**Ethernet Switch**

**Features:**
- Fully Integrated Ethernet Switch On A Chip
- Twenty four 10Base-X Ports, Two 10/100Base-X Ports
- All Ports can be either Full or Half Duplex
- Low Per-Port Cost
- 1.28 Gbps Bus Bandwidth
- High Capacity Rx and Tx FIFOs and Tx Queue Which Absorb Traffic Bursts to Maximize Throughput
- Scaleable Shared Memory Packet Buffer
- Store & Forward and Fragment Free Cut-Thru Switching
- Flow Control with Programmable Backpressure
- Broadcast/Multicast Isolation using VLAN
- Any-to-any Port Mirroring on all Ports
- Standard MDIO Interfaces for Manageable PHYs
- MII Interface on Both 10/100 Ports
- High Density Chip Design
- Low Voltage 3.3V Design
- 5V-tolerant I/Os
- 352-pin Taped Grid Array Package

**General Description:**
The CXD1700 is a fully integrated single-chip Ethernet/Fast Ethernet switch for a wide range of low cost high performance switched network applications. A powerful high density, low cost-per-port desktop, workgroup, or departmental LAN Layer 2 switch can be built around the CXD1700 on a single circuit board, with a minimal complement of external components. All ports can operate in either full or half duplex, with full flow control, switching packets either in store & forward or fragment free cut-thru mode.

In a minimum configuration design, the CXD1700 can be implemented into a low cost high bandwidth desktop switch, using an EEPROM startup data source and an SDRAM packet buffer. With a CPU, a scaleable SDRAM buffer, industry standard hardwired or MDIO manageable PHYs, and management statistics database storage logic, the CXD1700 can be integrated into a powerful yet very cost effective fully managed Layer 2 switch. With its 24 full or half duplex 10Base-X and two full or half duplex 10/100Base-X ports, its high capacity receive and transmit data FIFOs and transmit packet queues, and a 40 MHz 32-bit switching bus, the CXD1700 has plenty of power for sustained high throughput and traffic bursts. The CXD1700 can age over 10,000 MAC addresses and the nodes can be distributed among the ports in any combination. It has programmable internal backpressure, which triggers a choice of advanced flow control actions. CXD1700 has VLAN capability, port mirroring, and it supports the spanning tree algorithm. It has per-port management statistics probes, that generate vectors to satisfy all popular MIBs. Network management can be implemented via SNMP, a web server, or other means.

Contact your Sony Representative for reference designs.

Sony reserves the right to change products and specifications without prior notice. This information does not convey any license by any implication or otherwise under any patents or other right. Application circuits shown, if any, are typical examples illustrating the operation of the devices. Sony cannot assume responsibility for any problems arising out of the use of these circuits. The purchase of products is subject to availability and Sony's standard terms and conditions of sale.

© 1997 Sony Semiconductor Company of America. Sony and pacITman are trademarks of Sony.
Summary of Functions and Capabilities

- 24 10Base-X ports, each with an integrated media access unit (MAC) and serial network interface (SNI) interface to industry standard physical layer devices (PHYs).
- Two 10/100Base-X ports each has a MAC and full MII interface, including a 2-line MDIO interface, through which all standard manageable PHY functions, including autonegotiation and half/full-duplex selection, can be programmed. Any or all 26 PHYs connected to CXD1700 can be controlled through these two MDIO interfaces.
- Each 10 Mbps port has a 128-byte receive FIFO, a 256-byte transmit FIFO, and a transmit queue for 128 packet pointers. The 100 Mbps ports have 512 byte receive and transmit FIFOs and a 128 packet transmit queue.
- Packets are stored in per-port packet buffers in external SDRAM (scaleable from 4 to 16 Mbits). Packet buffer size and other memory mapping is user programmable.
- High and low water marks can be programmed in each transmit queue, to control when backpressure is initiated and released. In half duplex mode flow control (collision) can be programmed to start on current incoming packet, or after current packet for a programmable duration. In full duplex, IEEE 802.3x pause frames are used; pause duration is user programmable.
- Packet reception is always fragment free (minimum of 64 bytes before forwarding); packet length can exceed 1536.
- Switching bus is 32 bits wide, operated by a 40 MHz clock providing 1.28 Gbps bandwidth. The bus is extended off chip to SDRAM and is accessible by the host CPU (through bus bridge logic).
- Switching engine, ports, and other internal functions use the same 40 MHz clock.
- Learning tables have capacity for more than 10,000 MAC addresses and the addresses can be distributed among ports in any combination. Lookup uses a fast linking algorithm for destination port identification.
- Three learning tables are rotated at user controlled intervals.
- All CXD1700 read/write operations are done as burst mode transfers for maximum efficiency.
- SDRAM refresh is done by the CXD1700.
- Host CPU can gain direct access to SDRAM with bus request/bus grant lines to CXD1700.
- VLAN support for multicast/broadcast packets; Up to 1024 VLAN maps can be mapped and dynamically updated.
- Full spanning tree algorithm support, with programmable learning and forwarding registers.
- Each port has a comprehensive management statistics probe. Synchronous output of raw management statistics data to user’s counter/interface logic. Counter data can be used to construct MIB variables for most standard MIBs.

Complement of External Devices

To implement a full Layer 2 Ethernet switch, the following components are required with the CXD1700:

- A device controller. In a full featured, remotely or locally managed switch this will be a host CPU, but in a low end switch the CXD1700 can be booted and aged by an FPGA from an EEPROM source, provided network management support is not required.
- SDRAM for storing packets, learning tables, and VLAN maps.
- Bus bridge logic to facilitate CPU access to the SDRAM and for buffering packets to/from the CPU
- PHYs for connecting the CXD1700 ports to physical media.
- Optional counter logic (with generic CPU interface) for storage and forwarding of management statistics.

Sony will provide specifications for two reference designs. The first is a full featured and fully manageable Layer 2 Ethernet switch, with all of the CXD1700 capabilities implemented. This design utilizes all components listed above. It has 24 10Base-X ports, two 10/100Base-X ports; all ports can be full or half duplex, with autonegotiation and other MDIO manageable PHY features.

The second design is a low cost minimum configuration Layer 2 Ethernet switch, operated from an EEPROM. It has 24 10 Mbit ports and two 10/100 Mbit ports, but the PHYs are hardwired to power up in either half duplex or full duplex mode. This low cost design implements a fixed aging period for the learning tables, does not support VLAN, spanning tree algorithm, and has no provision for network management of the device, or storage of management statistics (does not have the management statistics counter logic).
# TABLE OF CONTENTS

## 1.0 CXD1700 Architecture

1.1 Implementing the CXD1700 .................................................. 5
1.2 Packet Switching Mode ....................................................... 5
1.3 Switching Bus and Switch Engine ........................................ 5
1.3.1 Lookup/Learning Logic, Learning Tables ....................... 6
1.3.2 Memory Control ........................................................... 7
1.3.3 Mailbox Controller/CPU Port Control .............................. 7
1.4 Ports ............................................................................. 7
1.4.1 Port Numbering ............................................................ 8
1.4.2 Per-Port Packet Buffers ................................................ 8
1.4.3 Port Circuits and Registers ........................................... 8
1.5 CXD1700 Registers .......................................................... 9
1.5.1 Switch Engine Control/Status Registers ....................... 9
1.5.2 Port Control Registers .................................................. 9
1.5.3 MAC Control Registers ............................................... 9
1.5.4 MII Control Registers ............................................... 10
1.6 Packet Header ................................................................ 10
1.7 Flow Control .................................................................. 11
1.8 VLAN ......................................................................... 12
1.9 Port Mirroring ............................................................... 12
1.10 Management Statistics Probe ......................................... 13

## 2.0 External Components and Interfaces

2.1 SDRAM .......................................................................... 14
2.2 Host CPU ....................................................................... 15
2.3 Bus Bridge Logic ............................................................. 16
2.4 Statistics Counter Logic ..................................................... 16
2.5 Physical Layer Device (PHY) Interface ............................ 16
2.5.1 Nonmanageable PHY Interface ................................ 16
2.5.2 Manageable PHY Interface ......................................... 17

## 3.0 Functional Description

3.1 Startup and Initialization ..................................................... 17
3.2 Packet Forwarding Decision Protocol ............................... 17
3.3 How Frames are Switched Thru the CXD1700 .................... 18
3.3.1 Store & Forward Mode Transfer ................................... 18
3.3.2 Cut-Thru Mode Transfer ............................................. 19
3.3.3 Multicast/Broadcast Packets ...................................... 20
3.3.4 CPU Packets — Receive ............................................ 20
3.3.5 CPU Packets — Transmit ........................................... 21
3.4 Lookup, Learning, and Aging .......................................... 21
3.4.1 Learning Table Structure ......................................... 22
3.4.2 Learning and Aging ................................................... 22
3.4.3 Lookup ................................................................... 23
3.5 VLAN Implementation ..................................................... 23
3.6 Spanning Tree ............................................................... 24
3.7 Generation and Output of Management Statistics ............. 25

## 4.0 Registers/Programming Information

4.1 SDRAM Memory Mapping ............................................... 26
4.2 Registers ....................................................................... 26
4.3 Initialization Procedure ..................................................... 36
4.4 Reading/Writing PHY Registers ........................................ 38
4.5 Activating Full Duplex Operation ..................................... 38
4.6 Setting Up Flow Control ................................................... 38
4.7 Setting Up Collision Window .......................................... 39
4.8 Setting Up VLAN .......................................................... 39
4.9 Setting Up Mirroring ....................................................... 39
4.10 Setting Up Spanning Tree Facilities ................................. 39

## 5.0 Hardware Information

6.0 Timing Diagrams  
(Figures 6.1 through)
Figure 1.1 -- CXD1700 Full Page Block Diagram
1.0 CXD1700 Architecture

1.1 Implementing the CXD1700

The CXD1700 is a functionally highly integrated ASIC, built around a 32-bit 40 MHz switching bus, a switch engine, 24 full duplex 10Base-X ports, and two full duplex 10/100Base-X ports (refer to the block diagram of Figure 1.1). Constructing a single circuit board Ethernet/Fast Ethernet switch with the CXD1700 requires several external functions. First, a host CPU is required to initialize the CXD1700 and thereafter time the aging of MAC address learning tables. Depending on whether extended CXD1700 features are implemented, the host CPU must also execute a spanning tree algorithm, VLAN management, physical layer (PHY) management, as well as execute resident network management agent software. In a low cost switch, the initialization and MAC address aging can be executed with an FPGA instead of a CPU, as long as the other features listed above do not have to be implemented.

In every application a synchronous DRAM must be used to serve as a packet buffer, to store the MAC address learning (aging) tables, VLAN maps, and to provide a mailbox interface through which the host CPU communicates with the CXD1700. Bus bridge logic is required to interface a host CPU bus to the SDRAM and to CXD1700 external control lines. This bridge logic, in conjunction with the SDRAM, must also provide buffer storage for packets going to and from the CPU. A 40 MHz system clock must be provided to the CXD1700.

Industry standard 10Base-X and 10/100Base-X physical layer devices (PHYs) can be connected to CXD1700 ports. The PHYs can be hardwired to operate either in half or full duplex data communications mode, or they can be MDIO manageable, with commands from the host CPU via the MII interfaces on the two 10/100 Mbit ports. To implement management statistics storage and statistics vector processing by a network management subagent, external logic with a bank of counters must be provided. These counters must include a data input interface to the CXD1700 and a data output interface to the host CPU. All external components and their interfaces are discussed in more detail in Section 2.0 External Components and Interfaces.

1.2 Packet Switching Modes

When implemented with the above components, the CXD1700 can perform high speed multiple address-per-port Layer 2 packet switching on 26 ports, either in half or full duplex mode (selectable port-by-port), with full flow control. The CXD1700 can switch packets in two different modes. In store & forward mode an incoming packet is written into a packet buffer in the SDRAM, queued for transmitting, and then read out and transmitted at a later time. In cut-thru mode an incoming packet is routed directly from the incoming port to the outgoing (destination) port. All packets forwarded in the cut-thru mode are also stored in the SDRAM packet buffer.

Switching a packet across the internal CXD1700 switching bus is controlled by the switch engine and separate receive and transmit controllers in the ports. Finite state machines in the ports and in the switch engine synchronize and sequence switching bus operations. At different times packet data, packet identifying information (header), or source and destination port numbers move across the switching bus. The switch engine also executes external write or read operations in the SDRAM packet buffers. To maximize traffic handling depth and throughput and to minimize congestion during bursts, large FIFO buffers are used in the receive and transmit side of each port, the overall SDRAM size is scaleable and the size of individual packet buffers is adjustable.

1.3 Switching Bus and Switch Engine

The switching bus is central in the architecture of the CXD1700. It consists of 32 data lines, and a separate group of control, internal handshaking, and bus arbitration lines (none of the latter lines are shown in the simplified block diagram of Figure 1.1). The switching bus operates at 40 MHz and extends to all major functional blocks inside the CXD1700. The 32 data lines also extend outside the chip to the SDRAM and to the bus bridge logic, however none of the internal bus control lines go outside the CXD1700. The internal segment of the switching bus is used only by CXD1700 internal logic. The external segment is utilized both by the CXD1700 and by the host CPU. CXD1700 internal logic uses it for reading and writing in the SDRAM, and the CPU also reads and writes in the SDRAM, via its bridge logic. Because the CPU cannot access the internal bus, there is no direct data communication between the host CPU and CXD1700. Instead, a fixed mailbox location is defined in the SDRAM, where the CPU, through its bus bridge logic, can write commands to the CXD1700 and read responses from CXD1700. The CPU is also allowed read/write access to any other location in the SDRAM, including those reserved for ports and for switch engine operations. The internal bus is isolated from the external segment by line buffers.
The switch engine consists of four major functional blocks. The bus control logic arbitrates and prioritizes access to the switching bus by the other switch engine logic blocks, by all the ports, as well as by the host CPU. Even though the CPU cannot access the internal bus, the internal and external segments are handled as a single entity for bus grant arbitration purposes. While any bus operation is in progress on one segment of the bus, another operation cannot be initiated on the other segment. All bus operations are prioritized, so that in general, lookup and learning operations receive higher priority than packet receive and transmit. The 100 Mbit ports receive higher priority than 10 Mbit ports, but the CPU, via its BREQ line, has the lowest priority.

The lookup/learning logic queries the MAC address learning tables, to identify the destination port(s) for any packet received by CXD1700, and performs MAC address learning and aging functions. Memory control services SDRAM read and write requests from all other logic blocks in the CXD1700. Such requests can come from lookup/learning logic, the receive or transmit side of every port, or the mailbox controller and CPU port control.

The mailbox controller/CPU port control manages host CPU read/write access to internal registers of the CXD1700. It also facilitates packet transfers to and from the CPU. The host CPU needs to write or read internal registers throughout the CXD1700 for powerup initialization, for aging the learning tables, for implementing a spanning tree algorithm, or for other device maintenance reasons. To read or write internal registers, the host CPU deposits commands in the SDRAM mailbox, bus bridge logic asserts the CMDRDY line to the CXD1700 mailbox controller, and the command is fetched out of the mailbox by the mailbox controller. All responses to CPU commands are written into the mailbox by mailbox controller and the completion of a command is indicated via a TASKDONE line.

