates completion by clearing its command bits. The findcontrol register keeps the code that the user has written until the process is complete. For example, if the user writes a 0x01 to findcontrol (LKUP) then this register reads 0x01 until the FIND logic operation is finished operating.

If an address is found as a result of the command just completed, it is indicated by asserting the FOUND bit in findcontrol. The FOUND bit indicates that the information in the findnode, findport, and findnodeage registers is valid.

The FIND command finds a specific user-defined address in the lookup table. The procedure for the FIND command is as follows:

- Write the 48-bit address query in the findnode register.
- Set the LKUP bit in the findcontrol register. The TNETX15AE scans the lookup table for that particular address.
- Poll the findcontrol register until the LKUP bit is cleared. A single read is necessary as the SRDY is delayed until the operation is complete.
- Read the findcontrol register. If FOUND is set, the address was found and the node's information is placed in the registers. If FOUND is not set, the address was not found within the lookup table.

The FINDFIRST command finds the first address contained in the lookup table. The procedure for the FINDFIRST command is as follows:

- Set the FIRST bit in the findcontrol register. No write to findnode or findport is required. The TNETX15AE scans the lookup table for the first address.
- Poll the findcontrol register until the FIRST bit is cleared. A single read is necessary as the SRDY is delayed until the operation is complete (up to 6.4 μs).
- Read the findcontrol register. If FOUND is set, an address was found and the node's address and information is placed in the registers. If FOUND is not set, an address was not found, i.e., the lookup table is empty.

The FINDNEXT command finds the next address from that contained in the findnode register. The user can either write a value in findnode and find the next address or keep the current value and continue finding next addresses. The procedure for the FINDNEXT command is as follows:

- Write the starting address in findnode (if desired) or keep the currently held address.
- Set the NEXT bit in findcontrol. The TNETX15AE scans the lookup table for the next address after the one contained in findnode.
- Poll the findcontrol register until the NEXT bit is cleared. A single read is necessary as the SRDY is delayed until the operation is complete (up to 6.4 μs).
- Read the findcontrol register. If FOUND is set, and the address was found, the node's address and information is placed in the registers. If FOUND is not set, and an address was not found, the rest of the lookup table is empty.

The commands can be combined sequentially to quickly read the whole address table. All that is required is a FINDFIRST, followed by FINDNEXT commands, until no more addresses are found.

PRINCIPLES OF OPERATION

summary of FIND operations supported

Table 8 summarizes the supported FIND operations, the bits used to select the operation, and the code written to the findcontrol register.

Table 8. FIND Operations

OPERATION	BITS USED	FINDCONTROL
FIND (LKUP)	LKUP	0x01
FINDNEXT	NEXT	0x02
FINDFIRST	FIRST	0x04

adding an address

The ADD logic operation is responsible for new address additions to the lookup table, address port changes, modifying the information stored in the lookup table, and keeping the address time stamp current. The TNETX15AE implements a single ADD logic operation and shares it between automatic adds from the wire and host port additions. The TNETX15AE prioritizes wire adds over management adds. It completes an add request before starting another request.

The ADD algorithm is summarized as follows:

- ADD performs a lookup to determine if the address exists in the table.
- If the address exists, ADD verifies that the port assignment has not changed. If the port assignment changes, ADD updates the port. In all cases, ADD updates the AGE time stamp.
- If the address does not exist, ADD adds the address to the table with the current time stamp.

Currently, adding an address requires available lookup tables. It is possible that during the ADD operation no more lookup tables are available for address additions. A FULL interrupt is issued to the host, and the ADD operation stalls, pending the resolution of the no-space-available situation. In this situation, the following occurs.

- If the NAUTO bit is set to 1 (host-managing table), the host deletes addresses in the table until enough are deleted to make space for the new address. Depending on the shared tables used by the addresses being deleted, and the tables that are shared by the inbound address, it is possible that many addresses may have to be deleted for the new address to fit. When enough space is available, the ADD operation completes.
- If the NAUTO bit is set to 0 (TNETX15AE managing table), the TNETX15AE deletes as many addresses as necessary to fit the new address into the table by deleting the oldest addresses, using the AGE function.

interrupts available to support both wire ADDs and host ADDs

The following interrupts are active, regardless of the state of NAUTO, to support monitoring of addresses added to the table (subject to the bandwidth available to look up source addresses):

- NEW and NEWM interrupts are indicated when a new address is found.
- CHNG and CHNGM interrupts are indicated when the address is not new but the port assignment is changed.
- SECVIO and SECVIOM interrupts are indicated when the address is not new, the port assignment is changed, and the address is secured.

The following modes indicate control and other options that affect the ADD logic operation.

PRINCIPLES OF OPERATION

NAUTO mode

NAUTO mode is selected by asserting the NAUTO bit in the control register. In NAUTO mode, the ADD logic operation does not add addresses off the wire. The only way addresses are added, moved, or edited is through the register interface. The host processor is assumed to have complete control of the contents of the address table.

NLRN0 mode

The ADD logic operation cannot add addresses learned from port 0 when this bit is set. The bus watcher does not extract these addresses from the DRAM bus. In this mode, the management CPU can add an address for port 0 (the hardware does not). Since the bus watcher does not provide addresses from these ports to ADD in this mode, ADD does not perform any age process on any addresses in the lookup table assigned to port 0 in this mode; nor, are the NEW, CHNG, or SECVIO interrupts active for port 0 in this mode.

NCRC mode

The NCRC bit controls whether or not the bus watcher waits for a complete valid (CRC checked) frame before the source address is given to ADD. The TNETX15AE performs additions faster in NCRC mode since it does not have to wait for the good CRC indication on the bus. If the TNETX15AE is waiting for a good CRC, after a bad packet on a seldom-used port, no addresses are being added on any port, although, lookup is functioning normally for destination addresses already in the table. If this bit is set, there is a possibility that addresses from bad (bad or missing CRC) frames are added, but the aging process deletes them eventually.

management address adding

The ADD logic operation also can add addresses to the table through the DIO interface management add/edit address interface registers as shown in the following:

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
			Adddelcontrol	0x2C
≈				≈
Addnode23–Addnode16	Addnode31–Addnode24	Addnode39–Addnode32	Addnode47–Addnode40	0x38
Addport		Addnode7–Addnode0	Addnode15–Addnode8	0x3C

Management ADDs are used to perform the following functions:

- The address flags SECURE, LOCKED, and the copy uplink flag, CUPLINK, can be set or cleared through management adds.
- DIO ADDs are used to change the address port assignment.
- DIO ADDs also are the only way multicast addresses are added to the lookup table.
- DIO ADDs also write the current time to the AGE time stamp for the node.

Management add commands are given through the ADD bit in the adddelcontrol register. The steps for adding an address are as follows:

- Write the node's address in the addnode registers.
- If it is a unicast address, write the node's flag information and port assignment in addport. If it is a multicast address, write the forwarding mask.
- Set the ADD bit in the adddelcontrol register to 1.

PRINCIPLES OF OPERATION

management address adding (continued)

The ADD logic operation locks the addnode, addVLANID, and addport registers to ensure that they do not change during the address ADD. Reads to these registers are still possible, but writes are ignored. The ADD bit in the adddelcontrol register remains a 1 until the ADD is complete.

Having a sticky bit (remains a 1) for ADD gives the programmer the opportunity to set up or perform other register operations without having to wait for the ADD completion. A polling method is used to determine if the ADD is finished.

adding unicast and multicast addresses

There is no significant change in procedure between adding unicast and multicast addresses. There is one difference that the programmer must observe. The TNETX15AE stores different information for multicast addresses than for unicast addresses. Unicast addresses use a 4-bit code for the port number and three flag bits. Multicast addresses store a 15-bit forwarding mask.

Both data formats are added through the addport register. The format for this register changes, depending on bit 40 in the addnode register. If set to 1, the addport data is interpreted as multicast (if not, it is interpreted as unicast).

deleting an address

The TNETX15AE has two ways to delete addresses from the lookup table – the internal aging algorithm and a host request through the DIO interface. The DEL state machine is responsible for deleting addresses from the lookup table. DEL takes its information from the DIO registers for DIO deletes and from the AGE logic operation for aging deletes.

The TNETX15AE implements a 16-bit timer, incrementing every second for the aging process. This timer is used to write the time stamp during ADDs and for comparing ages.

aging (AGE logic operation)

The AGE logic operation is responsible for automatic address deletes. The TNETX15AE implements two styles of aging: time-threshold aging and table-full aging. The aging style is selected through the agingtimer register. A value of 0x0000 or 0xFFFF in the agingtimer register selects table-full aging. Any other value selects time-threshold aging. The AGE logic operation is disabled when the TNETX15AE is placed in NAUTO mode. The aging algorithm works as shown in Figure 16 and is described in the following.

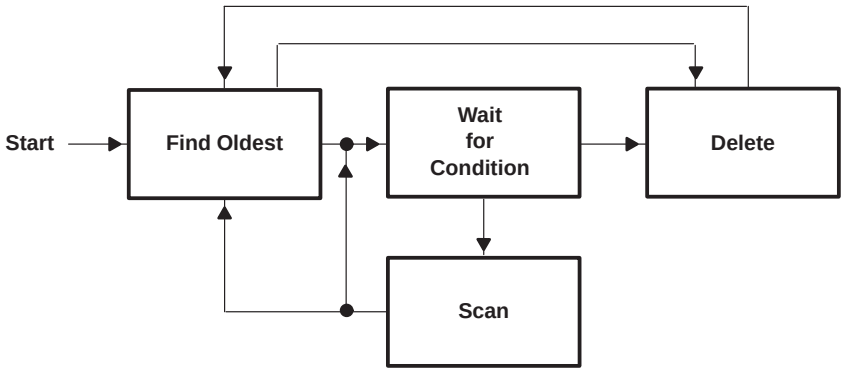


Figure 16. Aging Algorithm Block Diagram

PRINCIPLES OF OPERATION

aging (AGE logic operation) (continued)

- The operation AGE scans the table for the oldest address (state = find oldest state). AGE determines the oldest address by finding the address in the lookup table with the lowest time stamp. If more than one address has the same oldest time stamp, AGE picks the first address. The AGE scanning process omits all multicast addresses. This process also omits unicast addresses that have been secured by setting the SECURE flag. These addresses can be deleted only by a DIO-delete command.
- Once the oldest address is found, AGE keeps this address, enters a waiting state (state = wait for condition), until one of two conditions occurs:

1. If the address table is changed by either the ADD logic operation performing an address addition/time-stamp update or by deleting (DEL) an address, AGE checks the address that ADD or DEL is working on. If ADD has changed the time stamp for the current oldest address, it must scan the whole table for a new oldest address (state = find oldest). If DEL has deleted the current oldest address, it again must rescan the whole table for a new oldest address (state = find oldest). If neither ADD nor DEL touched the current oldest address, it still remains the oldest address and AGE returns to the wait state (state = wait for condition).
2. The aging condition is met. In this case, AGE calls on the DEL logic operation to delete the node from the table. After a successful deletion, AGE rescans the whole table for the next node to age (state = find oldest) and then gives an interrupt to the host to indicate the previous oldest address was deleted from the table.

- During the find-oldest process, and if AGE is in a time-threshold age, AGE deletes all addresses that are over the threshold value and reports them via the interrupt.

The aging condition is different for time-threshold aging and table-full aging, and both are described in the following paragraphs.

time-threshold aging

In time-threshold aging, the aging condition occurs when the address age is larger than the time threshold entered in the agingtimer (0x2) registers. The address age is not the time stamp written in the SRAM but the difference between current time and each time stamp. When this value becomes greater than agingtimer value, the address is deleted.

Example:

Time threshold value	= 192d seconds (0x00C0)
Current timer value	= 256d seconds (0x0100)
Address time-stamp value	= 80d seconds (0x0050)
Address aging time	= 256d – 80d = 176d seconds (0x00B0)

Conclusion: The address is not aged yet since the address aging time (176d seconds) is less than the time threshold value (192d seconds) by 16d seconds. It takes an additional 16d seconds (0x0010) for the address aging time to equal the time threshold (192d seconds) (0x00C0) and age (delete) the address.

PRINCIPLES OF OPERATION

table-full aging

Table-full aging is implemented for applications that do not want to use aging based on time, but still require aging. As its name implies, aging in this mode only happens when the lookup table is full and needs additional room to add a new address. The ADD logic operation initiates an aging request when it determines that it does not have enough tables to add the address it is working on. The time behaves differently in this mode. In table-full aging, the age timer does not increment every second, but rather when a new address is added. There must be a time difference between addresses to decide which is oldest, but a single count is sufficient. Since ADD time stamps every time it sees an address come through the bus, nodes that are actively transmitting between adds quickly move up to the same (new) age level. Those nodes that do not transmit remain at the lower age stamps. It is these nodes that are deleted in table-full aging when the table is full.

DEL logic operation (management address deletions)

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
			Adddelcontrol	0x2C
Delnode23–Delnode16	Delnode31–Delnode24	Delnode39–Delnode32	Delnode47–Delnode40	0x48
		Delnode7–Delnode0	Delnode15–Delnode8	0x4C

The DEL logic operation is controlled through the delnode and the adddelcontrol register. Management delete commands are given through the DEL bit in the adddelcontrol register.

The steps for deleting a particular address (multicast or unicast) are as follows:

- Writing the node address in the delnode register
- Asserting the DEL bit in adddelcontrol

The DEL logic operation locks the delnode on all delete commands to ensure that they do not change during the deletions. Reads to these registers are still possible. The DEL bit in adddelcontrol remains in the 1 state until the deletion is complete.

Much like the management adds, having a sticky bit for DEL gives the programmer the opportunity to set up or perform other register operations without having to wait for the delete completion. A polling method is used to find out if the delete is finished. This involves reading adddelcontrol to determine if the command bit used has returned to 0.

interrupts

The TNETX15AE implements interrupts to ease the management processor's tasks. The interrupts are used to indicate changes to the lookup table. It indicates that a new address is added, an address changed ports, an address changed ports and is secure, and an address is deleted due to the aging process. The statistic registers are half full (and the possibility for an overflow is present) and the lookup table is full. The TNETX15AE indicates that interrupts to the CPU exist by setting to 1 its level-sensitive EINT terminal. The EINT terminal is asserted when any of the possible interrupt conditions are met and the condition(s) is(are) enabled.

The interrupt register is readable at all times and contains all the current TNETX15AE interrupts. The interrupt register is byte clearable. That is, the interrupt bits in a byte are cleared when that byte is read, whether the int is masked or not.

PRINCIPLES OF OPERATION

masking interrupts

The TNETX15AE can mask interrupts. This is accomplished by an interrupt masking register (intmask). The int and intmask registers have a one-to-one correspondence. The only way EINT is asserted is if both the corresponding bits in the int and intmask are in a 1 state. The logic for the interrupt masking is shown in Figure 17.

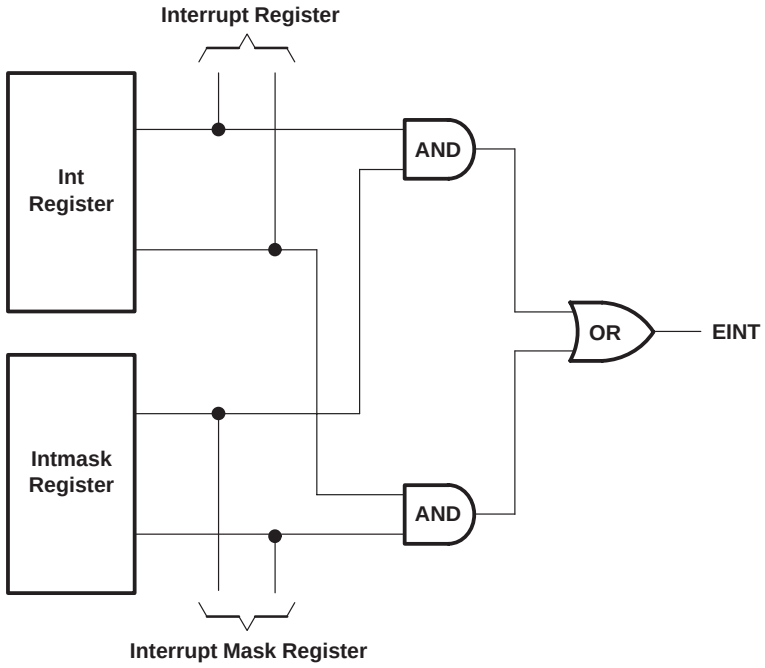


Figure 17. Interrupt Masking Logic

test interrupts (INT)

Test interrupts are generated by asserting the INT bit in the int register. The INT bit in the intmask register also must be set to a 1 state for the interrupt to take effect. The INT bit is used to give the programmer an easy way to test interrupt detection. This bit is the only bit in the int register that is writeable.

ADD interrupts (NEW, NEWM, CHNG, CHNGM, SECVIO, SECVIOM, and FULL)

The ADD interrupts generate collateral information in the following registers:

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Newnode23–Newnode16	Newnode31–Newnode24	Newnode39–Newnode32	Newnode47–Newnode40	0x30
Newport		Newnode7–Newnode0	Newnode15–Newnode8	0x34

ADD interrupts are sourced by the ADD logic operation based on information from the wire. ADD indicates a new address by a NEW interrupt, an address is on a new port by a CHNG interrupt, and a security violation by a SECVIO interrupt.

The FULL interrupt indicates that ADD needed to start AGE (to free up table space) if NAUTO is 0, or the host needs to delete some addresses if NAUTO = 1.

The FULL interrupt indicated that table resources are exhausted to the point where (wire or DIO) ADD cannot take place. If NAUTO = 0, AGE will delete oldest entries when another ADD is requested. IF NAUTO = 1, the host must do some deletes before the next DIO ADD can complete.