The host CPU can also receive or send network packets through the CXD1700 ports, in much the same manner as any other network node does. In fact, the mailbox controller/CPU port control logic, together with a FIFO type storage device in the bus bridge logic, simulate the essential functions of a network port. Incoming packets addressed to the CPU are processed by the receiving port and are stored in SDRAM packet buffers, exactly the same as network port-to-port packets. The CPU port control function of the switching engine issues a HDRSTRB and packet identifying data is gated by memory control onto the switching bus external extension, where it must be captured by bus bridge logic. Thereafter, the CPU reads the packet out of the packet buffer.

To send a packet out, the CPU gains access to the switching bus external extension with the BREQ and BGRT control lines. The CPU then stores the packet and a transmit command in SDRAM, and issues a CMDRDY to send the packet. This command is processed by the mailbox controller and the packet is queued in the destination port, read out of SDRAM, and transmitted the same as any other packet.

1.3.1. Lookup/Learning Logic and Learning (Aging) Tables. The lookup/learning logic in the switching engine maintains the learning tables and ages them under direction of the host CPU. During the first 12 bytes of every unicast packet arrival, the lookup/learning logic extracts the destination MAC address from the incoming packet and uses it to access the learning tables and identify the CXD1700 port on which the packet is to be transmitted. At the conclusion of lookup the receiving port and destination port numbers both appear on the switching bus. This appearance of the addresses is significant, because it is the first indication to the destination port that a packet is coming and allows it to instigate flow control action (see Section 1.7 below).

There are three parallel learning tables in the SDRAM, referred to as Learning Table 1, 2, and 3. Entries from the learning tables overflow into three other tables, identified as Link Table 1, 2, and 3. Each link table is functionally tied to a specific learning table, so that for the purposes of this discussion a learning table and its associated link table can be considered as one. All three aging/link tables are identical in their structure. Usage of the three tables is rotated under control of the host CPU, so that at any given moment lookup/learning logic accesses one table to get destination port ID numbers, does learning into that table and one other, while the third table is on standby, with all 0s written into it. For purposes of their instantaneous role in the rotation, the three tables are designated as Current, Next, and Empty, in that order. Lookup is done in the Current table, but learning is done into Current and Next. Whenever the aging command from the CPU is executed, the tables are rotated, so that Current becomes Empty, Next becomes Current, and Empty advances to Next. Thus, at any given time, for example Learning table 2 could be designated as the Current, Next, or Empty table. The rotation (aging) interval is nominally 5 minutes, but it is a programmable value, determined by the host CPU. The spanning tree algorithm also expands or contracts the aging interval, as needed for its purposes.

Learning is the opposite of lookup and puts new entries into the tables. Learning is also done for every packet received. To do learning, the lookup/learning logic extracts the source MAC address from the incoming packet and enters the Current learning table to see whether that MAC address and its associated port number are already in the learning table. If it is, nothing further is done; if it is not, the MAC address and the port number are recorded in the Current learning table. The learning process is then repeated a second time into the Next table.

To optimize memory space utilization, MAC addresses are written into the tables in linked lists, which extend from an learning table.
into its associated linking table. In this manner, in excess of 10,000 MAC addresses can be stored in an learning table and its linking table. The construction of the aging and linking tables and more details of the lookup and learning processes are described in Section 3.4, Lookup, Learning, and Aging.

Note that the learning tables are never used for multicast or broadcast packet destination port lookup; refer to VLAN description, Section 1.8

1.3.2 Memory Control. Memory control exists to service the SDRAM read and write requests from other functions within the CXD1700 and to execute SDRAM memory refresh operations. There are no other logic blocks within the CXD1700 that can read or write in the SDRAM, or, in fact, perform any operation on the external extension of the switching bus. To address, set up, and control the SDRAM, memory control has 16 dedicated address lines and 7 control lines that go outside the CXD1700 chip (see Figure 1.1) to the SDRAM. It uses the external extension of the 32 switching bus data lines to get the data to and from the SDRAM.

All SDRAM read and write operations are performed in burst mode. The requesting logic block places the memory starting address and the number or reads or writes to be performed onto the switching bus. Then, memory control uses its external lines to set up the burst parameters in the SDRAM mode register, supply the starting memory address, and sequences all control signals. It also turns on the corresponding line buffer to connect the internal switching bus to the external extension. As the burst mode operation starts, memory control prompts the requesting logic block to either place data onto the switching bus (for SDRAM writes) or fetch data off the bus (for reads).

1.3.3 Mailbox Controller/CPU Port Control. The mailbox controller/CPU port control has two functions. First, in conjunction with the mailbox in the SDRAM, the mailbox controller implements the host CPU read/write interface to all internal registers of the CXD1700. Second, the CPU port control circuits process packet transfers to and from the CPU. To implement register reading and writing and to transmit CPU packets, there are three 32-bit commands, as shown in Table 1.1.

<table>
<thead>
<tr>
<th>Command Type (Byte 3)</th>
<th>Port Number (Byte 2)</th>
<th>Starting Register (Byte 1)</th>
<th>Transfer Count (Byte 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Register  = 0x20</td>
<td>Netwk Ports = 0 - 25</td>
<td>Any Writeable Register (see Programming Section)</td>
<td>Number of sequential registers to be written</td>
</tr>
<tr>
<td>Read Register  = 0x40</td>
<td>Netwk Ports = 0 - 25</td>
<td>Any Readable Register (see Programming Section)</td>
<td>Number of sequential registers to be read</td>
</tr>
<tr>
<td>Transmit Packet  = 0x60</td>
<td>24-bit starting address of packet in SDRAM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The host CPU needs to write or read internal control and status registers throughout the CXD1700 during initialization and for aging the learning tables. The CPU also accesses internal registers to autonegotiation results in PHYs, for implementing a spanning tree algorithm, or for other local station maintenance reasons. To read or write internal registers, the host CPU deposits a command, along with all required data (for writes), in the SDRAM mailbox. Then, bus bridge logic issues a CMDRDY to the CXD1700 mailbox controller, and the mailbox controller requests memory control to fetch the command and data out of the mailbox. All responses to CPU commands are also written into the mailbox by memory control and the completion of a command is acknowledged by the mailbox controller when it asserts the TASKDONE line.

Note that the CPU can write or read multiple contiguous registers with one command by supplying the port number (see Section 1.4), the starting register address, and the number of registers to be accessed.

1.4 Ports
The CXD1700 has 24 FIFO buffered 10Base-X ports and two FIFO buffered 10/100Base-X ports. The 10Base-X ports can interface with any industry standard PHY device (either software manageable or hardwired) through a standard 7-line SNI. The 10/100Base-X ports can interface with all industry standard 10/100Base Ethernet PHYs, through a full MII serial management interface. The two standard MDIO lines of the MII interface can be daisychained to manageable 10 Mbps Ethernet PHYs on any of the CXD1700 ports. All ports can operate either in full or half duplex data communication mode. The 10Mbit ports can forward packets either in
cut-thru mode or in store & forward mode. The 10/100Mbit ports can operate only in store & forward. In each port, the receiving and transmitting functions can be turned on or off separately, by setting two bits in the port control register PCR0 (see below).

CXD1700 also has a host CPU port, with receive and transmit facilities constituted by various circuits within the switch engine, the SDRAM, and bus bridge logic.

1.4.1. Port Numbering. All CXD1700 ports are identified by logical designations as follows:

<table>
<thead>
<tr>
<th>Port Type</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Mbit ports</td>
<td>0 thru 23</td>
</tr>
<tr>
<td>10/100 Mbit ports</td>
<td>24 and 25</td>
</tr>
<tr>
<td>CPU port</td>
<td>26</td>
</tr>
</tbody>
</table>

These port designations are used for all packet switching operations, as well as by the host CPU for writing or reading port control registers, or when receiving or sending packets. These designations are also used throughout this document. Ports 0-25 are sometimes alternatively referred to as the network or physical ports. 26 is also called the host port. (Note that for purposes of CXD1700 internal communications protocol, the switch engine registers as a group are assigned the designation of port 27, but these registers do not have any physical attributes of a port, nor do they perform any port functions.)

1.4.2. Per-Port Packet Buffers. Each of the 26 network ports is assigned a block of memory in the SDRAM, where all packets received through that port are written. This is referred to as the packet buffer and there are 26 buffers, one corresponding to each network port. Every packet buffer is written into in a rotating manner — received packets are written starting with the first buffer address and continue until the buffer is full. After writing the last address, writing is restarted at the beginning and previously stored packets are overwritten. The CPU port does not have a dedicated packet buffer. Instead, incoming CPU packets are first stored in the packet buffer of the receiving port and later retrieved by the CPU. To transmit packets, the CPU generally stores them in the mailbox area in the SDRAM, but as described earlier, the CPU has access to all packet buffers, as well as other SDRAM locations.

1.4.3. Port Circuits and Registers. The 26 network ports consist of a separate receive side, transmit side, and an integrated media access controller (MAC). There is a separate management statistics probe in each port; it monitors the other three port functions for statistics gathering. The statistics probe is described separately, in Section 1.10, Management Statistics Probe.

Each network port has four 32-bit control and status registers, labeled as PCR0 through PCR3. These are programmed by the host CPU on powerup and can be reprogrammed at any time with a CPU register write command. Through the four port registers, the receive and transmit functions in any port can be individually turned on or off, cut-thru packet transfers can be enabled or disabled, various flow control attributes can be managed, and either full or half-duplex operation of the port can be selected. Port registers 2 and 3 store the starting and ending addresses of the packet buffer in the SDRAM. Each of the 26 ports also has eleven 32-bit control and status registers for the embedded MACs. These registers, designated MAC 0 thru MAC 8, LSA1, and LSA2, store parameters for implementing CSMA/CD and 802.3x flow control disciplines, selected error checking and statistics parameters, and miscellaneous CXD1700 internally used data.

Ports 24 and 25 (the 10/100 Mbps ports) each has five additional registers, MII 0 thru MII 4. These are used by the host CPU to manage 10/100 Mbps Ethernet PHYs on ports 24 and 25, via the two MDIO control lines. The MDIO lines can be daisychained to manageable 10 Mbps Ethernet PHYs on any CXD1700 ports.

The receive side of each port is constructed around a 32-bit wide receive FIFO (Rx FIFO) and a receive port controller. In the 10 Mbps ports the receive FIFO is 128 bytes (32x32) in size, but in the 100 Mbps ports it is 512 bytes (128x32). Incoming 8-bit wide packet data from the MAC is converted into 32-bit words before being clocked into the FIFO. The receive port controller, built around a finite state machine (FSM), executes a multistep packet receive sequence. In the process, the receive port controller loads and unloads the Rx FIFO and generates bus requests and lookup requests to the switch engine. It also manages and supplies current packet buffer addresses to memory control and gates packet data onto the switching bus. At the end of the packet, the receive port controller constructs and outputs a packet header that uniquely identifies the packet and lists any errors detected in it. It is on the basis of this packet header that the packet is recognized, reconciled, and queued in the transmit side of the destination port.

The transmit side of each port is constructed around a 32-bit wide transmit FIFO (Tx FIFO), a Tx queue, and a transmit port controller. In the 10 Mbps ports the transmit FIFO is 256 bytes (64x32) in size, but in the 100 Mbps ports it is 512 bytes (128x32).

The Tx queue is a 384-byte FIFO (128x24) in all ports. Like on the receive side, the transmit port controller is built around an FSM,
which sequences a multistep packet transmit sequence. It starts with the loading of the Tx queue and interacting with the bus control logic to gain access to the switching bus. To get packet data, the transmit port controller generates memory read requests and supplies packet starting addresses to memory control. It clocks packet data into the Tx FIFO, as the data is read out of the packet buffer. The Tx port controller also communicates with the MAC to process data to the MAC. In addition, the transmit port controller monitors congestion (with a user preprogrammed high-water mark for the Tx queue) and generates a backpressure signal to the port which is sending a packet. (This is described in more detail in Section 1.7 Flow Control)

1.5 CXD1700 Registers

The CXD1700 has some 144 32-bit internal registers that are variously utilized by the internal logic and/or are accessed by the host CPU. These registers are written to during powerup initialization to start up the CXD1700, to enable its ports, and to enter a wide variety of operating parameters. There are four general groups of registers. One set is in the switch engine, another set is in each of the 26 network ports, a separate set of registers exists for the integrated MACs, and ports 24 and 25 each has another set of registers for setting up and managing external PHYs connected via the MDIO lines. At power on (or following an external hardware RESET at device pin Y2) the CPU initializes registers throughout the CXD1700 by loading initial operating parameters. Among other things, the CXD1700 internal registers contain the SDRAM memory map, they are used to individually enable or disable each network port and define the network attributes of each port separately. Some of the CXD1700 internal registers can be written to and read by the CPU, others can only be written.

The Switch Engine Registers, or all Port Control Registers of any given port are at contiguous addresses. Therefore, the host CPU can write, for example, all Switch Engine Registers, or all Port Control Registers of a particular port, with one write command.

During normal operation, the host CPU does not typically need to access or write in any of the internal registers, except one -- the periodic learning table rotation command bit (in Switch Engine Register 7). Refer to Section 4.0 Registers/Programming Information for a full description of the contents of all registers and initialization requirements for the CXD1700.

1.5.1 Switch Engine Control and Status Registers. These registers deal with the core of CXD1700 operations, including the management and operation of the CXD1700 learning tables, VLAN operation, port mirroring, and spanning tree algorithm. They contain the SDRAM memory maps of all tables used in the above operations. The registers are physically located in the lookup/learning logic block.

| ECR0 | Control and Status (Learning Table Status, VLAN enable, Mirroring Enable) |
| ECR1 | Learning Table 1 and Link Table Addresses |
| ECR2 | Learning Table 2 and Link Table Addresses |
| ECR3 | Learning Table 3 and Link Table Addresses |
| ECR4 | Permanent Table and VLAN Table Addresses |
| ECR5 | Port Mirroring Setup |
| ECR6 | Learning Bitmap for Spanning Tree |
| ECR7 | Forwarding Bitmap for Spanning Tree |
| ECR8 | Learning Table Rotation Command |

1.5.2 Port Control Registers. Each of the 26 network ports of the CXD1700 has an identical set of four registers. These registers are used to enable receive and transmit functions in the port, select whether the port operates in half or full duplex mode, and enable or disable cut-thru operation. These registers also contain the values that determine at what congestion level backpressure will be activated on any given port (refer to Section 1.7). Port registers define the location and size of packet buffers in SDRAM for each port.

| PCR0 | Port Enable, Cut-thru Enable, Full/Half Duplex, Flow Control |
| PCR1 | Backpressure Control |
| PCR2 | Packet Buffer Start Address |
| PCR3 | Packet Buffer End Address |

1.5.3 MAC Control Registers. Each of the 26 network ports of the CXD1700 has an identical set of 10 registers. These registers deal with setting up the interface to the physical layer. such as CSMA/CD protocol implementation, 802.3x flow control, half or full duplex selection, and other related parameters.

| MACC | Main Control |
MAC1  Packet Length/IPG
MAC2  Flow Control Parameters
MAC3  Reserved (Do not write or read)
MAC4  Reserved (Do not write or read)
MAC5  Reserved (Do not write or read)
MAC6  Reserved (Do not write or read)
MAC7  Reserved (Do not write or read)
MAC8  Reserved (Do not write or read)
LSA1  Station Address, LSB (for 802.3x use)
LSA2  Station Address, MSB (for 802.3x use)

1.5.4 MII Control Registers (in 100 Mbit ports only). These registers are only in the 100 Mbps ports, 24 and 25. They are used entirely for reading and writing 802.3u registers in manageable physical layer devices (MII PHYs).

MCMD  MII Command
MADR  MII Device Address
MWTD  MII Write Data
MRRD  MII Read Data
MIND  MII Indicators

1.6 Packet Header
The CXD1700 uses an internal construct called a packet header. A packet header is attached to every packet received by the CXD1700 and it describes the packet in terms of the number of words in it, whether it is a unicast or a multicast/broadcast packet, lists errors detected, identifies the port through which the packet was received, and the port through which it is to be transmitted. The header also contains a unique 6-bit packet identifier used for CXD1700 internal housekeeping purposes. The header is used only for operations within the CXD1700; no part of it goes out when a packet is transmitted.

A packet header consists of two 32-bit words, as shown in Figure 1.2. The packet header originates in the receive port controller, during packet reception, and is output onto the switching bus as the very last two words of the incoming packet data stream. It is written into the packet buffer at the head of the packet. While it is on the switching bus, the header is monitored by all CXD1700 ports, because it contains the destination port ID. It also contains bits that describe various errors in the packet and the error bit contents are checked to reconcile the integrity of the packet and decide whether it is discarded (filtered) or forwarded. If forwarded, certain data is copied out of the header and written into the Tx queue of the destination port, along with the packet starting address in the packet buffer. This sets the packet up to be read out of SDRAM and transmitted at a later time.

Among other information, header word 2 contains a 6-bit packet ID number that is used to identify a specific packet among the packets in a particular packet buffer. It is this packet ID number which, together with its SDRAM address, positively identifies the packet within CXD1700; it is used to track the packet between the time it is received and the time it is transmitted. Other information (in word 1) includes a 5-bit number that identifies the port on which the packet was received (source port) and a 27-bit long bitmap that identifies the port(s) on which the packet is to be transmitted (destination port). Word 2 also contains the total byte count in the packet, type ID (unicast, broadcast, or multicast), and five different error bits.
### Packet Header -- Word 1

| 31 | 27 | Source port number, binary value. |
| 26 | 0  | Destination port ID, 27-bit bitmap, in which a 1 bit indicates the packet is to be transmitted through this port; LSB is port 0, MSB is port 26 (CPU port). |

### Package Header -- Word 2

| 31 | 26 | Packet ID Number; this value is different among packets in any one packet buffer. |
| 25 |    | Error in packet |
| 24 |    | CRC Append and Pad Insert; this bit position is used only by host CPU for packets it sends out. When set to 1, causes destination port controller to calculate and insert CRC value into packet. If packet is less than 64 bytes, pad bytes will be inserted. |
| 23 |    | Runt Error. Packet was received with less than 64 bytes |
| 22 |    | Reserved (not used) |
| 21 |    | Frame Error: Packet was received with less than integral number of bytes. |
| 20 |    | CRC Error: Packet received had CRCerror. |
| 19 |    | Broadcast packet |
| 18 |    | Multicast packet |
| 17 |    | Reserved (not used) |
| 16 |    | Overflow Error: An Rx FIFO overflow occurred during packet reception |
| 15 | 12 | Reserved (not used) |
| 11 | 2  | Word Count; the total number of 32-bit words in packet buffer that constitute this packet; (used by transmit port controller to schedule memory read operations) |
| 1  | 0  | Bytes in Last Word; the total number of bytes of valid data in the last 32-bit word of this packet (00=4, 01=1, 02=2, 03=3). This value is used by transmit port controller to schedule memory read operations. |

---

**Figure 1.2.** Packet Header Word Format

### 1.7 Flow Control

Flow control in the CXD1700 is highly user manageable -- the user programs for each port individually the level of congestion at which flow control is initiated. For full duplex operation the user can program the pause time value in pause frames, but for half duplex, the user has a choice of two different flow control mechanisms. If no flow control is enabled, an incoming packet is simply discarded when backpressure is applied.

The basis of flow control is a shared backpressure line between all ports. This line is used to activate the preprogrammed flow control response in the port which is receiving a packet at the time. As described in Section 1.3.1, during early stages of the reception of a packet, at the conclusion of lookup, the source and destination port numbers are put onto the switching bus. If at that time the destination port is in a congested state, it generates a backpressure signal and places it on the common backpressure line. All other ports ignore the backpressure signal, except the source port attempting to forward the packet to the congested port. The source port is at that time accumulating the first bytes of the packet in its Rx FIFO and applies whatever flow control action it has been programmed to use against the link partner.

Sensing congestion in the transmit side of a port is done by using two reference values associated with the Tx queue: a high water mark and a low water mark. Both these reference values are user programmed, for each port separately, in port control register PCR1. Whenever the number of packets in the Tx queue of a particular port exceeds the programmed high water mark, that port enters the congested state. It stays in the congested state until the number of packets falls below the low water mark. If the ID number of a port appears on the switching bus (at the conclusion of a lookup operation) while the port is in the congested state, the
port outputs the backpressure signal. The port which originated the lookup request (its port number is also on the switching bus, in the source port field) recognizes the backpressure event and proceeds to implement flow control on its network segment.

As a rule of thumb, the water mark in PCR1 should be programmed to be about half of the corresponding packet buffer size in SDRAM. The low water mark should be set sufficiently below the high water mark to provide some hysteresis for smooth flow control operation.

The flow control responses and choice of techniques are also user programmable. The responses are set individually for each port (in PCR0). Flow control can be turned off entirely for any port, causing any packet destined for a congested port to be discarded. If flow control is activated, the technique used depends on whether the port is operating in full or half duplex. If in full duplex, IEEE standard 802.3x pause frames are used, with the pause time preprogrammed in PCR1. In half duplex mode, two techniques are available (selected in PCR0): per-packet and per-port. If per-packet flow control is selected, an immediate collision with the incoming packet is generated whenever backpressure is received from the destination port. If per-port flow control is selected, the current packet is received, queued in the Tx queue of the destination port, and stored into the packet buffer, all in the normal manner, but a collision is caused with any incoming packet arriving within the time window specified in PCR1.

Note that under the per-packet flow control only the packets destined for a congested port are affected. Any packet packet destined for an uncongested port is switched through, whereas under the per-port flow control, collision occurs with every packet within the time window, regardless of it destination port. It should also be noted that in each port (in PCR0) a user programmable option exists to disable flow control for broadcast packets. When this option is selected, the packet will be broadcast to all ports, and any port that is currently congested drops the packet.

1.8 VLAN
The CXD1700 has source MAC address based VLAN capability, which allows the definition of up to 1024 different VLANs, or broadcast domains. The CXD1700 provides the mechanical infrastructure for VLANs, but the VLANs must be defined by a system administrator, and/or program control, and written into designated SDRAM locations. In setting up VLANs, the host can assign nodes on the same network segment to the same or to different VLANs, or the same node can be assigned to multiple VLANs. Alternatively, the CPU can be programmed to implement port based VLAN, in which all nodes on one or more ports belong to the same VLAN.

The VLAN infrastructure is implemented in the CXD1700 with a control bit in a switch engine register and three lookup tables in SDRAM. The three tables are the Permanent Table, the Permanent Link Table, and the VLAN Table, and together they limit the forwarding of any multicast or broadcast packet according to its VLAN membership. The control bit is the Perm Table bit in ECR0 and either enables or disables the use of the Permanent and VLAN tables for looking up destination ports for incoming multicast or broadcast packets. Thus, the tables function as an alternate lookup source to the normal learning tables and the basic VLAN mechanism is such that whenever a broadcast or multicast packet (or a spanning tree BPDU) is received, the Perm Table bit is checked. If it is set, lookup is done in the Permanent Table. The source MAC address is used to locate a VLAN index, and this index is then used to enter the VLAN table to locate all destination port IDs that belong to the particular VLAN. The source and destination port IDs are output onto the switching bus, same as at the conclusion of a unicast packet lookup operation.

The VLAN table and Permanent Table are user programmed to reflect the topology of the VLANs. Refer to Sections 3.5 and 4.9 for more detailed information.

1.9 Port Mirroring
The CXD1700 port mirroring feature allows incoming packet traffic on any number of network ports to be echoed or mirrored to one user designated port. That means that every packet received on any mirrored port is forwarded to and transmitted from the user designated port. Mirroring is set up in Switch Engine Register 5, which contains a 5-bit address of the port to receive the mirrored traffic and a bitmap of all ports that are being mirrored. After the ports are selected, mirroring is turned on or off with a bit in Switch Engine Register 0.

During packet processing mirroring is implemented by the lookup/learning logic. Near the end of the lookup process, lookup/learning logic checks whether the source or destination port is to be mirrored and if so, it modifies the destination port ID map to include the user designated port that receives the mirrored traffic. Note that with cut-thru enabled, a packet destined for two or more ports can be switched to multiple destination ports simultaneously.

Refer to Section 4.9 for instructions on how to set up mirroring setup to avoid oversubscribing any single port.
1.10 Management Statistics Output

There is a management statistics probe in each of the 26 ports. The combined output of these probes provides a complete statistics output that profiles the receive and transmit activity in the CXD1700 and on the network segments. The CXD1700 provides raw data from which MIB variables can be constructed, in support of the requirements of the following standard MIBs:

- RFC1757 (RMON), Groups 1 through 4
- RFC1439 (Bridge MIB)
- MIB-II
- RFC1573 (The Interface Extensions MIB)
- RFC1643.

The statistics are output from the CXD1700 on a 4-bit parallel synchronous bus, in the form of two vectors: a 32-bit receive vector and a 49-bit transmit vector (see Table 1.2 and Table 1.3). A vector is generated every time there is a transmit or receive event in a port, a backpressure event, or a packet ID error event in a transmitting port.

The vector data output by the CXD1700 must be accumulated in the user’s statistics counter logic. To support any of the above MIBs, this counter logic needs to include a corresponding set of 32-bit counters, a vector parser, and a generic CPU interface, to provide access for the SNMP agent. (Note: Sony Semiconductor will have an External Statistics chip available in the near future.)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RxBackpressure</td>
<td>Packet forwarding was back pressured by a destination port.</td>
</tr>
<tr>
<td>30</td>
<td>RxPause</td>
<td>Received a valid Pause Control Frame</td>
</tr>
<tr>
<td>29</td>
<td>RxControlFrame</td>
<td>Received a valid Control Frame</td>
</tr>
<tr>
<td>28</td>
<td>RxCFUndfOp</td>
<td>Received a valid Control Frame with an Undefined Op Code</td>
</tr>
<tr>
<td>27</td>
<td>CarrierEvent</td>
<td>Carrier Event seen before this frame</td>
</tr>
<tr>
<td>26</td>
<td>RxDVEvent</td>
<td>Receive Data Valid seen before this frame</td>
</tr>
<tr>
<td>25</td>
<td>ReceiveOK</td>
<td>Received Frame was Good, no CRC error or code violation</td>
</tr>
<tr>
<td>24</td>
<td>Broadcast</td>
<td>Broadcast Frame</td>
</tr>
<tr>
<td>23</td>
<td>Multicast</td>
<td>Multicast Frame</td>
</tr>
<tr>
<td>22</td>
<td>CRCerr</td>
<td>Received Frame had CRC error</td>
</tr>
<tr>
<td>21</td>
<td>Dribble</td>
<td>Non-integral number of bytes received</td>
</tr>
<tr>
<td>20</td>
<td>CodeViolation</td>
<td>Received a code violation on MII interface (ports 25 and 26 only)</td>
</tr>
<tr>
<td>19</td>
<td>DropEvent</td>
<td>IPG too short, preamble too long, or non-pure preamble (if MACC bit 7 is set)</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>(Not used)</td>
</tr>
<tr>
<td>17:7</td>
<td>RxByteCount</td>
<td>Received byte count</td>
</tr>
<tr>
<td>6:2</td>
<td>PortNumber</td>
<td>Port originating this vector</td>
</tr>
<tr>
<td>1</td>
<td>Type</td>
<td>Tx Vector = 1; Rx Vector = 0</td>
</tr>
<tr>
<td>0</td>
<td>StartBit</td>
<td>Indicates start of vector; always 0</td>
</tr>
</tbody>
</table>
### Table 1.3 Transmit (Tx) Status Vector

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>TxIDError</td>
<td>Packet ID in Tx queue did not match packet ID read out of packet buffer</td>
</tr>
<tr>
<td>47</td>
<td>TxBackpressure</td>
<td>Tx queue congested — backpressure indication sent to source port</td>
</tr>
<tr>
<td>46:35</td>
<td>TxTotalByte</td>
<td>Total transmitted byte count (including those transmitted as retries)</td>
</tr>
<tr>
<td>34</td>
<td>TxPauseFrame</td>
<td>Transmit Pause Frame sent</td>
</tr>
<tr>
<td>33</td>
<td>TxControlFrame</td>
<td>Transmit Control Frame sent</td>
</tr>
<tr>
<td>32</td>
<td>TxDone</td>
<td>Transmit done</td>
</tr>
<tr>
<td>31</td>
<td>LateCollision</td>
<td>Late collision occurred during transmission</td>
</tr>
<tr>
<td>30</td>
<td>ExDefer</td>
<td>Excessive defer time, frame was aborted</td>
</tr>
<tr>
<td>29</td>
<td>MaxRetries</td>
<td>Maximum retries reached, frame aborted</td>
</tr>
<tr>
<td>28</td>
<td>Underrun</td>
<td>Transmit FIFO underrun</td>
</tr>
<tr>
<td>27</td>
<td>Jumbo</td>
<td>Frame larger than max. allowable size</td>
</tr>
<tr>
<td>26</td>
<td>Deferred</td>
<td>Transmission of this frame was deferred</td>
</tr>
<tr>
<td>25</td>
<td>Broadcast</td>
<td>Broadcast Frame Transmitted</td>
</tr>
<tr>
<td>24</td>
<td>Multicast</td>
<td>Multicast Frame Transmitted</td>
</tr>
<tr>
<td>23</td>
<td>CRCError</td>
<td>Transmitted Frame had CRC error</td>
</tr>
<tr>
<td>22:19</td>
<td>RetryCount</td>
<td>Number of retries made for this frame</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>(Not used)</td>
</tr>
<tr>
<td>17:7</td>
<td>TxByteCount</td>
<td>Transmitted byte count</td>
</tr>
<tr>
<td>6:2</td>
<td>PortNumber</td>
<td>Port originating this vector</td>
</tr>
<tr>
<td>1</td>
<td>Type</td>
<td>Tx Vector = 1; Rx Vector = 0</td>
</tr>
<tr>
<td>0</td>
<td>StartBit</td>
<td>Indicates start of vector; always 0</td>
</tr>
</tbody>
</table>

### 2.0 External Components and Interfaces

#### 2.1 SDRAM

The SDRAM is a standard configuration memory, that provides all external data storage for the CXD1700 (with the exception of management statistics storage). The SDRAM must be at least 4 Mbytes in size, expandable up to 16 Mbytes. Because the CXD1700 performs read and write transfers on every clock cycle, the SDRAM must be capable of full page burst mode operation at 40 MHz. It can have either LVTTL or TTL level I/O. The CXD1700 accesses the SDRAM using a 32-bit data bus (an extension of the CXD1700 internal switching bus) and therefore, multiple chips must be connected into a 32-bit wide configuration. The CXD1700 uses 12 address lines and 8 control lines to access the SDRAM.