PRINCIPLES OF OPERATION

ADD interrupts (NEW, NEWM, CHNG, CHNGM, SECVIO, SECVIOM, and FULL) (continued)

The add interrupts are indicated in the interrupt register and the information for the particular interrupt is placed in the newnode and newport registers. Since there is only one set of registers that is shared for these interrupts, and to ensure that the information placed in these registers is not corrupted during reads, ADD locks the newnode and newport registers until the most-significant byte of newport is read.

Locking these registers means that ADD does not have a place to put information for new events. If additional events are missed, they are indicated in the int register as missed interrupts (NEWM, CHNGM, and SECVIOM).

On a NEW interrupt, the newport register contains information about the port for the new address. On a CHNG interrupt, this register identifies the new port address. On a SECVIO interrupt, the address does not move to the port, but the newport register indicates the port to which it tried to move.

aging interrupts (AGE, AGEM)

When an address is aged out by the internal AGE logic, it generates an interrupt and reports about that address in the following registers:

BYTE 3	BYTE 2	BYTE 1	BYTE 0	DIO ADDRESS
Agednode23–Agednode16	Agednode31–Agednode24	Agednode39–Agednode32	Agednode47–Agednode40	0x40
	Agedport	Agednode7–Agednode0	Agednode15–Agednode8	0x44

AGE indicates an interrupt every time that it ages out a node. It places the information on the node (being aged out) on the agednode register, the node's port on agedport. These registers are locked when a new interrupt is given to protect the information contained.

Missed interrupts due to these registers being locked are indicated as an AGEM interrupt. These registers are unlocked when the agedport register is read.

statistic interrupt (STAT)

The statistic interrupt is given when one of the statistic registers (except for numnodes) becomes one-half full; the most-significant bit becomes a 1. This is an indication to the management CPU that the statistic registers should be read to avoid counter overrun. Reading the most-significant byte of a statistic counter, except for numnodes, clears it. The host should read least-significant byte, then most-significant byte, and be prepared for a carry between byte reads.

If a carry took place between byte needs, the statistic count will appear to have jumped by 255 instead of one count.

IEEE Std 1149.1 test-access port (JTAG)

The test-access port consists of five terminals that are used to interface serially with the TNETX15AE package for boundary-scan testing.

The TNETX15AE is fully JTAG compliant, with the exception of requiring external pullup resistors on the following terminals: TDI, TMS, and $\overline{\text{TRST}}$.

The following instructions, with their 4-bit opcodes, are supported by JTAG.

INSTRUCTION TYPE	NAME	JTAG OPCODE
Mandatory	EXTEST	0000
Mandatory	SAMPLE/PRELOAD	0001
Private	ATPG	0010
Private	SELFEXERCISE	0011
Optional	IDCODE	0100
Optional	HIGH Z	0101
Private	IDDQ	1100
Mandatory	BYPASS	1111

The IDCODE for the TNETX15AE is:

	VARIANT	PART NUMBER	MANUFACTURER	LSB
Bit number	31–28	27–12	11–1	0
Binary code	0000b	1011 0001 1010 1110b	000 0001 0111b	1b

absolute maximum ratings over operating case temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} (see Notes 2 and 3) . . . . . –0.5 V to 3.6 V
Supply voltage range, $V_{CC(5V)}$ (see Notes 2 and 3) . . . . . –0.5 V to 5.5 V
Input voltage range, V_I . . . . . –0.5 V to $V_{CC(5\ V)}$ + 0.5 V
Output voltage range, V_O . . . . . –0.5 V to V_{CC}
Thermal impedance, junction-to-ambient package, airflow = 0, $Z_{\theta JA}$ . . . . . 47.5°C/W
Thermal impedance, junction-to-ambient package, airflow = 100 ft/min, $Z_{\theta JA}$ . . . . . 38.2°C/W
Thermal impedance, junction-to-case package, $Z_{\theta JC}$ . . . . . 9.9°C/W
Operating case temperature range, T_C . . . . . 0°C to 95°C
Storage temperature range, T_{stg} . . . . . –65°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” can cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods can affect device reliability.

NOTES: 2. All voltage values are with respect to GND.

3. Turning power supplies on and off (cycling sequence) within a mixed 5-V/3.3-V system is an important consideration. The designer must observe a few rules to avoid damaging the TNETX15AE. Check with the manufacturers of all components used in the 3.3-V to 5-V interface to ensure that no unique device characteristics exist that would lead to rules more restrictive than the TNETX15AE requires.

The optimum solution to power-supply sequencing in a mixed-voltage system is to ramp up the 3.3-V supply first. A power-on reset component operating from this supply forces all 5-V tolerant outputs into the high-impedance state. Then, the 5-V supply is ramped up. On power down, the 5-V rail deenergizes first, followed by the 3.3-V rail.

The second-best solution is to ramp both the 3.3-V and 5-V rails at the same time, making sure that no more than 3.6 V exists between these two rails during the ramp up or down. If the 3.3 V is derived from the 5 V, then the 3.3 V rises as the 5 V rises so that the 5-V rail never exceeds the 3.3-V rail by more than 3.6 V. Both the optimum and second-choice algorithms for power up prevent any device damage. If it is impractical to implement ramping, follow these rules:

- When turning on the power supply, all 3.3-V and 5-V supplies should start ramping from 0 V and reach 95 percent of their end-point values within 25 ms. All bus contention between the device and external devices is eliminated by the end of 25 ms.
- When turning off the power supply, 3.5-V and 5-V supplies should start ramping from steady-state values and reach 5 percent of these values within 25 ms. All bus contention between the device and external devices is eliminated by the end of 25 ms. There is a 250-second lifetime maximum at greater than 3.6 V between the supply rails. Holding this period to 25 ms per power-on/off cycle should not significantly contribute to shifts in mean time between failures (MTBF) during product lifetimes.

recommended operating conditions

		MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage	3	3.3	3.6	V
$V_{CC(5V)}$	Supply voltage	4.5	5	5.5	V
V_{IH}	High-level input voltage	2		$V_{CC(5V)}$	V
V_{IL}	Low-level input voltage (see Note 4)	0		0.8	V
I_{OH}	High-level output current			–4	mA
I_{OL}	Low-level output current			4	mA

NOTE 4: The algebraic convention, where the more-negative (less-positive) limit is designated as a minimum, is used for logic-voltage levels only.

electrical characteristics over recommended operating conditions (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage	I_{OH} = rated	V_{CC}–0.6			V
V_{OL}	Low-level output voltage	I_{OL} = rated			0.4	V
I_{OZ}	High-impedance-state output current	$V_O = V_{CC}$			20	µA
		V_O = 0			–20	
I_{IH}	High-level input current	$V_I = V_{I(MAX)}$			–20	µA
I_{IL}	Low-level input current	V_I = GND			20	µA
I_{CC}	Supply current, 3.3 V				75	mA
$I_{CC(5V)}$	Supply current, 5 V				2	mA

timing requirements over recommended operating conditions (see Figure 18)

external SRAM read cycle

NO.			MIN	MAX	UNIT
1	$t_{c(R)}$	Cycle time, read		40	ns
2	$t_{su(ED)}$	Setup time, ED15–ED0 valid before OSCIN↑	10		ns
3	$t_{h(ED)}$	Hold time, ED15–ED0 valid after OSCIN↑	2		ns

operating characteristics over recommended operating conditions (see Figure 18)

external SRAM read cycle

NO.	PARAMETER		MIN	MAX	UNIT
4	$t_{d(EA)1}$	Delay time, from OSCIN↑ to EA20–EA0 valid	2	17	ns
5	$t_{d(EOE)1}$	Delay time, from OSCIN↓ to $\overline{EOE}$↓		13	ns
6	$t_{d(EOE)2}$	Delay time, from OSCIN↑ to $\overline{EOE}$↑	2	12	ns
7	$t_{d(EA)2}$	Delay time, from $\overline{EOE}$↑ to EA20–EA0 valid	–2		ns
4+2	$t_{d(EA)1} + t_{su(ED)}$	Delay time, total for same device		24	ns

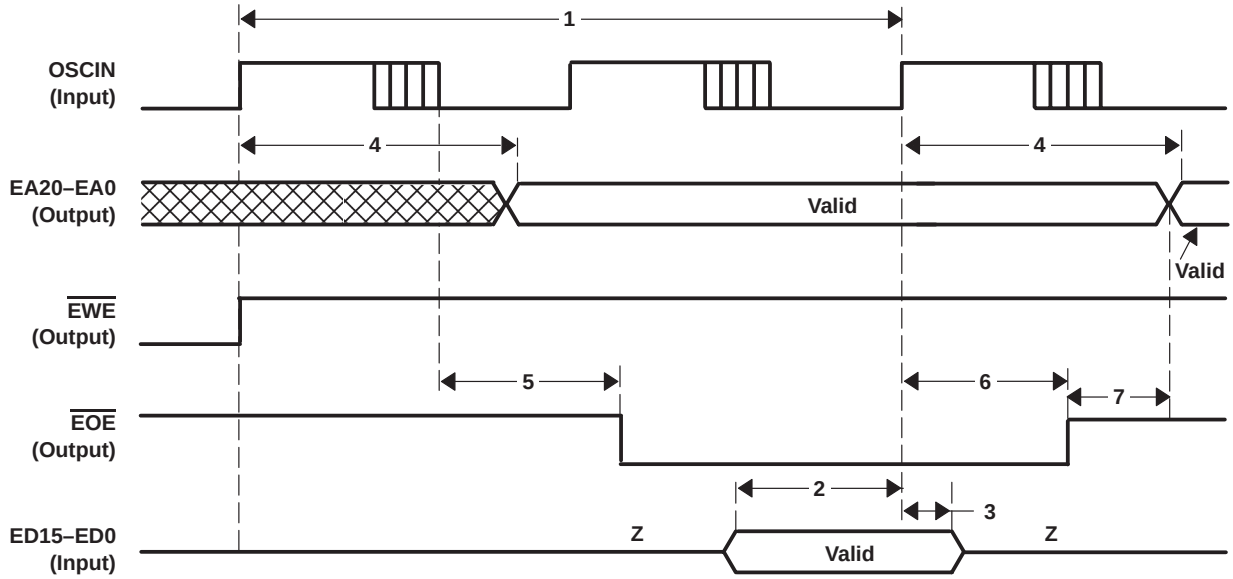


Figure 18. External SRAM Read Cycle

timing requirements (see Figure 19)

external SRAM write cycle

NO.			MIN	MAX	UNIT
1	$t_{c(W)}$	Cycle time, write		40	ns

operating characteristics over recommended operating conditions(see Figure 19)

external SRAM write cycle

NO.	PARAMETER		MIN	MAX	UNIT
2	$t_{d(EA)1}$	Delay time, from OSCIN↑ to EA20–EA0 valid	2	17	ns
3	$t_{d(EWE)1}$	Delay time, from OSCIN↓ to $\overline{EWE}$↓		12	ns
4	$t_{d(ED)1}$	Delay time, from OSCIN↓ to ED15–ED0 valid		17	ns
5	$t_{d(EWE)2}$	Delay time, from OSCIN↑ to $\overline{EWE}$↑	2	12	ns
6	$t_{d(ED)2}$	Delay time, from $\overline{EWE}$↑ to ED15–ED0 invalid	0		ns
7	$t_{d(EA)2}$	Delay time, from $\overline{EWE}$↑ to EA20–EA0 invalid	0		ns
8	$t_{d(ED)3}$	Delay time, from OSCIN↑ to ED15–ED0 invalid		17	ns

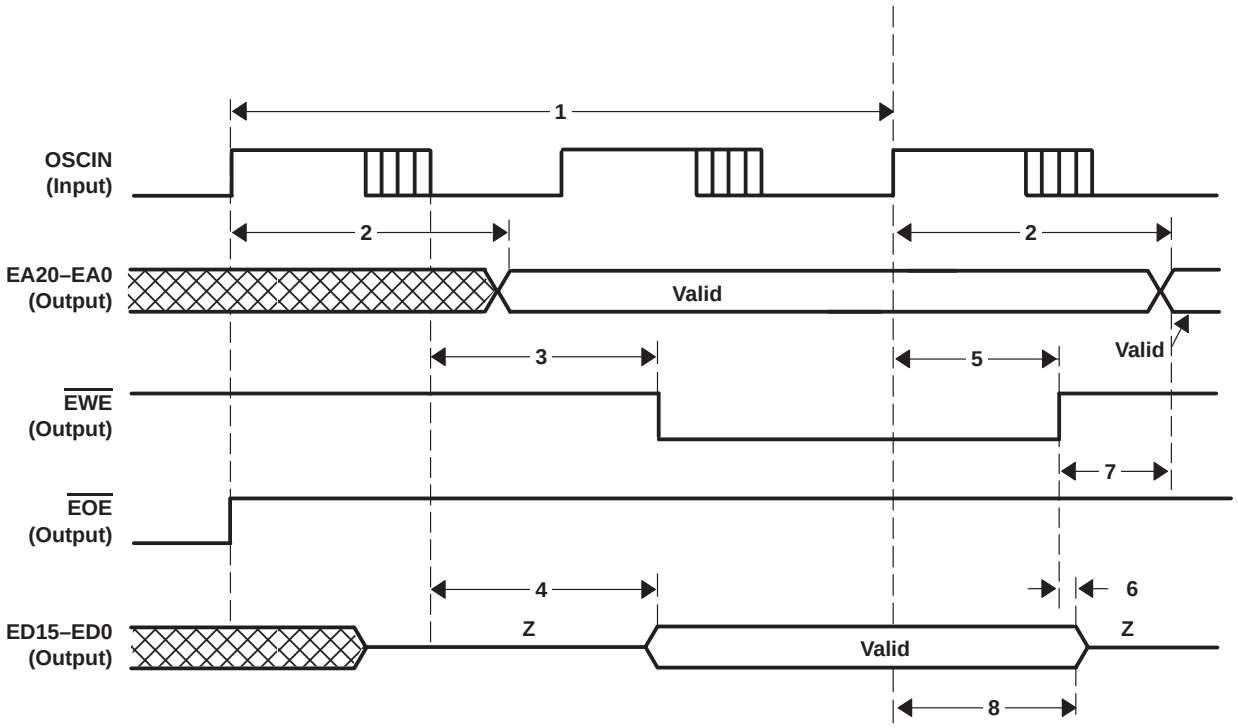


Figure 19. External SRAM Write Cycle

operating characteristics over recommended operating conditions (see Note 5 and Figure 20)

EAM routing code

NO.	PARAMETER		MIN	MAX	UNIT
1	$t_{d(EAM)1}$	Delay time, from EAM15–EAM0 valid to EAM15–EAM0 invalid	100		ns
2	$t_{d(EAM)2}$	Delay time, from OSCIN↑ to EAM15–EAM0 valid		17	ns

NOTE 5: The TNETX3150/TNETX3150A/TNETX3100 latches the EAM data (at certain edges) on frames that write the first 64 bytes of frame data to the DRAM. The edges where EAM is valid are shown in Figure 20. At other times, the value in EAM is not valid and is ignored by the TNETX3150/TNETX3150A/TNETX3100. The TNETX15AE outputs the EAM data on the rising edge of OSCIN before the TNETX3150/TNETX3150A/TNETX3100 latches the data and holds the data until the end of the transfer burst. This ensures adequate setup and hold times for TNETX3150/TNETX3150A/TNETX3100s EAM input. The particular frame, where the EAM code is valid, is identified by the start-of-frame (SOF) indicator in the frames flag fields and by the $\overline{DWE}$ strobe being low. The EAM code is not valid on DRAM reads [$\overline{DWE}$ = (1)] on refresh cycles or on IOB buffer writes and reads.