The Sony reference design uses two 16 Mbit NEC Part No. μPD4516161G5-A15 SDRAMs, each 512Kx16 bits, connected into a 512Kx32 configuration. Each chip has two memory banks and together the two chips give 4 Mbytes of memory space. The SDRAM can be expanded in 4 Mbyte increments by adding two chips at a time to expand to 8, 12, or 16 Mbytes. The CXD1700 has four
A suggested memory map is contained in Section 4.0. This memory map optimizes the memory space of a 4 Mbyte SDRAM, but all address locations in the switch SDRAM are user programmable, except the mailbox address. The mailbox must be mapped to 0X0, because the switch is hardwired to always access at that address. Sizes of the aging and the Permanent tables are fixed at 512 Kbytes each, the sizes of all link tables are variable, as are the packet buffers. Section 4.0 also describes certain constraints on where the starting boundaries of tables must be and discusses memory partitioning tradeoffs.

At powerup time the CXD1700 initializes the SDRAM and all addresses that make up the memory map are transferred into various CXD1700 internal control registers. The CXD1700 also contains all circuitry required for refreshing the SDRAM.

Refer to section 6.0 for selected SDRAM timing diagrams.

2.2 Host CPU

The CXD1700 can be operated by a host CPU, or a simple FPGA and EPROM device. If a host CPU is used, the CPU, together with the bus bridge logic (Section 2.3), employs three separate mechanisms to do three basic tasks:

1. Read/write in the internal registers of the CXD1700.
2. Read and write in SDRAM
3. Receive and send packets through the CXD1700.

With these mechanisms, the CPU is able to operate the CXD1700. It can initialize the CXD1700 on powerup, set up and change operating modes throughout the CXD1700 at any time, it can age the learning tables, monitor status in the CXD1700, and manage external PHYs through the port 24 and 25 MDIO lines. The CPU can read and write at any location of the SDRAM, including all packet buffers, and thus is able to monitor packet traffic and selected operational processes of the CXD1700. The CPU can also receive and send packets through any of the 26 ports of the CXD1700, just the same as any other network node. With these same mechanisms the CPU can also implement VLAN, mirroring, the spanning tree algorithm, receive and send SNMP traffic, and execute network management agent/subagent programs. The CPU can, of course, also interface to other facilities, such as the management statistics counter logic described in Section 2.4.

The host CPU does all interfacing with the CXD1700 through the bus bridge logic (Section 2.3). The CPU (i.e. bridge logic) can never access the internal switching bus of the CXD1700. Instead, the CPU interfaces to the CXD1700 thru the mailbox location at address 0x0 in the SDRAM. There the CPU can place any one of three different commands to the CXD1700 (read registers, write registers, send packet). The CPU also places the data to be written into the registers, or the packet it wants to send out, into the mailbox. The CXD1700 deposits data read out of its internal registers into the mailbox. (Refer to Sections 1.3.3 and 4.2 for details about mailbox operations.)

The host CPU can access all SDRAM locations, including all packet buffers, by using a bus request/bus grant handshake with the CXD1700 to gain control of the external extension of the CXD1700 switching bus. The CPU must make burst mode read or write transfers to the SDRAM.

When a packet is to be sent to the CPU (designated as port 26), only the header of the packet is sent directly to the CPU (i.e. bus bridge logic), along with the address of the packet in the SDRAM packet buffer (the full packet, including the header, is written into the packet buffer of the receiving port). The starting address and the header are output by the CXD1700 onto the external extension of the switching bus concurrent with a HDRSTRB signal. The bridge logic must capture three 32-bit words off the bus and forward them to the CPU (refer to Section 3.3.4).

In choosing a CPU, there are, in general, no limitations on the CPU type, width of its data bus, or clock speed. Because the CPU does not carry out any bus operations on the CXD1700 internal switching bus, the only major requirement on the CPU is that in conjunction with the bridge logic, it be able to conform to the burst mode operation protocol of the SDRAM.

If the CXD1700 is designed into a minimum configuration unmanaged switch, and an FPGA type device is used in lieu of a host CPU, the FPGA and its bus interface logic need do only the first two of the three tasks listed above. It must be able to use the mailbox to write into the CXD1700 registers during initialization, and thereafter, it must be able to use the mailbox to execute a periodically timed write into register ECR8 to age the learning tables.

2.3 Bus Bridge Logic
Bus bridge logic is the interface between the host CPU and the CXD1700, as well as the SDRAM. It functions as a bus controller to handle data transfers between the host CPU bus and the external extension of the CXD1700 switching bus, thus facilitating access by the host CPU to the SDRAM. The bus bridge logic also connects directly to the CXD1700 across five command/control lines -- CMDRDY, TASKDONE, BREQ, BGRT, and HDRSTRB. With the CMDRDY and TASKDONE lines the CPU can command the CXD1700 to read or write its internal registers, and it can also send a packet in a manner similar to any other network node. With the BREQ and BGRT lines, the CPU can gain access to the external extension of the CXD1700 switching bus in order to read or write in SDRAM. The HDRSTRB signal is a data strobe the CXD1700 issues at the end of every packet reception, when the header is written to memory. The bridge logic parses the header to identify packets destined for the CPU.

The complexity of the bus bridge logic design depends on the specific host CPU used, its bus type and width, and its clock speed. However, the general requirements for the bus bridge logic are as follows:

1. Provide a CPU interface to translate and relay the five command and handshake signals between the CXD1700 and the CPU. Refer to section 6.0 for selected control signal timing diagrams.
2. Buffer read and write data between the CPU and SDRAM (mailbox and all other SDRAM read/write accesses). Handle 32-bit data to/from the CXD1700 bus extension and clock synchronization, as necessary to accommodate any difference between CPU bus speed and the 40 MHz bus clock. Generate and output all signals (address and control) for executing DMA burst mode reads and writes in SDRAM. Refer to section 6.0 for selected SDRAM timing diagrams.
3. Monitor the HDRSTRB signal, capture and store packet header data from the CXD1700. Then, determine whether the packet is for the CPU and signal the CPU to fetch the packet data. Refer to Section 6.0 for HDRSTRB timing diagram.

2.4 Statistics Counter Logic

The optional statistics counter logic provides external accumulation of the management statistics data output by the CXD1700 (refer to Sections 1.10 and 3.7). It can be a single or multiple chip circuit, that interfaces to the CXD1700 and the host CPU, and provides counters for accumulating raw management statistics data. The data is input from the CXD1700 across a 4-line synchronous interface, with the 40 MHz clock provided by the CXD1700. The host CPU interface is largely determined by the specific CPU used, but it must allow the CPU to read the counters as needed and allow the CPU to initialize and operate the control logic.

The statistics counter logic needs to have the following basic functional blocks:

1. Data input translation and buffering circuits.
2. A vector parser that can segregate incoming data by port and selects the counter group.
3. A memory for the counter groups, segregated by port.
4. A CPU interface, including a memory access interface and data output buffering facilities.

The number and size of counters depends somewhat on how the stored data is organized and the specific MIBs that need to be supported. If all data output by the CXD1700 is to be stored in 32-bit counters in raw format (i.e. not as MIB variables), the number of counters required is on the order or 40 per port. Peak data flow rates of 160 Mbps may need to be accommodated (see timing diagram in Section 6.0).

2.5 Physical Layer Device (PHY) Interface

Industry standard PHY devices — either manageable (MII PHYs) or unmanaged — can be connected to CXD1700 ports. Each of the 24 10Base-X ports has an integrated MAC unit with a standard serial network interface (SNI). The 7-line SNI can be connected to unmanaged PHYs, whose control lines are hardwired in either half or full duplex mode. However, because the two 100Base-X ports each has a built-in 802.3u compliant MII, the two pairs of MII lines can be routed to control manageable PHYs on all 26 ports. Thus, a mix of manageable and nonmanageable PHYs can be interfaced to the CXD1700. Refer to Section 6.0 for Rx and Tx timing diagrams for both, the 10 Mbps and 100 Mbps PHY interfaces.

It is important to note that for ports 0-23 the TxCLK input is fanned out to four ports, thus the use of quad PHYs is recommended on these ports. On ports 24 and 25 the TxCLK lines for each port are separate.

2.5.1 Nonmanageable PHY Interface. A nonmanageable PHY is connected to its port through the SNI. The PHY can be hardwired to operate in a known mode — either half duplex or full duplex — or it can be controlled by the CPU. During initialization the corresponding port registers must be programmed accordingly, as described in Section 4.5.

2.5.2 Manageable PHY Interface. A managed PHY is connected to a port through its data interface (either SNI or MII) and the two
serial management MDIO lines from either port 24 or 25. The serial management lines are daisychained to multiple PHYs. Each PHY device must be provided a unique 5-bit device address, so that the host CPU can address an individual PHY in the daisychain and access the management functions (registers) inside.

By distributing the PHYs among the two 100 Mbit ports in any convenient manner, manageable PHYs can be used on all 26 ports of the CXD1700. The serial management line pair per each port consists of a bidirectional MDIO line and the associated MDC clock line. Both lines are daisychained to all PHYs connected to that port. The clock is supplied by the port and its frequency is programmable in the main control register of its MAC. There are also five MII registers in each 100 Mbit port and these registers are used by the host CPU to send commands and data out to each PHY, or to read any of the PHY registers. In one register the CPU loads the PHY device address and the address of the specific register within that PHY that is to be accessed. In other registers the CPU can load data and either a write or read command. In this manner the CPU can, for example, enable or disable autonegotiation, read results of autonegotiation in the Link Partner Ability register, or force the PHY to operate in half duplex or full duplex. (Refer to Section 4.4 for detailed procedures for reading and writing the PHY registers.)

In general, there are two ways the serial management interface can be utilized. First, it is possible to allow the PHY to autonegotiate and set the link configuration. Then, the CPU must read the Link Partner Ability register and program the PCR and MACC registers of the port to conform to the configuration discovered by the PHY. Alternatively, autonegotiation can be disabled, and the PCR, MACC and PHY registers can all be set to either full or half duplex operation manually.

### 3.0 Functional Description

#### 3.1 Startup and Initialization

The CXD1700 is initialized on powerup, as well as after a hardware reset. The two events are essentially the same, because in each case the hardware RESET signal must be asserted, thus forcing all internal registers to their default states.

On powerup, the boardwide RESET must be asserted on CXD1700 pin Y2 and then CXD1700 must be initialized by writing selected registers and performing some startup operations. The host CPU sets up all switch engine control and status registers (ECRs), port control registers (PCRs) in each of 26 ports, MAC control registers (MACs, LSA1, LSA2) in all ports, and MII registers (MCMD, MADR, MWTD, MRDD, MIND) in ports 24 and 25. In the process, the host CPU loads lookup table addresses into the ECRs, and packet buffer addresses into the PCRs. At the end of initialization, the SDRAM memory map has been loaded into the CXD1700. In addition, each port is set up for either half or full duplex operation, cut-thru and flow control are set up for each port individually, and other per port features are selected. The common features, such as VLAN, port mirroring, and spanning tree facilities are set up in the ECRs.

If manageable PHYs are used, the CPU sets up the Autonegotiation Advertisement register in each PHY, checks autonegotiation results in the Autonegotiation Link Partner Ability register, and then sets the corresponding CXD1700 port and MAC operating mode accordingly.

At the conclusion of initialization, the CXD1700 is ready to process packets on all ports that have been specifically turned on. Refer to Section 4.3 for a more detailed description of initialization procedures.

#### 3.2 Packet Forwarding Decision Protocol

The CXD1700 implements the following lookup protocol to determine the destination port IDs of all incoming unicast packets.

1. Lookup is executed by searching the Current learning table with the destination MAC address. If the MAC address is found, the destination port ID is supplied to the receiving port in the normal manner, described under Lookup and Learning.
2. If the MAC address and the corresponding destination port ID are not found in the Current table, the VLAN bit (bit 27 in ECR0) is checked. If this bit is 0, VLAN is not enabled, and the packet is broadcast to all ports (including CPU port) except to the source port.
3. If a destination port ID is not found in the Current table, but the VLAN bit is set (VLAN enabled), a second lookup is done, by entering the Permanent table with the source MAC address. If an entry is found with a corresponding VLAN index, that index is used to access the VLAN table to find the port map of a specific VLAN. The packet is then distributed to all destination ports belonging to that VLAN.
4. If the VLAN bit is set, but the destination port cannot be determined from the Permanent table, the packet becomes an unresolved packet and is sent to the host CPU. That is, the packet and its header are written into the packet buffer in the normal manner (as in store & forward mode), but the packet starting address and the packet header are forwarded to the host CPU (ref. Section 3.3.4). It is up to the system administrator to provide resources in the host CPU to resolve the disposition of the packet. The CPU can resend the
packet with a mailbox command, without moving the packet (refer to section 3.3.5).

The CXD1700 implements the following lookup protocol to determine the destination port IDs of all incoming multicast and broadcast packets.

1. The VLAN bit (bit 27 in ECR0) is checked. If this bit is 0, VLAN is not enabled, and the packet is broadcast to all ports (including CPU port), except the source port.
2. If the VLAN bit is set (VLAN enabled), lookup is done in the Permanent table with the source MAC address. If an entry with a corresponding VLAN index is found, that index is used to access the VLAN table to find the port map of a specific VLAN. The packet is then distributed to all destination ports belonging to that VLAN.
3. If the VLAN bit is set, but the destination ports cannot be determined from the Permanent table, the packet becomes an unresolved packet and is sent to the host CPU. That is, the packet and its header are written into the packet buffer in the normal manner (as in store & forward mode), but the packet starting address and the packet header are forwarded to the host CPU (ref. Section 3.3.4). It is up to the system administrator and the resources provided to the host CPU to resolve the disposition of the packet. The CPU can resend the packet with a mailbox command, without moving the packet (refer to section 3.3.5).

3.3 How Packets are Switched Thru the CXD1700

The following sections 3.3.1 through 3.3.4 describe the sequence of events by which packets are processed through the CXD1700. The description in 3.3.1 starts with basic store & forward mode processing of a unicast packet from a receive port to a single transmit port. Then, variations of the basic process, such as cut-thru, multicast/broadcast packets, and packets forwarded to the CPU, each is described separately. Other special conditions that affect packet forwarding, such as congestion, mirroring, or the spanning tree are described separately elsewhere. Note that regardless of the forwarding mode, all incoming packets without exception, are stored in the receiving port SDRAM packet buffer. Any incoming packet whose byte count is less than 64, or more than maximum length (nominally 1518, but adjustable in MACC register) is filtered.