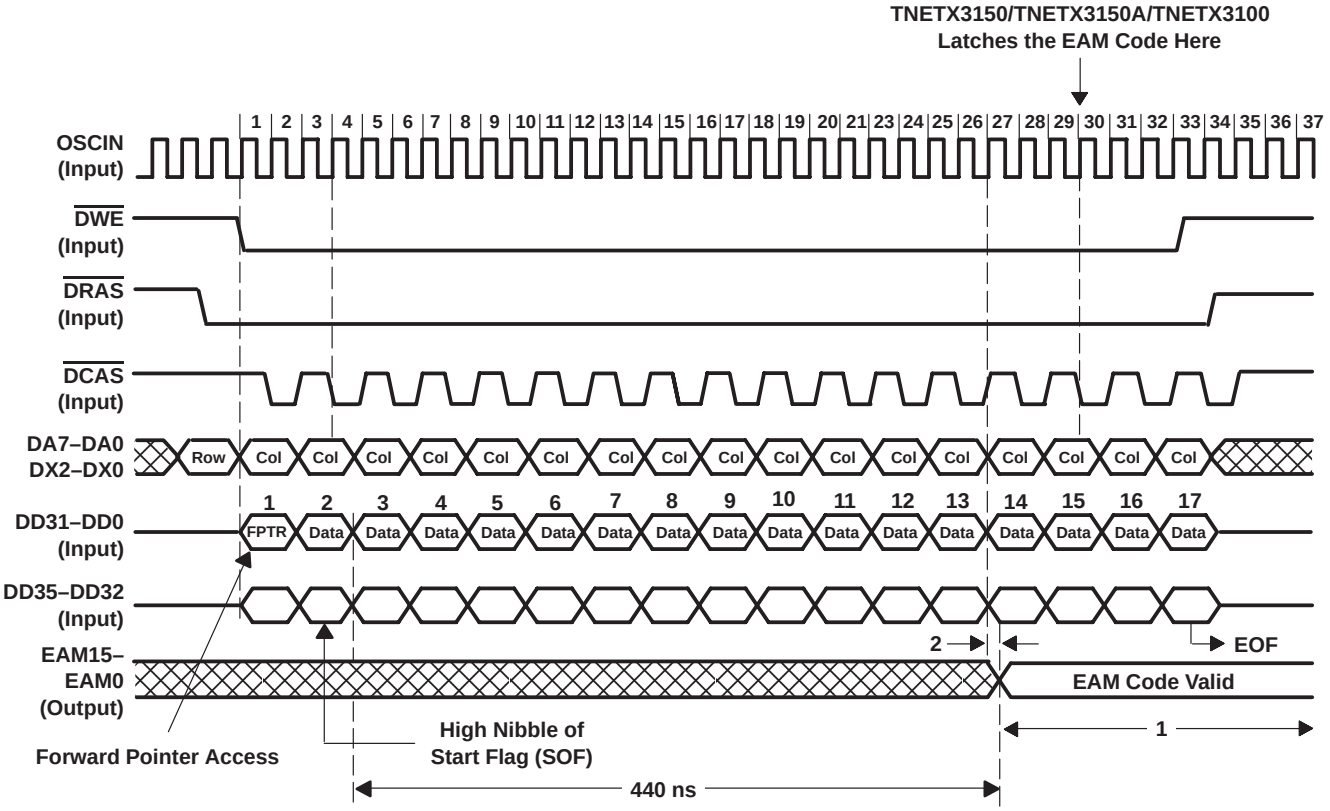


Figure 20. EAM Bus Timing

timing requirements (see Note 6 and Figure 21)

DRAM interface†

NO.			MIN	MAX	UNIT
1	$t_{su(DD)}$	Setup time, DD35–DD0 valid before OSCIN↑	3		ns
2	$t_{su(DWE)}$	Setup time, from $\overline{DWE}$↓ to OSCIN↑	4		ns
3	$t_{su(DRAS)}$	Setup time, from $\overline{DRAS}$↓ to OSCIN↑	4		ns
4	$t_{su(DCAS)}$	Setup time, from $\overline{DCAS}$↓ to OSCIN↑	5		ns
5	$t_{h(DD)}$	Hold time, DD35–DD0 valid after OSCIN↑	3		ns
6	$t_{h(DWE)}$	Hold time, $\overline{DWE}$ low after OSCIN↑	3		ns
7	$t_{h(DRAS)}$	Hold time, $\overline{DRAS}$ low after OSCIN↑	3		ns
8	$t_{h(DCAS)}$	Hold time, $\overline{DCAS}$ low after OSCIN↑	3		ns

† DD35–DD0 data is available for two cycles

NOTE 6: The TNETX15AE listens on the DRAM interface to extract frame information. The TNETX15AE extracts information on only the write cycles. The TNETX15AE does not output any signals to the DRAM interface. The only parameters that apply to the TNETX15AE on this interface are delay times, with respect to OSCIN, that translate into the TNETX15AE setup times.

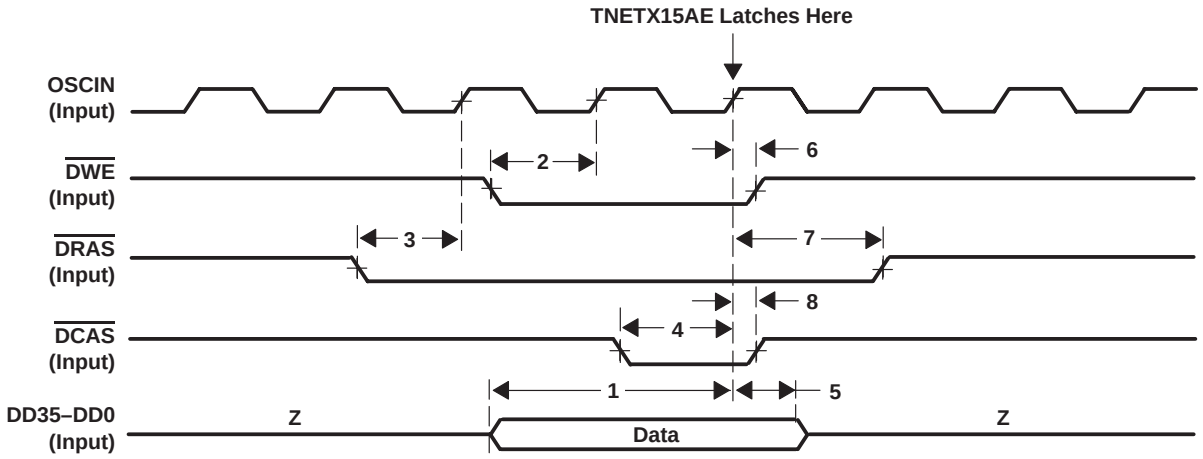


Figure 21. DRAM Interface Timing

timing requirements (see Note 7 and Figure 21)

DIO read cycle

NO.			MIN	MAX	UNIT
1	$t_{w(ESCSL)}$	Pulse duration, $\overline{ESCS}$ low	40		ns
2	$t_{su(SRNW)}$	Setup time, SRNW high before $\overline{ESCS}\downarrow$	3		ns
3	$t_{su(SAD)}$	Setup time, SAD1–SAD0 valid before $\overline{ESCS}\downarrow$	3		ns

NOTE 7: The DIO interface is a byte-wide, asynchronous interface that is designed for use with a wide variety of CPUs. The interface can be run at any speed; there is no minimum speed requirement. DIO cycle lengths can vary, depending on the type of register accessed. The TNETX15AE withholds the $\overline{SRDY}$ acknowledge signal on registers until it has gathered the necessary data.

operating characteristics over recommended operating conditions (see Note 7 and Figure 22)

DIO read cycle

NO.	PARAMETER		MIN	MAX	UNIT
4†	$t_{w(SRDYH)}$	Pulse duration, $\overline{SRDY}$ high		25	ns
5	$t_{d(SRNW)}$	Delay time, from $\overline{SRDY}\downarrow$ to SRNW$\downarrow$	0		ns
6	$t_{d(SAD)}$	Delay time, from $\overline{SRDY}\downarrow$ to SAD1–SAD0 invalid	0		ns
7	$t_{d(SDATA)1}$	Delay time, from $\overline{SRDY}\downarrow$ to SDATA7–SDATA0 valid	5		ns
8	$t_{d(ESCS)}$	Delay time, from $\overline{SRDY}\downarrow$ to $\overline{ESCS}\uparrow$	0		ns
9	$t_{d(SDATA)2}$	Delay time, from $\overline{ESCS}\uparrow$ to SDATA7–SDATA0 high-impedance Z		17	ns
10†	$t_{d(SRDY)1}$	Delay time, from $\overline{ESCS}\downarrow$ to $\overline{SRDY}\downarrow$	40	‡	ns
11†	$t_{d(SRDY)2}$	Delay time, from $\overline{ESCS}\uparrow$ to $\overline{SRDY}\uparrow$	0	25	ns

† These parameters are specified by design, but not tested.

‡ Max = min + N (internal cycles), where N equals the number of internal cycles required to arbitrate internally to obtain the value being requested. N is equal to a minimum of 1, and can be large during RAM INIT and FINDNEXT operation, since DIO operations will not complete during these operations (no $\overline{SRDY}$ true), and both of these operations can read every SRAM location to finish.

NOTE 7: The DIO interface is a byte-wide, asynchronous interface that is designed for use with a wide variety of CPUs. The interface can be run at any speed; there is no minimum speed requirement. DIO cycle lengths can vary, depending on the type of register accessed. The TNETX15AE withholds the $\overline{SRDY}$ acknowledge signal on DIO accesses until it has gathered the necessary data.

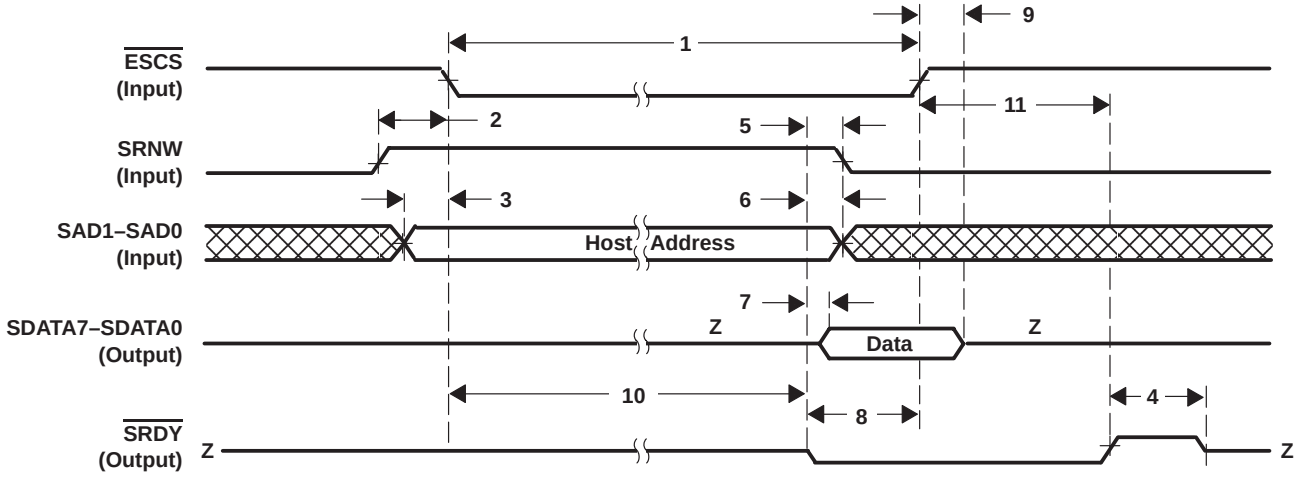


Figure 22. DIO Read Cycle

timing requirements (see Note 7 and Figure 23)

DIO write cycle

NO.			MIN	MAX	UNIT
1	$t_{w(ESCSL)}$	Pulse duration, $\overline{ESCS}$ low	40		ns
2	$t_{su(SRNW)}$	Setup time, SRNW low before $\overline{ESCS}\downarrow$	3		ns
3	$t_{su(SAD)}$	Setup time, SAD1–SAD0 valid before $\overline{ESCS}\downarrow$	3		ns
4	$t_{su(SDATA)}$	Setup time, SDATA7–SDATA0 valid before $\overline{ESCS}\downarrow$	3		ns

NOTE 7: The DIO interface is a byte-wide, asynchronous interface that is designed for use with a wide variety of CPUs. The interface can be run at any speed; there is no minimum speed requirement. DIO cycle lengths can vary, depending on the type of register accessed. The TNETX15AE withholds the $\overline{SRDY}$ acknowledge signal on registers until it has gathered the necessary data.

operating characteristics over recommended operating conditions (see Note 7 and Figure 23)

DIO write cycle

NO.	PARAMETER		MIN	MAX	UNIT
5†	$t_{w(SRDYH)}$	Pulse duration, $\overline{SRDY}$ high		25	ns
6	$t_{d(SRNW)}$	Delay time, from $\overline{SRDY}\downarrow$ to SRNW$\uparrow$	0		ns
7	$t_{d(SAD)}$	Delay time, from $\overline{SRDY}\downarrow$ to SAD1–SAD0 invalid	0		ns
8	$t_{d(SDATA)}$	Delay time, from $\overline{SRDY}\downarrow$ to SDATA7–SDATA0 high-impedance Z	0		ns
9	$t_{d(ESCS)}$	Delay time, from $\overline{SRDY}\downarrow$ to $\overline{ESCS}\uparrow$	0		ns
10†	$t_{d(SRDY)1}$	Delay time, from $\overline{ESCS}\downarrow$ to $\overline{SRDY}\downarrow$	40	‡	ns
11†	$t_{d(SRDY)2}$	Delay time, from $\overline{ESCS}\uparrow$ to $\overline{SRDY}\uparrow$	0	25	ns

† These parameters are specified by design, but not tested.

‡ Max = min + N (internal cycles), where N equals the number of internal cycles required to arbitrate internally to obtain the value being requested. N is equal to a minimum of 1, and can be large during RAM INIT and FINDNEXT operation, since DIO operations will not complete during these operations (no $\overline{SRDY}$ true), and both of these operations can read every SRAM location to finish.

NOTE 7: The DIO interface is a byte-wide, asynchronous interface that is designed for use with a wide variety of CPUs. The interface can be run at any speed; there is no minimum speed requirement. DIO cycle lengths can vary, depending on the type of register accessed. The TNETX15AE withholds the $\overline{SRDY}$ acknowledge signal on registers until it has gathered the necessary data.

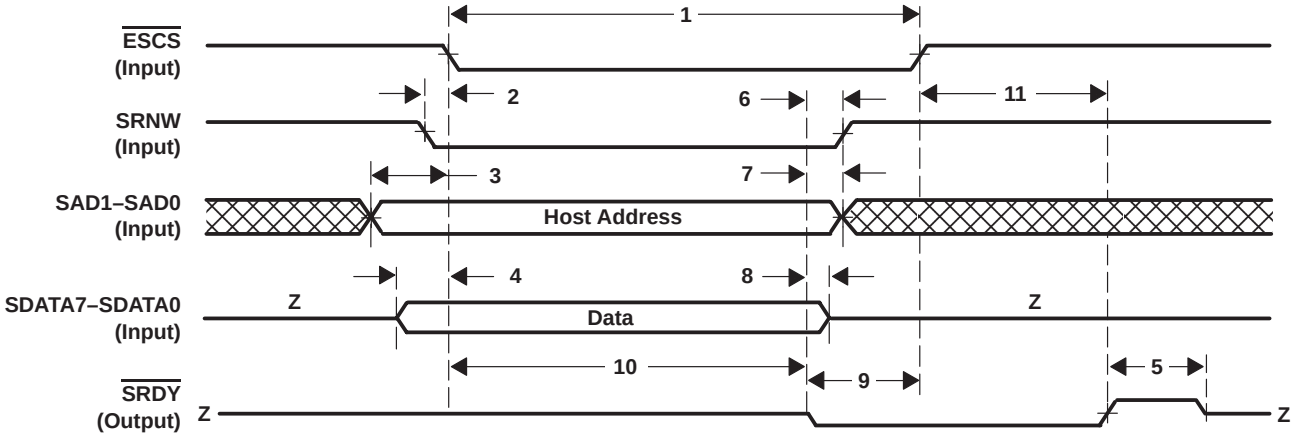


Figure 23. DIO Write Cycle

timing requirements over recommended operating conditions

OSCIN clock

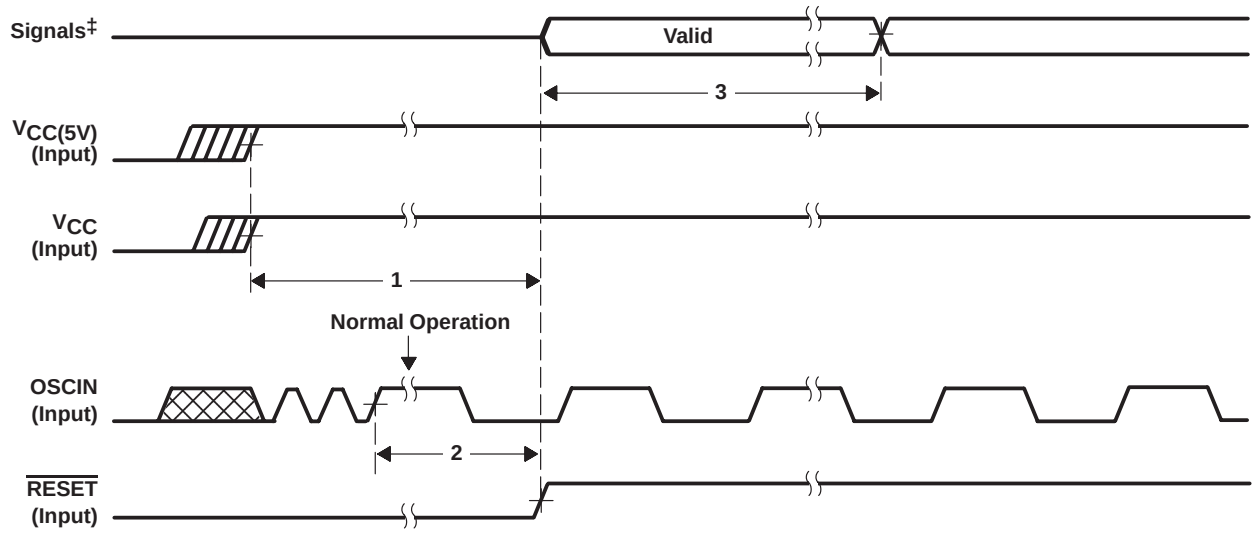
		MIN	NOM	MAX	UNIT
$t_{c(OSCIN)}$†	Cycle time, OSCIN		20		ns
$t_{w(OSCINH)}$	Pulse duration, OSCIN high	8	10		ns
$t_{w(OSCINL)}$	Pulse duration, OSCIN low	8	10		ns
	Frequency drift†			±50	ppm

† TNETX15AE is designed to use the OSCIN input from the companion TNETX3150/TNETX3150A/TNETX3100 to be able to snoop on the TNETX3150/TNETX3150A/TNETX3100-DRAM bus. The accuracy and drift are constrained by TNETX3150/TNETX3150A/TNETX3100 use of OSCIN for Ethernet state machines.

timing requirements over recommended operating conditions (see Figure 24)

power-on reset

NO.			MIN	MAX	UNIT
1	$t_{d(OSCIN)}$	Delay time, from V_{CC}↑ to $\overline{RESET}$↑	25		ms
2	$t_{d(RESET)}$	Delay time, from OSCIN↑ to $\overline{RESET}$↑	25		ms
3	$t_{d(AUTO)}$	Delay time, from $\overline{RESET}$↑ to EEPROM autoload over		50	ms