3.3.1 Store & Forward Mode Transfer. The input of a packet to a port starts when the carrier sense signal CRS is asserted by the PHY. The serial data that follows is converted to 8-bit bytes in the MAC and clocked into the receive (Rx) FIFO. The 8-bit bytes are assembled into 32-bit words. As soon as there are 12 bytes in the Rx FIFO (3 words, representing destination and source MAC addresses of the frame), the receive port controller sends out an interrupt to the bus control logic of the switch engine, in order to gain access to the switching bus. When the bus control logic issues a bus grant to the receive port (source port) controller, the controller requests a lookup operation and places its port ID onto the bus. This is followed sequentially by three 32-bit words of the destination and source MAC addresses (48 bits each). The lookup/learning logic in the switch engine fetches both 48-bit addresses off the bus and loads these into its own registers.

Lookup/learning logic then uses the destination MAC address to perform a lookup operation in the Current learning table, to determine the CXD1700 port through which the packet is to be forwarded (the destination port). (The lookup sequence is described in more detail in Section 3.4, Lookup, Learning, and Aging.) As the result of lookup, the port ID of the destination port, along with the port ID of the source port, are placed onto the switching bus (destination port ID is a bitmap, source port ID is 5-bit hex code). The destination port ID is stored in a register in the source port controller for later use. (Flow control decision is made at this time also — see Flow Control, Section 1.7)

Next, the lookup/learning logic executes a learning operation. Using the packet source MAC address to enter the learning table, a check is made whether the MAC address and its associated port ID already are recorded in the Current learning table. If they are not, the source MAC address and its CXD1700 port ID are written into the Current table. Then, the same learning operation is performed once more on the Next learning table. This completes the learning operation.

While lookup and learning occur, additional packet data is accumulating in the Rx FIFO of the source port. As soon as there are 64 bytes (16 words) in the FIFO, the receiving port controller is ready to initiate the storage and forwarding of the packet. The port controller issues a bus request to bus control (bus requests are prioritized, with 100 Mbit ports having higher priority than 10 Mbit ports). When the bus is granted, the receive port controller places a start-of-transfer message onto the switching bus. This message includes its own port ID, and the ID of the destination port. The transmit port controllers of all ports are monitoring this message and thus the destination port is alerted of an upcoming packet to be transmitted. This start-of-transfer message is followed by a memory write request, in which the receiving port controller first supplies the memory starting address in its designated SDRAM packet buffer area, and then sequentially transfers sixteen 32-bit words onto the switching bus. The memory control executes a burst mode write into the SDRAM, until all 16 words are written. Unless the incoming packet is only 64 bytes long, the first memory write is followed by others, whenever another 64-byte increment of the incoming packet accumulates in the receiving port Rx FIFO. At the
start of each memory write operation the receiving port controller again places the source and destination port IDs on the bus, followed by a write request and an incremented memory start address.

Such memory write operations continue, until all bytes of the incoming packet are written into the designated SDRAM packet buffer. During the last memory write operation, the integrity of the packet is verified in the MAC of the receiving port, as well as its port controller, and a multistep end-of-packet sequence starts. First, the receiving port controller places a header presentation message onto the bus, which contains the starting memory address of the packet in the SDRAM. This address is recorded by the destination port transmit controller in a temporary register. Next, two words of packet header are placed onto the bus sequentially and written into the SDRAM at the head of the packet just written. The packet header, Figure 1.2, contains several different packet error bits, the packet word count, a packet ID number (used only within the CXD1700 packet memory management environment), along with the source and destination port IDs. The error bits in the header are read by the transmit port controller and an error check is made. If the packet contains any error, it is dropped. If there is no error, the packet ID number is stored in a temporary register and then, together with the memory start address, the packet ID is transferred from the temporary register into the Tx queue. With this transfer, all necessary packet identification information is queued and the packet is ready for transmitting. The destination port controller can now initiate the transmit operation whenever it is ready.

Transmission of a packet starts when its packet identifying information moves to the top of the Tx queue and when another transmit operation is not in progress. The transmit controller first requests the bus. Upon bus grant, it requests a memory read and places the packet starting address onto the bus, along with the number of words to be read (the initial read is always eighteen 32-bit words, or 72 bytes). Memory control logic performs a burst mode read of the 18 sequential memory locations. The data words are read by the destination port. The first two words read by the destination port are the packet header. These are stripped from the rest of the packet and some of its information is used by the port controller during the transmit operation. For example, to confirm the identity of the packet, the packet ID number from the header is reconciled against the packet ID number stored in the Tx queue earlier. The packet byte count is used to program subsequent memory read cycles. The remaining 16 words (64 bytes), following the two header words, are loaded into the Tx FIFO.

The transmit controller issues a start-of-frame command to the MAC and a handshake process is used by the MAC to meter data out of the transmit FIFO. The MAC may delay request for more data, or it may refuse data altogether and request a transmit restart. The transmit controller maintains data intact in the Tx FIFO until at least 64 bytes have been transmitted successfully (i.e. the standard collision window has passed). The MAC converts 32-bit data into 8-bit bytes, the data is encapsulated into the ethernet frame structure and is sent out to the PHY.

Similar to receive, multiple memory read operations are executed, as necessary to read the entire packet out of SDRAM into the transmit FIFO. For any packet longer than the minimum size, the transmit port controller issues a new memory read request whenever the transmit FIFO is less than one third full (in 10 Mbit ports the FIFO is 64 words long, in 100 Mbit ports 128 words). Each subsequent read operation fetches enough words from the packet buffer to fill the FIFO, until the last read fetches only the remaining bytes of the packet.

After the last data has been unloaded from the transmit FIFO and forwarded to the MAC, along with an end-of-packet marker, the transmit operation is complete. The transmit port controller FSM resets to its starting state and transmission of the next packet can begin.

### 3.3.2 Cut-thru Mode Transfer

The following description of cut-thru packet forwarding is an extension of the store & forward mode above. In cut-thru forwarding, a packet passes directly from the receiving port Rx FIFO via the switching bus to the destination port Tx FIFO. Thus, the packet is switched through the CXD1700 with minimum forwarding latency.

A cut-thru packet transfer can occur only if all of the following conditions are satisfied:

1. The receiving port is cut-thru enabled.
2. The destination port is cut-thru enabled.
3. The Tx queue in the destination port is empty.
4. A transmit operation is not currently in progress in the destination port.

(Whether or not any port is cut-thru enabled is determined by bits 29 and 28 in its PCR0 register. Bit 29 allows the port to cut thru any packet it receives; bit 28 allows the port to transmit any packet sent to it by a cut-thru enabled receiving port.)

The input of the packet, lookup, and learning, all occur exactly as in store & forward mode, but when the first 64 bytes are accumulated
in the Rx FIFO, the receive port controller outputs a variation of the store & forward start-of-transfer message, which specifically indicates that the packet may be cut through, thus satisfying condition 1 above. In all other ways the cut-thru start-of-transfer message is identical, in that it contains the source and destination port IDs. If the destination port is cut-thru enabled (second condition above), its transmit controller reads the ID of the receiving port off the bus, stores it in a temporary register and goes into an alert state. If the remaining two conditions are also satisfied, the transmit port controller sets up to receive and transmit data in the cut-thru mode. From there on, the events are again the same as in store & forward: the receiving port controller issues a memory write request and puts the starting address in the packet buffer onto the bus. When the bus is granted, the receiving port controller outputs sequential 32-bit data payload words onto the switching bus. However now the data words, in addition to being written into the SDRAM packet buffer, are also loaded into the Tx FIFO of the transmit port. As the Tx FIFO accumulates data words, the transmit port controller initiates transmission — data is read out of the Tx FIFO, goes to the MAC and is transmitted via the PHY out onto the network media. If the packet is longer than 64 bytes, more memory write requests are issued by the receiving port controller and the above three simultaneous events continue, until the entire packet is written into the Tx FIFO, as well as into the packet buffer.

Reading of the data out of the Tx FIFO and transmitting it, of course, continues after the last data has been transferred into the Tx FIFO and the packet buffer. During this time the normal end-of-packet sequence also occurs. It is the same as in store & forward mode: receiving port puts out the end-of-packet message onto the bus, followed by the packet header, and the error bit check in the destination port (the header is also written into the packet buffer in the normal manner). The one difference is that in cut-thru no packet ID number or memory start address need be entered into the transmit port Tx queue. Instead, at the end of the cut-thru transfer the transmit port controller temporary registers are simply cleared and the Tx queue remains empty.

There are several aspects to be noted about cut-thru transfers. First, because the transmission of the packet normally starts before the last word is loaded into the Tx FIFO and the header comes onto the bus, it continues to the end, even if an error is detected at the end, when the header error bits are checked. The error is noted in the management statistics Tx vector, but the whole packet goes out as is and no further action is taken.

Second, once started, cut-thru is aborted only under two conditions: if the Tx FIFO overflows or if the Tx FIFO underflows. In either case, the transmit port controller automatically reverts back to store & forward mode — when the complete packet has been received it stores the packet ID information in the Tx queue and it is subsequently transmitted in store & forward mode. The backing up of every cut-thru packet in the packet buffer, also has the advantage that it allows a multicast packet to be transmitted at a later time on those ports where cut-thru could not be done.

### 3.3.3 Multicast and Broadcast Packets
Multicast and broadcast packets are forwarded by the CXD1700 in one of two different ways, depending on whether VLAN processing is active or not. In each case, multicast and broadcast packets are handled identically by the CXD1700. If VLAN processing is disabled (ECR0 bit 27=0), all multicast and broadcast packets are simply transmitted through all ports. If VLAN processing is enabled (bit 27 = 1), the source port address is used to do lookup in the Permanent table. If an index is found, if index is used to access the VLAN table to identify all ports that are part of the same VLAN as the sending node. The packet is distributed only to those nodes included in the VLAN map. If no index is found, in the Permanent table, the packet becomes an unresolved packet and is sent to the CPU.

### 3.3.4 CPU Packets — Receive
Any network node can send packets to the CPU in the same manner as to any other node. The CXD1700 forwards two categories of packets to the CPU: 1.) those specifically addressed to the CPU (by MAC address, or as multicast/broadcast packets) and 2.) those whose destination port cannot be determined by the CXD1700 either in its learning tables or in the Permanent table (the process is described in Section 3.2, Packet Forwarding Decision Protocol).

As a matter of course, the mailbox controller and CPU port control logic monitors all packet traffic on the switching bus. Specifically, each time a receiving port controller initiates the end-of-packet sequence (ref. Section 3.3.1), the CPU port controller generates a HDRSTRB pulse. The HDRSTRB triggers the bus bridge logic to fetch three words and write them into a bus bridge logic storage device: the packet starting address from the SDRAM address lines, and the two header words from the switching bus (see HDRSTRB timing diagram in Section 6.0). The bus bridge logic must examine the second header word to determine whether the packet is destined for the CPU. If it is not, the three words are simply discarded, but if the packet is for the CPU, bridge logic notifies the CPU. Thereafter, it is up to the CPU to read the memory starting address out of the bridge logic storage device and retrieve the full packet. If the CPU will delay processing the packet, the packet can be read out of the SDRAM packet buffer and transfer it to a CPU memory location. This removes the packet from the packet buffer and avoids the possibility that it will be overwritten by incoming packet traffic, as packet buffer addressing loops around.
When operating in the VLAN enable mode, with port based VLAN maps, any packet sent to the CPU for address resolution can be forwarded from the packet buffer, after the appropriate entry is made in the Permanent table and VLAN table.

3.3.5 CPU Packets — Transmit. The CPU can transmit a packet to any network node using the 0x60 mailbox command (Section 1.3.3). The packet is transmitted with the aid of the CXD1700 mailbox controller, which performs selected functions that are the same as those performed by the Rx port control during a normal store & forward operation. The CPU can construct a new packet, or resend an existing packet stored in any packet buffer in SDRAM.

To send a new packet, the CPU must first assemble and store it in the SDRAM. Typically, this will be done in the mailbox area, starting after the first 64 words (all packets must start on 64 word boundaries). Alternatively, the CPU can assemble the packet in any other available location in the SDRAM. As a part of the packet, the CPU must also construct and write certain parts of the standard two header words at the head of the packet. The CPU must fill in all data into the first header word, including the packet ID number. In the first header word, the CPU also has a specific option to set bit 24 if a CRC value needs to be calculated and inserted in the FCS field of the frame. If the bit is set, the CRC calculation will be done by the transmit port controller at the time the packet is transmitted. If bit 24 is set, the transmit port MAC will also pad any packet with less than 64 bytes to the minimum size (64 bytes, including FCS).

To send a packet already existing in a packet buffer, the CPU simply must provide the starting address of the packet with the 0x60 transmit command.

To execute the transmit of a packet, the CPU writes the packet transmit command (0x60), including the SDRAM starting address of the packet, into the mailbox and then issues a CMDRDY to the CXD1700. In response, the CXD1700 mailbox controller reads the command out of the mailbox and proceeds to process it. It starts by putting the packet starting address on the switching bus and issuing a combined memory read request/lookup command. This command is acted on by both the memory control and lookup/learning logic. Memory control reads 12 bytes of source and destination address from the packet and lookup/learning logic fetches the addresses and executes a normal lookup operation. At conclusion of lookup, it publishes the destination and source port IDs on the bus in exactly the same manner as in store & forward mode. As in store & forward, all ports monitor these port IDs published by lookup/learning.

The mailbox controller stores the port IDs and performs several sequential tasks. First, it checks the backpressure line to ascertain that the destination port is not applying backpressure. If it is, the mailbox controller executes a write to the mailbox (to memory location 1) and sets the MSB at that location. This bit indicates to the CPU that the packet cannot be transmitted at this time. If there is no backpressure, the transmit sequence proceeds and the mailbox controller issues the same header presentation message that a receive port does at the end of a packet in store & forward mode: packet starting address, followed by two header words. All ports monitor this message, but that port which is identified in the second header word as the destination port loads the packet identifying information into its Tx queue. (Note also that after it outputs the header, the mailbox controller sends a TASKDONE signal back to the CPU.) From there on, the destination port Tx controller takes over and transmission of the packet occurs exactly the same as described for the store & forward mode.

3.4 Lookup, Learning, and Aging

As described in Section 1.3.1, there are three identical designated learning tables in SDRAM. Into these tables MAC addresses and corresponding port ID numbers are learned from each incoming unicast packet. At any given moment learning is done into two of the tables, designated as Current and Next, whereas the third table, designated Empty, has all 0s written in it. Periodically, the host CPU issues an aging command and the tables are rotated, so that Next becomes Current, Empty advances to Next, and Current is cleared and becomes Empty. In this manner MAC addresses are kept current and inactive addresses are purged, after the interval programmed into the host CPU. After the MAC addresses and port IDs are written into the learning tables, the tables are used for looking up port IDs for the purpose of forwarding unicast packets. Destination MAC addresses are always looked up in the Current table.

Each learning table has a linking table permanently associated with it. During the aging rotation, the linking table rotates along with its learning table. MAC addresses and port IDs are stored in the tables in multiple parallel linked lists, with the first address in every list located in the learning table, but the rest of the list always in the linking table. The following paragraphs describe the data storage formats in the learning tables, how data is entered during the learning process, how it is aged, and how data is read out during lookup.