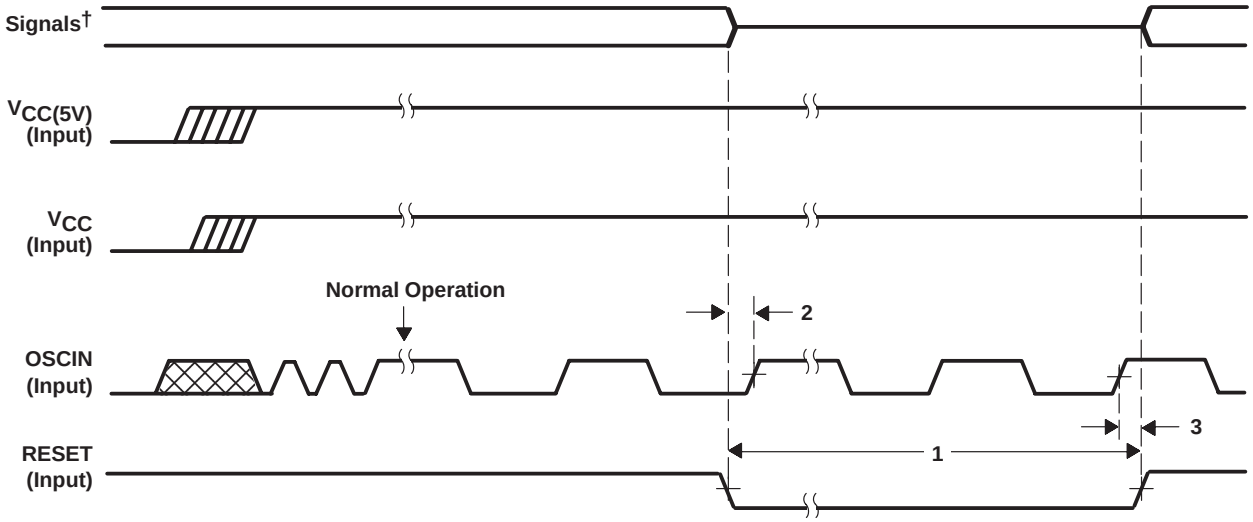
‡ A pullup or active logic is required for each input to have known logic levels at all times, especially during reset when an I/O output is disabled.

Figure 24. Power-On Reset Timing

timing requirements over recommended operating conditions (see Figure 25)

RESET (software) timing

NO.			MIN	MAX	UNIT
1	$t_{w(RESET)}$	Pulse duration, RESET low	3		µs
2	$t_{su(RESET)}$	Setup time, RESET low before OSCIN↑		10	ns
3	$t_{h(RESET)}$	Hold time, RESET low after OSCIN↑		12	ns



† A pullup or active logic is required for each input to have known logic levels at all times, especially during reset when an I/O output is disabled.

Figure 25. RESET (Software) Timing

operating characteristics over recommended operating conditions (see Note 8 and Figure 26)

EEPROM and MDIO (MII) interfaces

When the host is driving the EEPROM interface through the SIO register (0x0A), the output terminals are copies of register bits synchronized by OSCIN rising, and the input register bits are copies of the device terminals synchronized by OSCIN rising. This adds a maximum of 20 ns of delay in each direction, which is negligible in terms of the maximum signal change rate allowed for the EEPROMs (approximately 100 kHz), or the MDIO (MII management). The timing for these interfaces is set at the rate that register bits are set in software. For detailed descriptions of the EEPROM requirements, refer to the data sheet for the part in question. For detailed descriptions of the MII management interface, refer to IEEE Std 802.3.

When the TNETX15VE is driving the EEPROM during the automatic register load cycle, the timing on the EEPROM is as shown in Figure 26.

EEPROM interface timing (TNETX15AE to EEPROM transfer) (see Note 8)

NO.	PARAMETER†		MIN	MAX	UNIT
	f_{clock}	Clock frequency, ECLK		98	kHz
1	$t_{w(ECLKL)1}$	Pulse duration, ECLK low during start/stop	10.24		µs
2	$t_{w(ECLKH)1}$	Pulse duration, ECLK high during start/stop	10.2		µs
3	$t_{w(ECLKH)2}$	Pulse duration, ECLK high during data	5.1		µs
4	$t_{w(ECLKL)2}$	Pulse duration, ECLK low during data	5.12		µs
5	$t_{d(EDIO)1}$	Delay time, from ECLK↑ to EDIO↓, start condition	5.1		µs
6	$t_{d(ECLK)1}$	Delay time, from EDIO↓ to ECLK↓, start condition	5.1		µs
7	$t_{d(EDIO)2}$	Delay time, from ECLK↓ to EDIO↑, data condition	0		µs
8	$t_{d(ECLK)2}$	Delay time, from EDIO↑ to ECLK↑, data condition	5.1		µs
9	$t_{d(EDIO)3}$	Delay time, from OSCIN to EDIO		18	ns
10	$t_{d(ECLK)3}$	Delay time, from OSCIN to ECLK		18	ns

† These parameters are specified by design, but not tested.
NOTE 8: The timing for the EEPROM interface occurs with the ETEST bit set to 0.

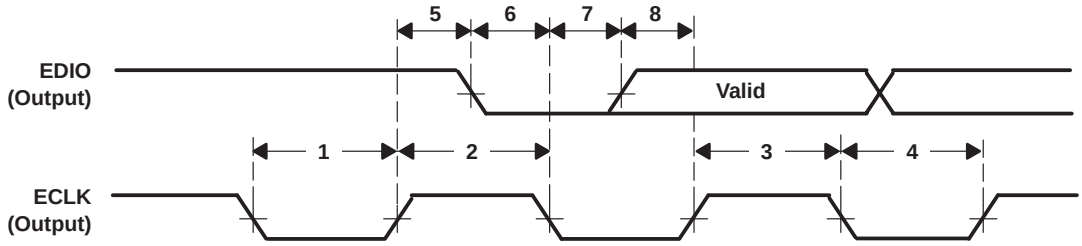


Figure 26. EEPROM Interface Timing

PARAMETER MEASUREMENT INFORMATION

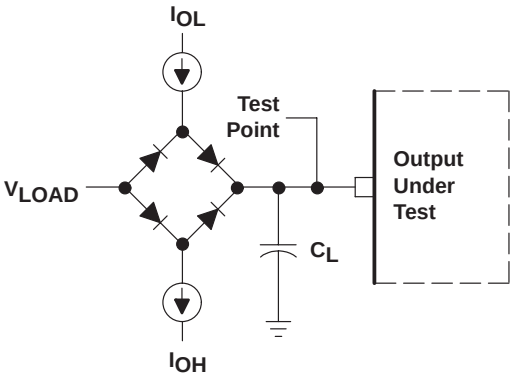
Outputs are driven to a minimum high-logic level of 2.4 V and to a maximum low-logic level of 0.6 V. These levels are compatible with TTL devices.

Output transition times are specified as follows: all transition times are measured at the point where the output signal crosses 1.3 V.

The rise and fall times are not specified but are assumed to be those of standard TTL devices, which are typically 1.5 ns.

test measurement

The test-load circuit shown in Figure 27 represents the programmable load of the tester pin electronics that is used to verify timing parameters of the TNETX15VE output signals.



OUTPUT TEST LOAD

Where:

I_{OH} = refer to I_{OH} in recommended operating conditions
I_{OL} = refer to I_{OL} in recommended operating conditions
V_{LOAD} = 1.5 V, typical
C_L = 25 pF, typical load-circuit capacitance

Figure 27. Test and Load Circuit

APPLICATION INFORMATION

TNETX15AE/TNETX3150/TNETX3150A stand-alone applications

unmanaged switch

The simplest application for the TNETX15AE device is shown in Figure 28. This is an unmanaged multiport switch. The TNETX15AE is responsible for matching addresses, learning addresses, and for eliminating (aging out) old addresses. The TNETX15AE also provides options to the manufacturer through its EEPROM. The manufacturer could program this EEPROM through a parallel-port interface to the TNETX15AE. Options that can be set are SRAM size, the aging time, and where unknown unicast and multicast frames go. This is the lowest-cost solution for an unmanaged multinode-per-port switch. This is the architecture of the TI evaluation module (EVM).