3.4.1 Learning Table Structure. For every MAC address, two 32-bit words are stored in the learning tables, as shown in Figure 3.1. To limit memory space requirements and to facilitate the indexing of 48-bit long MAC addresses in the learning tables, a special addressing scheme is used. The SDRAM is addressed with the lower 16 bits of every MAC address and only 31 of the remaining 32 bits of a MAC address are stored in the SDRAM. This scheme has two implications. First, it dictates that an learning table must be
65K locations long, otherwise it must be able to store 65,536 individual MAC addresses, each with different lower 16 bits. Second, it means that if two MAC addresses come in with identical lower 16 bits (but different upper bits), they must be accessible at the same SDRAM address. The first requirement is satisfied by allotting 512K bytes of SDRAM space to each of the three learning tables, but to satisfy the second requirement, linked lists are used. To conserve memory space, the first entry of every linked list is stored in an learning table, but the second and all subsequent entries are stored in the linking table associated with that learning table. In the linking table, entries belonging to different linked lists can be stored at contiguous memory locations and therefore linking tables can be as small as 4K bytes each. The above data storage structure yields multiple parallel linked lists. The first MAC address of any linked list is in an learning table, but all subsequent addresses that hash to the same location are in the associated linking table.

The contents of each 32-bit word pair entry at a MAC address are as follows. Word 1 contains the high order 31 bits of the MAC address and a housekeeping bit. Word 2 contains a 5-bit ID of the port to which the MAC address belongs, and a linking address that points to the memory location where the next entry in the linked list resides. The LSB of Word 2 is the link bit and it simply indicates whether there is another entry in the linked list, or this is the end of the list. Note: the MSB of every 48-bit MAC address is the I/G bit (individual/group address flag) and it merely distinguishes between multicast or unicast addresses. The I/G bit is read by the switch engine and used to decide whether the MAC address belongs to the learning tables or the Permanent table; the bit does not need to be written into the tables.

Learning Table Entry --Word 1

<table>
<thead>
<tr>
<th>31</th>
<th>1</th>
<th>MAC Address, MSB</th>
<th>0</th>
</tr>
</thead>
</table>

Legend:
- 31 : 1 Upper 31 bits of MAC address
- 0 Valid Entry Bit; if this bit = 1, this is a valid MAC address entry; if 0, entry is not valid

Learning Table Entry --Word 2

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Legend:
- 31 : 29 Unused
- 28 : 24 Port number from which the address was learned
- 23 : 1 Link address for accessing the link table
- 0 Link bit; a logic 1 indicates there is a further entry in this link; a 0 indicates end of link

Figure 3.1. Learning Table Word Format

3.4.2 Learning and Aging. Learning MAC addresses on any individual port is enabled or disabled by setting a bit in ECR6. When it is enabled, learning is done on every incoming packet and is done twice — once into the Current table and a second time into the Next table. When a packet comes into a port, the 48-bit source and destination MAC addresses are extracted and loaded into the lookup/learning logic block. From its internal registers, the lookup/learning logic reads a 7-bit fixed address, that defines the start of the Current learning table in SDRAM (the 7-bit value for each of the three learning tables is loaded during initialization into ECR1, ECR2, and ECR3). The 16 LSBs of the source MAC address are appended to the fixed address, to come up with a 24-bit SDRAM address (last bit is always 0).

This address is used to access one of the 65K locations in the Current learning table. If no entry is found in the learning table, two 32-bit words that define the 31 MSBs of the incoming MAC address and the corresponding port are written into the memory. No linking address is entered into bits 23:1 of Word 2 and a 0 is written into the LSB of Word 2 (the link bit) to indicate that there is no further link from this memory location. This completes the learning of the MAC address into the Current table.

If, however, an entry is found in the learning table, the 31 MSBs are read out from that location and compared to the 31 bits of the incoming MAC address stored in lookup/learning. If there is a match, it indicates that the memory entry is the same as the incoming MAC address and no learning entry need be made. If there is no match, the link bit (word 2 LSB) is checked. If a 1, it indicates that there is a further entry in this linked list, the 23 bit linking address is read out and used to access the linking table to read the second entry. The second entry is checked for a match in the same manner as the first and if no match is found, reading proceeds on to the third linked entry, and so on. If the last entry in the linked list is reached (indicated by its link bit being a 0) and no match is found, it
means that the incoming MAC address needs to be added at the end of the linked list. To do this, the next link address is obtained from a stack in the lookup/learning logic and two tasks are performed. First, the new link address is written into bits 23:1 of Word 2 of the last entry, the link bit is changed from a 0 to a 1, indicating that now there is a further entry in the linked list. Second, a new 2-word entry for the incoming MAC address is written at the link address location; its link address is now left blank and its link bit a 0.

This completes the learning process in the Current learning table. It is then repeated a second time in the Next table. Note that the Next table will always have fewer MAC address entries than the Current table, because the Next table has been accumulating entries for one aging interval, whereas Current has been accumulating entries for two.

Aging occurs when the host CPU writes a logic 1 bit into bit position 31 of ECR8. This causes lookup/learning logic to execute the rotation by which the Current table becomes the Empty table, Next becomes Current, and Empty becomes Next. The interval of the rotation is determined by the CPU; the typical value used is 5 minutes. After a rotation occurs, the table that becomes Empty, is cleared of all entries: the memory controller writes all 0s in it as a background task. While this background task is in progress, bit 31 of ECR0 is a logic 1 and is reset to 0 as soon as the task is complete.

3.4.3 Lookup. Lookup is done for every incoming packet. A request for lookup is always initiated by the receiving port, when it places the destination and source MAC addresses onto the switching bus. The two MAC addresses are loaded into the lookup/learning logic block and the lookup operation is started.

The basic mechanics of a lookup are similar to learning. Whereas the object of learning is to write a source MAC address and its port ID into the learning tables, lookup is done to read out a port ID corresponding to a MAC address written into the table during learning. The lower 16 bits of the destination MAC address are again used as the address for accessing the learning table, the same as during learning. The linked entries are examined by comparing the upper 31 MAC address bits found in the aging/link tables to those of the destination MAC address, also same as during learning. When the correct entry in the aging or linking table is found, the port ID is read out, processed by the lookup/learning logic, and a 27-bit bit map identifying the destination port is output onto the CXD1700 switching bus, as described Section 3.3.1. Note that if port mirroring is active, this bit map may be further modified by the inclusion of the port designated to receive mirrored packets.

A slight variation of the lookup process occurs whenever the CPU sends a packet. In that case the destination and source addresses are not published on the switching bus by the port, but must instead be first read out of SDRAM, in order that they can be loaded into the lookup/learning logic (refer to Section 3.3.5). From there on, lookup is the same as described above.

3.5 VLAN Implementation

The VLAN mechanism in CXD1700 is modeled after the lookup and learning mechanism for unicast packets (Sec. 3.4.1). It makes use of the same kind of lookup table structure, stores very similar two 32-bit words per MAC address, uses the same method of addressing SDRAM with 16 LSBs of a MAC address, and uses linked lists. The difference is that no dynamic learning is done into the tables, nor are the tables aged. The VLAN related tables are written once by the host CPU and to make any change to a VLAN, the tables must be rewritten by the CPU. Another difference is that lookup is a two tier process: instead of reading a destination port ID directly out of Word 2, an index is provided to an additional table, where the VLAN port IDs are stored in the form of a bit map. VLAN lookup is done with the source instead of destination MAC address.

The VLAN infrastructure consists of a Permanent table, its associated Permanent Linking table, a VLAN Map table, and bit 27 in ECR0. The Permanent table is structured the same as the learning tables are — it has 64K entries and each MAC address is represented by two 32-bit words, as shown in Figure 3.2. The two words are very similar to those in the learning tables, except that in Word 2, in the place of a port ID, there is an index to the VLAN table, where all port IDs constituting the VLAN are found. The linking address is shortened to 21 bits (from the 23 bits in learning tables). The starting addresses of the Permanent and VLAN tables are stored in ECR4; the starting address of the Permanent Link table is not stored in any switch engine register, because writing of that table, along with the implementation of the linking algorithm, is done by the host CPU and does not involve the switch engine.

Whenever a multicast or broadcast packet is received (as indicated by a 1 in the I/G bit position of the destination MAC address), and the Perm Table bit in ECR0 is set, the following lookup procedure is performed. The destination address is loaded into the lookup/learning logic, which then uses the lower 16 bits to address the Permanent table. The upper 31 bits of Word 1 at the addressed location are compared in lookup/learning logic to 31 bits of the destination MAC address to find if there is a match. If there is a match, the index into the VLAN table is read out from Word 2 and used to address the VLAN table. From the VLAN table a 27-bit long bitmap (Figure 3.3) is read out and output onto the CXD1700 switching bus. If no match is found at the entry location into the Permanent table, reading is continued along the linked list in the same manner as described in section 3.4.3 for unicast packets, until a
match is found.

**Permanent Table Entry --Word 1**

<table>
<thead>
<tr>
<th>31</th>
<th>MAC Address, MSB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

- 31 : 1 Upper 31 bits of MAC address.
- 0 : Valid Entry Bit; if this bit = 1, this is a valid MAC address entry; if 0, entry is not valid

**Permanent Table Entry --Word 2**

| 31 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    |    | 21 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |

**Legend:**

- 31 : 22 Index to VLAN table entry
- 21 : 1 Link address
- 0 : Link bit; a logic 1 indicates there is a further entry in this link; a 0 indicates end of link.

![Figure 3.2. Permanent Table Word Format](image)

**VLAN Table Entry**

| 31 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 27 | 26 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |

**Legend:**

- 31 : 27 Reserved; on write set to 0
- 21 : 1 Bit map of VLAN destination ports, network ports 0-25, port 26 (CPU port)

![Figure 3.3. VLAN Table Word Format](image)

### 3.6 Spanning Tree

The CXD1700 provides the infrastructure with which the host CPU can develop and implement an active spanned tree topology using the IEEE 802.1D spanning tree algorithm. The infrastructure includes a mechanism for forwarding BPDUs to and from the host CPU, ECR6 and ECR7 for storing the spanning tree learning and forwarding maps, and network station addresses in MAC registers LSA1 and LSA2.

BPDUs are forwarded to the CPU using the VLAN infrastructure: a special VLAN map is constructed to act as a mask and identify and forward BPDUs only to the CPU. The CPU generates BPDUs and uses its normal packet transmit facilities to send them out. The ECR6 register allows the CPU to selectively enable and disable the learning of MAC addresses on individual ports during the development of the active spanned tree topology. The ECR7 register allows the CPU to selectively enable or disable forwarding on individual ports to implement the active topology.
3.7 Generation and Output of Management Statistics

The management statistics in each port come from three sources, as shown in Figure 3.4. The MAC generates the main body of both, the receive and transmit vector. The transmit and receive port controllers each generates a backpressure event marker, and the transmit port controller also generates a Tx ID error (which indicates that the packet ID queued in the Tx queue did not match the ID in the packet header read out of packet buffer).

All statistics data is clocked into two registers — a Tx vector register and an Rx vector register. The Tx vector register is 49 bits wide and the Rx vector register is 32 bits wide. Whenever a vector is assembled in one or both registers, a ready flag is set, which indicates that there is data to be output. To process the statistics vectors from all 26 ports, the common statistics control logic does round robin polling of all 26 ports. Wherever it encounters a port ready to output data, that port takes control of the bus. The port probe circuitry then proceeds to shift the vector out, four bits at a time, onto the ES DATA 3:0 bus. Once a port has been granted the bus, it can output either one or two vectors, before relinquishing the bus.

![Figure 3.4. Block Diagram of Management Statistics Logic](image)

The output on ES DATA 3:0 is a synchronous signal, accompanied by the 40 MHz clock. As shown in the timing diagram in the Timing Specifications section, all four data lines are maintained at logic high prior to a vector, but the start of a vector is marked by a logic 0 bit on ESDATA 0 line. This start bit is accompanied by a vector type bit on ESDATA 1 line (Tx = 1) and the first two of five port ID bits on ESDATA 2 and 3 lines. The rest of port ID bits and the first of vector data bits constitute the second 4-bit byte, followed by all remaining data bits in subsequent bytes. The Rx vector is 8 clock cycles long and the Tx vector is 13 clock cycles. Data always changes on the rising edge of the clock and is readable on the trailing edge. At the end of a vector all data lines again go to logic 1 state for at least one clock cycle, before the next start bit and another vector appear on the lines.

Note that the three events registered in the Tx vector each can occur at a different time, each is considered a separate event, which
generates a vector, and is sent to the external statistics counter logic. Thus, if for example, a TxID Error occurs and is latched into
the Tx vector register as the only input, it will set the ready flag and is sent out when statistics control grants the output bus to the port.
If, however, another event is latched into the Tx vector register before the TxID Error is sent out, the two events are combined into a
single vector and are sent out together. Thus, on Tx side any single event, any two, or all three events can create a valid vector and be
output by the statistics probe. On the Rx side, Rx Backpressure always occurs concurrent with the Rx vector data.

4.0 Programming Information

This section contains programming reference information for the CXD1700 and instructions for setting up and utilizing various
resources of the CXD1700.

4.1 SDRAM Memory Mapping
As described in Section 2.1, the SDRAM must be a minimum of 4 Mbytes in size and is expandable to 16 Mbytes, in increments of 4
Mbytes. All data is stored in 32-bit words. Table 4.1 is a suggested memory map, based on a minimum 4-Mbyte size SDRAM. The
following are some general guidelines for partitioning any size SDRAM:
1. The SDRAM is addressed on a word boundary.
2. The mailbox must be mapped at location 0x0. All other SDRAM locations are user definable.
3. Learning tables 1, 2, and 3, as well as the Permanent table, each has to have 65,563 entries, each entry consisting of two words.
4. Link tables 1, 2, and 3, as well as the Permanent Link table sizes, are variable. Overflow of link tables generates a flag, but
continued data entry will cause the loss of data.
5. Packet buffers have to start on 64 word boundaries (256 bytes), which means that the six LSBs of memory address must be all 0s
at the start of any packet buffer (packets within each buffer also start on 64 word boundaries). Packet buffer sizes can be distributed
differently from that shown in Table 4.1, to reallocate more or less packet memory to specific ports. Increasing packet buffer size
allows the raising of the high water mark in the Tx queue and thus handle higher levels of traffic bursts before congestion
(backpressure) occurs. With a larger memory packet memory buffers can be made larger. The CXD1700 writes into the packet buffer
in a rotating manner (i.e. it is a ring buffer). Whenever the last space in a packet buffer is filled, writing resumes at the start of the
memory space; thus it is theoretically possible that a valid packet can be overwritten. A 65K byte packet buffer provides storage for
43 max. size packets and 256 min. size packets. A 235K byte packet buffer provides storage for 155 max. size packets and 3680 min.
size packets.
6. The mailbox should be made large enough to allow ample space for storage of packets. The CPU will normally assemble and
store in the mailbox packets it is going to transmit. As elsewhere, all packets must start on 64 word boundaries.

4.2 CXD1700 Internal Registers
There are 144 registers in the CXD1700, as summarized in Section 1.5. All registers are 32 bits wide and are located either in the
network ports or in the switch engine. The following tables describe individual bit positions in all CXD1700 internal registers that are
used by the host CPU. The table heading in each case lists the table address and the Default column lists the value in which each bit comes up after a RESET.

To execute a register write or read, the CPU uses the two 32-bit wide commands shown in Table 4.2. The CPU can write or read
multiple contiguous registers with a single command that includes the port number, the starting register address, and the number of
registers to be written or read (for register addressing purposes, the switch registers are arbitrarily assigned port number 27). To do a
write, the 4-byte 0x20 command shown in Table 4.2 is written into the mailbox, at location 0x0 of the SDRAM, followed by as many 32-
bit data words as the number of contiguous registers to be written. Then, the CPU asserts the CMDRDY line to the CXD1700. The
CXD1700 fetches the command and data words out of the mailbox and executes the write. When complete, the CXD1700 asserts the
TASKDONE line. A read is done in the same manner, but with the 4-byte 0x40 command in the mailbox. When the CXD1700 is done
reading, it writes register contents (32-bit words) into the mailbox, starting at the location immediately after the command word.
### Table 4.1  SDRAM Memory Map

<table>
<thead>
<tr>
<th>Partition Name</th>
<th>Starting Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailbox</td>
<td>0x0</td>
<td>32Kbytes</td>
</tr>
<tr>
<td>Link Table 1</td>
<td>0x002000</td>
<td>4096 bytes = 1 Kwords = 512 entries (each entry is 2 words)</td>
</tr>
<tr>
<td>Link Table 2</td>
<td>0x002400</td>
<td>4096 Bytes = 1 Kwords = 512 entries (each entry is 2 words)</td>
</tr>
<tr>
<td>Link Table 3</td>
<td>0x002800</td>
<td>4096 Bytes = 1 Kwords = 512 entries (each entry is 2 words)</td>
</tr>
<tr>
<td>Permanent Link Table</td>
<td>0x002C00</td>
<td>4096 Bytes = 1 Kwords = 512 entries (each entry is 2 words)</td>
</tr>
<tr>
<td>VLAN Map Table</td>
<td>0x003000</td>
<td>1024 VLANs x 4 bytes per VLAN = 4096 bytes</td>
</tr>
<tr>
<td>Packet Buffer Port 0</td>
<td>0x003400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 1</td>
<td>0x007400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 2</td>
<td>0x00B400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 3</td>
<td>0x00F400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 4</td>
<td>0x013400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 5</td>
<td>0x017400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 6</td>
<td>0x01B400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 7</td>
<td>0x01F400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 8</td>
<td>0x023400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 9</td>
<td>0x027400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 10</td>
<td>0x02B400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 11</td>
<td>0x02F400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 12</td>
<td>0x033400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 13</td>
<td>0x037400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 14</td>
<td>0x03B400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 15</td>
<td>0x03F400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 16</td>
<td>0x043400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 17</td>
<td>0x047400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 18</td>
<td>0x04B400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 19</td>
<td>0x04F400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 20</td>
<td>0x053400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 21</td>
<td>0x057400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 22</td>
<td>0x05B400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 23</td>
<td>0x05F400</td>
<td>64 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 24</td>
<td>0x063400</td>
<td>230 Kbytes packet memory space</td>
</tr>
<tr>
<td>Packet Buffer Port 25</td>
<td>0x071A00</td>
<td>230 Kbytes packet memory space</td>
</tr>
<tr>
<td>Learning Table 1</td>
<td>0x080000</td>
<td>64K entries x 2 words per entry x 4 bytes per word = 512 kbytes</td>
</tr>
<tr>
<td>Learning Table 2</td>
<td>0x0A0000</td>
<td>64K entries x 2 words per entry x 4 bytes per word = 512 kbytes</td>
</tr>
<tr>
<td>Learning Table 3</td>
<td>0x0C0000</td>
<td>64K entries x 2 words per entry x 4 bytes per word = 512 kbytes</td>
</tr>
<tr>
<td>Permanent Table</td>
<td>0x0E0000</td>
<td>64K entries x 2 words per entry x 4 bytes per word = 512 kbytes</td>
</tr>
</tbody>
</table>
Table 4.2  Register Read/Write Commands  (CPU to CDX1700 via mailbox)

<table>
<thead>
<tr>
<th>Byte 3 Command Type</th>
<th>Byte 2 Port Number</th>
<th>Byte 1 Starting Register</th>
<th>Byte 0 Transfer Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Register = 0x20</td>
<td>Netwk Ports = 0 - 25 Switch Engine = 27</td>
<td>Any Writeable Register</td>
<td>Number of sequential registers to be written</td>
</tr>
<tr>
<td>Read Register = 0x40</td>
<td>Netwk Ports = 0 - 25 Switch Engine = 27</td>
<td>Any Readable Register</td>
<td>Number of sequential registers to be read</td>
</tr>
</tbody>
</table>

Switch Engine Register 0 (ECR0)  Control and Status  Address: 0x00

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>TBRT</td>
<td>RO</td>
<td></td>
<td>Table Rotation In Progress: A 1 at this location means an aging command has been received, the table rotated to Empty status is now being cleared (all 0s are being written to it). Monitor this bit to determine when clearing is complete (look for 0).</td>
</tr>
<tr>
<td>30:28</td>
<td>CRTB</td>
<td>RO</td>
<td></td>
<td>Current Lookup Table: Bitmap that identifies the Current table. A logic 1 in bit position: 30 = Learning Table 3 29 = Learning Table 2 28 = Learning Table 1 (These are status bits, a write to this location is ignored)</td>
</tr>
<tr>
<td>27</td>
<td>VLNEN</td>
<td>R/W</td>
<td>0</td>
<td>VLAN Enable: Logic 1 indicates that Permanent table is being used to do lookup for multicast and broadcast packets as well as packets that have no destination address available in current lookup table. If this bit is logic 0, all multicast and broadcast packets and all packets without destination port address are forwarded to all ports 0-27.</td>
</tr>
<tr>
<td>26</td>
<td>MIREN</td>
<td>R/W</td>
<td>0</td>
<td>Enable Port Mirroring: Set to 1 in order to enable port mirroring (see Sec. 1.9). This bit must be set only after ports have been selected in ECR 5. When this bit is 0, no mirroring is done, regardless of contents in ECR 5</td>
</tr>
<tr>
<td>25</td>
<td>CLRT</td>
<td>R/W</td>
<td>0</td>
<td>Clear Table Enable: Set this bit always to 1. (Setting it to 0, interferes with Current table clearing action; it should never be set to 0 for normal operation of the CXD1700.)</td>
</tr>
<tr>
<td>24:1</td>
<td></td>
<td>R/W</td>
<td>Reserved: On write set to 0; ignore on read</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>LNKOF</td>
<td>RO</td>
<td></td>
<td>Link Overflow: A 1 at this location means writing in the Current Link Table has overflowed address space allotted (that table identified by bits 30:28 of this register). Note: this register should be read before rotating tables (command to ECR8, bit 31)</td>
</tr>
</tbody>
</table>

Switch Engine Register 1 (ECR1)  Aging/Link Table 1 Addresses  Address: 0x01

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:27</td>
<td></td>
<td>WO</td>
<td></td>
<td>Reserved: On write set to 0; ignore on read</td>
</tr>
<tr>
<td>26:21</td>
<td>EOLNK</td>
<td>WO</td>
<td></td>
<td>End of Link Table: These bits are used to detect Current Link table overflow and set bit 0 of ECR0. Write here bits 17:12 of the 24-bit address of the last available entry in all three Link Tables.</td>
</tr>
<tr>
<td>20:7</td>
<td>LNK1</td>
<td>WO</td>
<td>0</td>
<td>Link Table 1: 14 MSB of the 24-bit starting address for Link Table 1 (see Sec. 3.4.1).</td>
</tr>
<tr>
<td>6:0</td>
<td>LKUP1</td>
<td>WO</td>
<td>0</td>
<td>Learning Table 1: 7 MSB of the 24-bit starting address</td>
</tr>
</tbody>
</table>
### Switch Engine Register 2 (ECR2)  
#### Aging/Link Table 2 Addresses  

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 20:7     | LNKT2 | WO     | 0       | **Reserved:** On write set to 0; ignore on read  
          |       |        |          | **Link Table 2:** 14 MSB of the 24-bit starting address for Link Table 2 (see Sec. 3.4.1). |
| 6:0      | LKUP2 | WO     | 0       | **Learning Table 2:** 7 MSB of the 24-bit starting address |

### Switch Engine Register 3 (ECR3)  
#### Aging/Link Table 3 Addresses  

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 20:7     | LNKT3 | WO     | 0       | **Reserved:** On write set to 0; ignore on read  
          |       |        |          | **Link Table 3:** 14 MSB of the 24-bit starting address for Link Table 3 (see Sec. 3.4.1) |
| 6:0      | LKUP2 | WO     | 0       | **Learning Table 3:** 7 MSB of the 24-bit starting address |

### Switch Engine Register 4 (ECR4)  
#### Permanent/VLAN Table Addresses  

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 20:7     | VLANT | WO     | 0       | **Reserved:** On write set to 0; ignore on read  
          |       |        |          | **VLAN Table:** 14 MSB of the 24-bit starting address for VLAN port map table |
| 6:0      | PRMT  | WO     | 0       | **Permanent Table:** 7 MSB of the 24-bit starting address for Permanent table |

### Switch Engine Register 5 (ECR5)  
#### Port Mirroring Setup  

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:27</td>
<td>MIRR</td>
<td>WO</td>
<td>0</td>
<td><strong>Mirroring Receive Port:</strong> 5-bit binary number that defines the port through which the traffic on all ports selected by bits 26:0 of this register will be mirrored (transmitted) to. This can be any one of the network ports (0-25) or CPU port (26).</td>
</tr>
<tr>
<td>26:0</td>
<td>MIRP</td>
<td>WO</td>
<td>0</td>
<td><strong>Port Mirroring Bitmap:</strong> A 27-bit bitmap that determines which ports are mirrored (bit 25 = port 25). A logic 1 in any position causes that port to mirror its traffic to the designated receive port. Except for the bandwidth of the link, there is no limitation on how many ports can be mirrored, including the CPU port.</td>
</tr>
</tbody>
</table>
### Switch Engine Register 7 (ECR7)

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:27</td>
<td></td>
<td></td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>26:0</td>
<td>STFWD</td>
<td>WO</td>
<td>0</td>
<td><strong>Spanning Tree Forwarding:</strong> 27-bit bitmap of ports for which forwarding of packets is allowed. BPDUs are still forwarded to the CPU, in accordance with the VLAN map. For normal operation without spanning tree, all 1s must be written in these bit positions during initialization.</td>
</tr>
</tbody>
</table>

### Switch Engine Register 8 (ECR8)

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>AGET</td>
<td></td>
<td></td>
<td><strong>Age Tables:</strong> Set this bit to 1 in order to age tables. This causes the Current table to be rotated to Empty and all 0s to be written in it (Next becomes Current; Empty becomes Next). During the writing process, ECR0 bit 31 is in logic 1 state (see Sec. 1.3.1)</td>
</tr>
<tr>
<td>30:0</td>
<td></td>
<td></td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
</tbody>
</table>

### Port Register 0 (PCR0)

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RXEN</td>
<td>R/W</td>
<td>0</td>
<td><strong>Receive On/Off Switch:</strong> 1 = This port is enabled to receive packets in normal manner 0 = All incoming packets are ignored</td>
</tr>
<tr>
<td>30</td>
<td>TXEN</td>
<td>R/W</td>
<td>0</td>
<td><strong>Transmit On/Off Switch:</strong> 1 = This port is enabled to queue packets for transmitting in the normal manner 0 = Tx queue is flushed and no further entries are allowed into the Tx queue.</td>
</tr>
<tr>
<td>29</td>
<td>RXCUT</td>
<td>R/W</td>
<td>0</td>
<td><strong>Receive Cut-Thru Enable:</strong> 1 = Received packets can be cut through to a willing transmit port 0-23 (For ports 24 and 25 this bit cannot be set to 1) 0 = No cut-thru allowed for packets received on this port</td>
</tr>
<tr>
<td>28</td>
<td>TXCUT</td>
<td>R/W</td>
<td>0</td>
<td><strong>Transmit Cut-Thru Enable:</strong> 1 = Accept packets for cut-thru when not busy (Tx queue empty and not currently transmitting). Allowed only on ports 0-23; on ports 24 and 25 this bit cannot be set to 1. 0 = Do not cut packets through under any circumstances</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>RO</td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>RO</td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>25</td>
<td>FDPLX</td>
<td>R/W</td>
<td>0</td>
<td><strong>Full/Half Duplex Switch:</strong> 1 = This port is operating in full duplex mode (Note: must also set bit 1 in MACC; see also Sec 4.5) 0 = This port is operating in half duplex mode</td>
</tr>
<tr>
<td>24:23</td>
<td>FCMOD</td>
<td>R/W</td>
<td>00</td>
<td><strong>Flow Control Mode Select:</strong> 00 = No flow control - discard packets when backpressure is applied 01 = Per-Port Collision flow control when in half duplex mode 10 = Per-Packet Collision flow control when in half duplex mode 11 = Full duplex flow control (pause frames per 802.3x) (See Section 1.7)</td>
</tr>
</tbody>
</table>
### Port Register 0 (PCR0) (Continued)

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| 22       | BRCFC | R/W    | 0       | **Broadcast Packet Flow Control Switch:**  
|          |      |        |         | 0 = Broadcast/multicast packets are flow controlled, as determined by bits 24:23 of this register.  
|          |      |        |         | 1 = Backpressure is not issued against a broadcast/multicast packet even when port is congested. Packet is discarded in Tx port. |
| 21       | NOBP | WO     | 0       | **Backpressure On/Off Switch:**  
|          |      |        |         | 1 = No backpressure signal is issued when Tx queue is congested  
|          |      |        |         | 0 = Backpressure is issued when Tx queue is congested.  
|          |      |        |         | Note: can set to 1 in mirroring receive port to prevent interference with mirrored ports. |
| 20:11    | PTMR | RO     | 0       | **Per-Port Collision Timer On:**  
|          |      |        |         | 1 = When port is operating in half duplex, this bit indicates that Per-Port Collision flow control is in effect, until timer set by PRC1, bits 15:0 expires.  
|          |      |        |         | 0 = Port is not under Per-Port Collision flow control regimen. |
| 9        | TPAUS | RO     | 0       | **Port is in Pause:**  
|          |      |        |         | 1 = Transmit port controller is in 802.3x pause state because it has received a pause frame from link partner.  
|          |      |        |         | 0 = Transmit port controller is not in pause state -- can transmit normally |
| 8        | TXCNG | RO     | 0       | **Congested State:**  
|          |      |        |         | 1 = Tx queue is in congested state  
|          |      |        |         | 0 = Tx queue is not in congested state |
| 7:0      | TXCGS | RO     | 0       | **Tx Queue Count:** Number of entries currently in Tx queue |

### Port Register 1 (PCR1)

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td>64</td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>30:24</td>
<td>TXQHI</td>
<td>WO</td>
<td>32</td>
<td><strong>Tx queue high water mark:</strong> the number of packets in Tx queue at which this port goes into congested state and asserts backpressure. (See Sec. 1.7). Range: 0-128</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td>0</td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>22:16</td>
<td>TXQLO</td>
<td>WO</td>
<td>0</td>
<td><strong>Tx queue low water mark:</strong> the number of packets in Tx queue at which this port returns to uncongested state (backpressure removed). (See Sec. 1.7) Range: 0-128</td>
</tr>
</tbody>
</table>
| 15:0     | PTIME | WO     | 0       | **Pause time:** this value determines pause duration for two different flow control techniques. When the port is operating in full duplex, this is pause time value in Control Parameters (data) field of an 802.3x pause frame. When the port is operating in half duplex and using per-port flow control, this value determines the time period during which collisions are caused with incoming packets.  
|          |      |        |         | The time value defined by these bits is in terms of slot times, with a range from 0 to 65,536 (for 10 Mbit ports slot time = 51.2µs; for 100 Mbit ports slot time = 5.12 µs) |
### Port Register 2 (PCR 2)  
Packet Buffer Start Address  
Address: 0x02