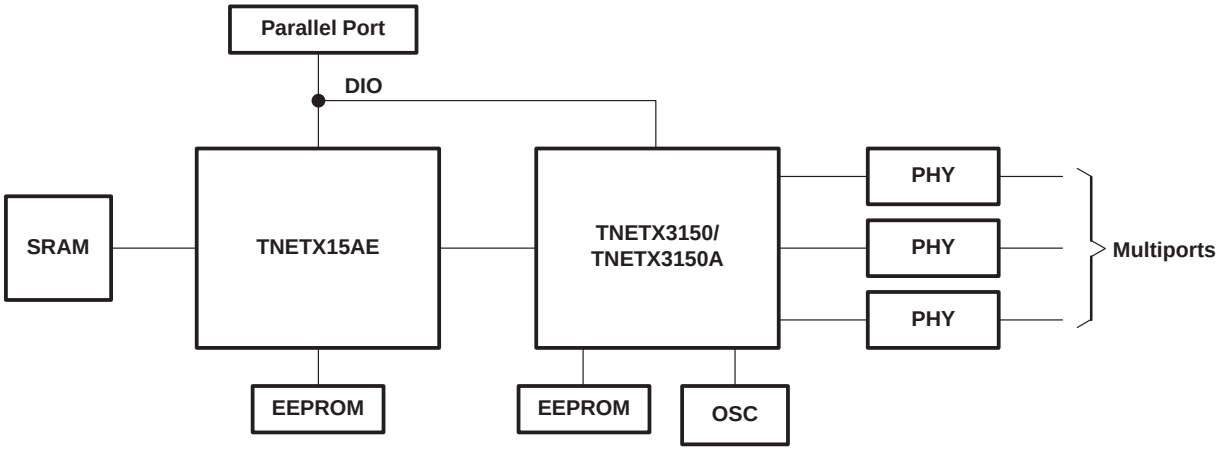


Figure 28. Lowest Cost Unmanaged Multiport Switch

managed switch

The microprocessor (CPU) interfaces to the TNETX3150/TNETX3150A/TNETX3100 through a common DIO interface. The microprocessor also can manage any PHY through an IEEE Std 802.3u MII interface using the SIO register in the TNETX15AE.

The microprocessor's tasks are minimized mainly because the CPU does not have to participate in frame matching. The microprocessor is used to set TNETX15AE/TNETX3150/TNETX3150A modes, secure addresses so that the node does not move ports (useful for routers, attached switches, and servers), and respond to control MIB requests.

The TNETX15AE is designed for easy management of the lookup table. Address-table lookups, adds, edits, and deletes are performed easily through the TNETX15AE registers. Interrupt support in the TNETX15AE also simplifies the management tasks. The TNETX15AE can give an interrupt to the CPU when the lookup table changes due to a wire event. This minimizes code, as the CPU does not have to actively poll a large address table for changes.

In-band signaling (see Figure 29) must be done with a MAC device connected to one of the switch ports. There is no access to the data stream via the DIO port on the TNETX3150/TNETX3150A/TNETX3100 or the TNETX15AE. Both devices affect the ability of a port to receive or forward packets; the packet itself is received and transmitted via a regular switch port.

APPLICATION INFORMATION

TNETX15AE/TNETX3150/TNETX3150A stand-alone applications (continued)

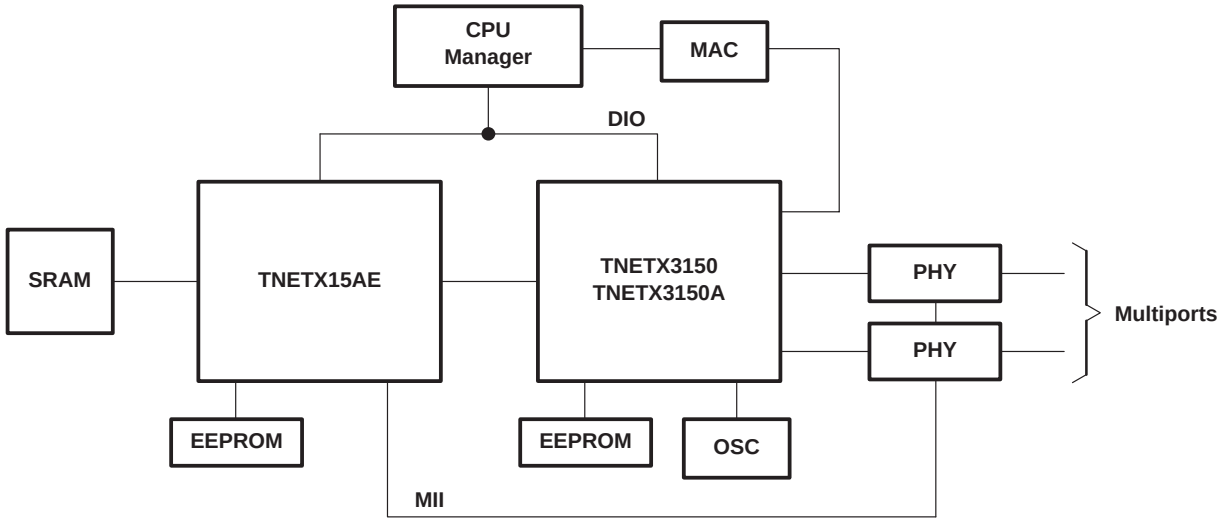
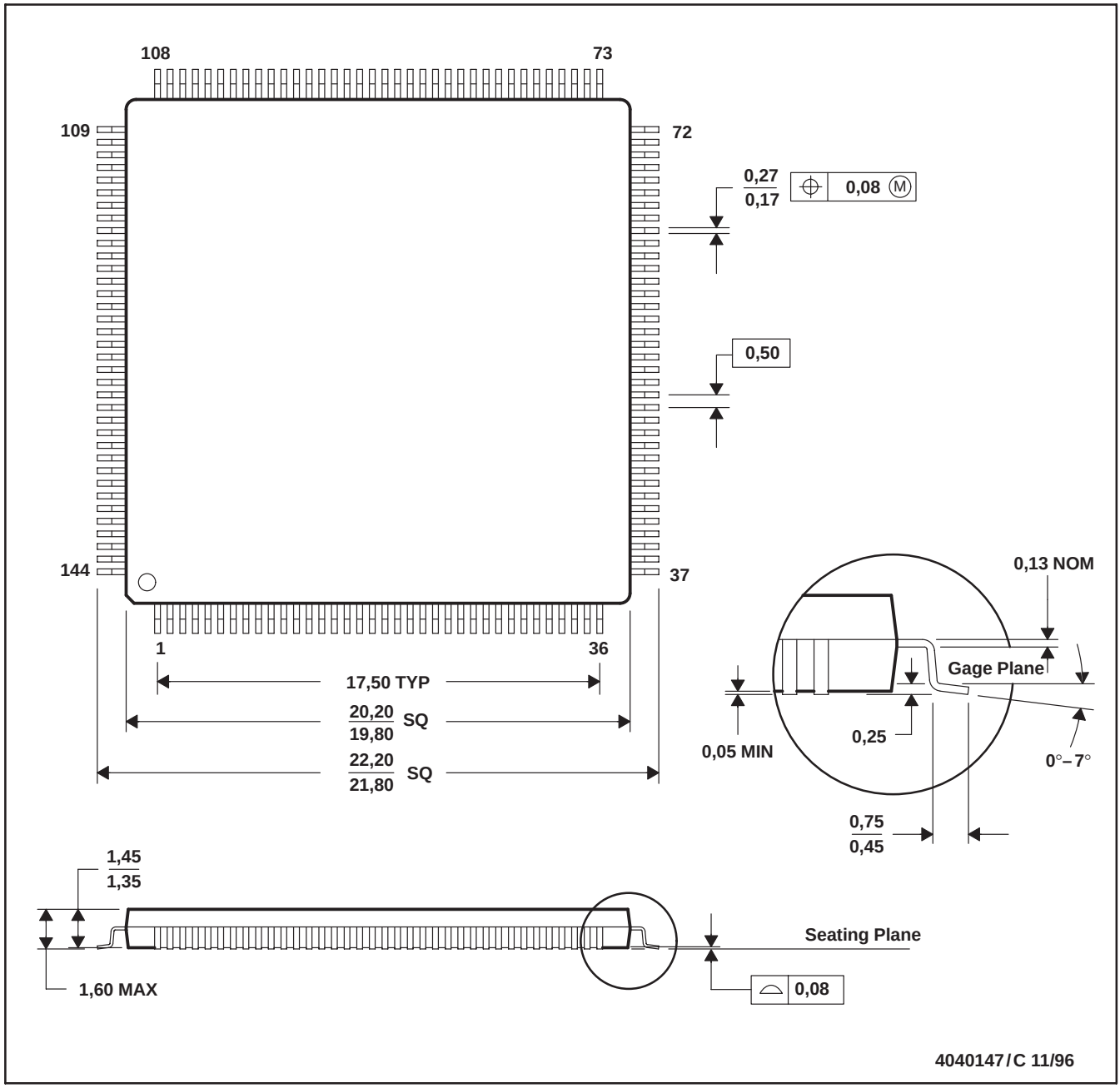


Figure 29. In-Band Managed Multiport Switch

With access to a port, the CPU can receive frames destined to other nodes. By tagging the CUPLNK bit for that particular address in the address table, the CUPLNK bit copies all frames destined for that address to the ports specified in UPLINKport. By setting UPLINKport to direct these frames to the management CPU, it can receive frames of interest.

MECHANICAL DATA

PGE (S-PQFP-G144) **PLASTIC QUAD FLATPACK**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS-026

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.