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 17:0     | BSTRT | WO     | 0       | **Reserved:** On write set to 0; ignore on read  
**Packet Buffer Start Address:** 18 MSB of 24-bit SDRAM address where packet buffer for this port starts. |

### Port Register 3 (PCR3)  
Packet Buffer End Address  
Address: 0x03

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 17:0     | BEND | WO     | 0       | **Reserved:** On write set to 0; ignore on read  
**Packet Buffer End Address:** 18 MSB of 24-bit SDRAM address where packet buffer for this port ends. |
<table>
<thead>
<tr>
<th>Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:20</td>
<td>R/W</td>
<td></td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>19:18</td>
<td>CLKS2, CLKS</td>
<td>R/W</td>
<td>0,0</td>
<td><strong>MDIO Clock Speed:</strong> Sets the clock rate for MDC line from port 24 and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>port 25: These bits are not used on ports 0-23 (have no effect). 00 =</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.0 MHz  10 = 2.0 MHz  01 = 2.86 MHz  11 = 1.65 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Nominal value:</strong> 10</td>
</tr>
<tr>
<td>17</td>
<td>TXFC</td>
<td>R/W</td>
<td>0</td>
<td><strong>Transmit 802.3x Flow Control:</strong> Setting this bit to 1 enables the MAC to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transmit 802.3x pause frames when port is operating in full duplex mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Note: must also set pause frame duration with bits 15:0 in PCR1.)</td>
</tr>
<tr>
<td>16</td>
<td>RXFC</td>
<td>R/W</td>
<td>0</td>
<td><strong>Receive 802.3x Flow Control:</strong> Setting this bit to 1 enables the MAC to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>receive 802.3x pause frames when port is operating in full duplex mode.</td>
</tr>
<tr>
<td>15</td>
<td>SRST</td>
<td>R/W</td>
<td>0</td>
<td><strong>Software Reset of MAC:</strong> To reset the MAC, write a 1, followed by a 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Normally MAC will be reset on powerup (by the hardware RESET at device pin Y2),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>or by a separate RESET anytime thereafter. However, if during the RESET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transmit clock [TXCLK] from the PHY is not active, perform a separate reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of the MAC by setting this bit to 1. (Separate MAC reset never needs to be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>done on 100 Mbit ports.)</td>
</tr>
<tr>
<td>14</td>
<td>INTLB</td>
<td>R/W</td>
<td>0</td>
<td><strong>MAC Internal Loopback:</strong> Write 0 for normal operation; for loopback self-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>test, set this bit to 1; it causes every packet transmitted from the port to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>be received by the port.</td>
</tr>
<tr>
<td>13:9</td>
<td>R/W</td>
<td></td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>8</td>
<td>PARF</td>
<td>R/W</td>
<td>0</td>
<td><strong>Pass All Received Frames:</strong> Write 0 for normal operation, which blocks the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>passage of 802.3x control frames; if this bit is 1, all such control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>frames will pass on to the Rx FIFO.</td>
</tr>
<tr>
<td>7</td>
<td>PUREP</td>
<td>R/W</td>
<td>0</td>
<td><strong>Pure Preamble:</strong> Write 0 for normal operation; if this bit is 1, any frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>with an error in the preamble is filtered.</td>
</tr>
<tr>
<td>6</td>
<td>R/W</td>
<td></td>
<td>0</td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>5</td>
<td>NOBO</td>
<td>R/W</td>
<td>0</td>
<td><strong>No Backoff After Collision:</strong> Write 0 for normal operation; if this bit is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1, the MAC exponential backoff function is disabled, transmit retry will</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>occur without delay.</td>
</tr>
<tr>
<td>4</td>
<td>R/W</td>
<td></td>
<td>0</td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
<tr>
<td>3</td>
<td>CRCEN</td>
<td>R/W</td>
<td>0</td>
<td><strong>CRC Append Enable:</strong> Write 0 for normal operation; this causes CRC of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transmitted packets to be checked and errors to be recorded in transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>statistics vector, but CRC is not regenerated. For CPU packets only, CRC is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>generated if bit 24 in packet header word 1 is set). If this bit is 1, CRC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is generated for all packets when transmitting.</td>
</tr>
<tr>
<td>2</td>
<td>PADEN</td>
<td>R/W</td>
<td>0</td>
<td><strong>Pad Enable:</strong> Write 0 for normal operation; processing of incoming packets</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>discards all runt packets, therefore this function is normally unnecesssary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this bit is 1, pad bytes are inserted in all packets less than 64 bytes in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>length. For CPU packets, if bit 24 in packet header word 1 is set, pad data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is inserted regardless of the position of this bit.</td>
</tr>
<tr>
<td>1</td>
<td>FULLD</td>
<td>R/W</td>
<td>0</td>
<td><strong>Full Duplex Enable:</strong> Write 1 for full duplex operation; write 0 for half-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>duplex operation. (Note: must also set bit 25 in PCR0; see also Sec 4.5).</td>
</tr>
<tr>
<td>0</td>
<td>HUGEN</td>
<td>R/W</td>
<td>0</td>
<td><strong>Unlimited Frame Length:</strong> If this bit is 1, length of frames is unlimited;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>if this bit is 0, maximum frame length is 1536 bytes, plus offset value in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bits 6:0 of MAC1. Note that the above frame length limitations apply equally</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>to received and transmitted frames.</td>
</tr>
</tbody>
</table>
### MAC Control Register 1 (MAC1)

<table>
<thead>
<tr>
<th>Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:15</td>
<td>Reserved</td>
<td>R/W</td>
<td>0</td>
<td>Reserved: On write set to 0; ignore on read</td>
</tr>
<tr>
<td>14:8</td>
<td>IPGT</td>
<td>R/W</td>
<td>0</td>
<td><strong>Back-to-Back Transmit IPG:</strong> Used to set delay counter in the MAC, to accommodate incremental delays in different PHYs, so as to give total delay equal to standard IPG. Delay counter implements IPG only after a packet transmission from this port. <strong>Nominal value, 10Mb:</strong> 0x52 <strong>Nominal value, 100Mb:</strong> 0x15</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>R/W</td>
<td>0</td>
<td>On write set to 0; ignore on read</td>
</tr>
<tr>
<td>6:0</td>
<td>MAXLN</td>
<td>R/W</td>
<td>0</td>
<td><strong>Frame Length Offset:</strong> If bit 0 in MACC is 0, this value determines maximum length of frames: it is 1536 bytes plus offset value entered here. The range is 0-24; if a value larger than 24 is entered, maximum frame length defaults to 1560. Note that the above frame length limitations apply equally to received and transmitted frames.</td>
</tr>
</tbody>
</table>

### MAC Control Register 2 (MAC2)

<table>
<thead>
<tr>
<th>Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:27</td>
<td>Reserved</td>
<td>R/W</td>
<td>0</td>
<td>Reserved: On write set to 0; ignore on read</td>
</tr>
<tr>
<td>26:20</td>
<td>IPGR1</td>
<td>R/W</td>
<td>0</td>
<td><strong>Transmit IPG Counter 1:</strong> Use this value to set first stage defer counter in the MAC, to accommodate incremental delays in different PHYs, so as to give total delay equal to one half of standard IPG. This delay counter implements that segment of IPG, following receipt of a packet, during which MAC always defers to a transmitting node. <strong>Nominal value, 10Mb:</strong> 0x18 <strong>Nominal value, 100Mb:</strong> 0xA</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>R/W</td>
<td>0</td>
<td>On write set to 0; ignore on read</td>
</tr>
<tr>
<td>18:12</td>
<td>IPGR2</td>
<td>R/W</td>
<td>0</td>
<td><strong>Transmit IPG Counter 2:</strong> Use this value to set delay counter in the MAC, to accommodate incremental delays in different PHYs, so as to give total delay equal to standard IPG. (IPGR2 - IPGR1 implements that segment of IPG, following receipt of a packet, during which the MAC, with a queued packet, is committed to transmit regardless of CRS status.) <strong>Nominal value, 10Mb:</strong> 0x30 <strong>Nominal value, 100Mb:</strong> 0x15</td>
</tr>
<tr>
<td>11:10</td>
<td>Reserved</td>
<td>R/W</td>
<td>0</td>
<td>On write set to 0; ignore on read</td>
</tr>
<tr>
<td>9:4</td>
<td>LCOL</td>
<td>R/W</td>
<td>0</td>
<td><strong>Collision Window:</strong> These bits define the collision window recognized by the transmit side of the MAC. Any collision detected outside this window is a late collision and causes the packet just transmitted (or being currently transmitted) to be aborted (i.e. transmit is not retried). The value is defined in byte times and its range is 0 to 0x3F. <strong>Nominal value:</strong> 0x37 (56 bytes) for all ports.</td>
</tr>
<tr>
<td>3:0</td>
<td>RETRY</td>
<td>R/W</td>
<td>0</td>
<td><strong>Maximum Retry Count:</strong> Sets the number of times a retransmission of a packet is attempted after a collision. <strong>Nominal value:</strong> 0xF</td>
</tr>
</tbody>
</table>
### MAC Control Register LSA1 (LSA1)  
**Station Address 1**  
**Address:** 0x19

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>LSA1</td>
<td></td>
<td></td>
<td><strong>Station address</strong>, bits 31:0. (This station address is used by MAC for all 802.3x control frames).</td>
</tr>
</tbody>
</table>

### MAC Control Register LSA2 (LSA2)  
**Station Address 1**  
**Address:** 0x1A

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>LSA2</td>
<td></td>
<td></td>
<td><strong>Station address</strong>, bits 47:32 (See note in table for LSA1)</td>
</tr>
<tr>
<td>15:0</td>
<td></td>
<td></td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
</tbody>
</table>

### MII Command Register (MCMD)  
**MII Command**  
**Address:** 0x1B

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td></td>
<td></td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
</tbody>
</table>
| 1        | SCAN | RO     | 0       | **Link Fail Scan:** To execute a continuous read of the Link Fail bit in a PHY, set this bit to 1. (Note: FIAD and RGAD in MADR register must be defined first, before writing to this bit.) Reading of the Link Fail bit is continued, until this bit is reset to 0 (no use of the resulting data stream is made use of in the CXD1700).  
**Read Command:** To execute a read of a PHY register, write a 1 to this bit; note that reading of the PHY register is triggered by rising edge of this 1 bit (FIAD and RGAD in MADR register must be defined first, before writing here). For a write to a PHY register, set this bit to 0. (See Sec. 4.4 for PHY read/write procedures.) |
| 0        | RSTAT| RO     | 0       | **Reserved:** On write set to 0; ignore on read |

### MII Address Register (MADR)  
**Device Address**  
**Address:** 0x1C

<table>
<thead>
<tr>
<th>Bit Pos.</th>
<th>Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:13</td>
<td></td>
<td></td>
<td></td>
<td><strong>Reserved:</strong> On write set to 0; ignore on read</td>
</tr>
</tbody>
</table>
| 12:8     | FIAD | R/W    |         | **Physical Address:** Use these bits to select one of up to 32 PHY devices to access. A 5-bit binary encoded address, used for read, write, or link fail bit scan.  
**Reserved:** On write set to 0; ignore on read |
| 7:5      |      | R/W    |         | **Register Address:** Use these bits to access any one of the registers inside the PHY. A 5-bit binary encoded address, used for read, write, or link fail bit scan. |
| 4:0      | RGAD | R/W    |         | **Reserved:** On write set to 0; ignore on read |
### MII Write Data Register (MWTD)

<table>
<thead>
<tr>
<th>Pos.</th>
<th>Bit Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved: On write set to 0; ignore on read</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:0</td>
<td>CTLD</td>
<td>R/W</td>
<td></td>
<td>Write Data: Place the data to be written into the PHY register here; note that writing to this location triggers the writing of the data to the PHY register. (FIAD and RGAD values in register MADR and a 0 in RSTAT position of MCMD must be written first.) (See Sec. 4.4 for PHY read/write procedures.)</td>
</tr>
</tbody>
</table>

### MII Read Data Register (MRDD)

<table>
<thead>
<tr>
<th>Pos.</th>
<th>Bit Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved: On write set to 0; ignore on read</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:0</td>
<td>PRSD</td>
<td>RO</td>
<td></td>
<td>Read Data Result: The data read out of a PHY register, as a result of a read command, is placed in this location.</td>
</tr>
</tbody>
</table>

### MII Indicators Register (MIND)

<table>
<thead>
<tr>
<th>Pos.</th>
<th>Bit Name</th>
<th>Access</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td>Reserved: On write set to 0; ignore on read</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SCAN</td>
<td>RO</td>
<td></td>
<td>Link Fail Scan Result: the status of the Link Fail bit, as a result of the last scan in the selected PHY, is written here. (1 typically indicates the link is up, however, refer to the PHY documentation to determine the sense of this bit)</td>
</tr>
<tr>
<td>0</td>
<td>BUSY</td>
<td>RO</td>
<td></td>
<td>Busy: When this bit is a 1, the MII interface is in the process of executing a read, write, or is testing the Link Fail bit in the PHY.</td>
</tr>
</tbody>
</table>

### 4.3 Initialization

This section contains information for developing a procedure for initializing the CXD1700 following powerup, or following a hardware RESET. The steps described here are not a complete initialization procedure, nor do they represent a specific required sequence. Rather, they constitute a model for identifying the main tasks that must be performed to bring the CXD1700 to an operational status. Note that if, for example, VLAN is to be used, or if port mirroring is to be done, additional setup is required (refer to Sections 4.8 and 4.9). In general, the programmer can use and expand the information in this section to design and develop a specific initialization procedure for a particular application.

1. **Load switch engine registers ECR0 through ECR8.** ECR0 bit 25 should always be set to 1, otherwise it will interfere with normal clearing of the Empty learning table. In the Link Match bits location(26:21 of ECR1) write bits 17:12 of the 24-bit address of the last location in all three Link Tables. Starting addresses in ECR1 through ECR4 must agree with the starting addresses in the SDRAM memory map, earlier in this section. The learning and forwarding bits in ECR6 and ECR7 should be set to 1 on all ports that are utilized. If the spanning tree algorithm is implemented, altering these bits during operation will be done by the algorithm, as it develops and implements the active topology of the network.

2. **Load MAC Registers in All Ports and MIIM Registers in 100 Mbit Ports.** There are nine MAC registers, two LSA registers in each of the 26 ports. The 100 Mbit ports each has five MIIM registers. All these registers must be initialized one port at a time, therefore writing must be repeated 26 times. The following are suggested initial settings for the MAC registers. Read and observe the comments following each load value.

MACC, 10 Mbit ports: 0x00000000
MACC, 100 Mbit ports: 0x000B0000

The CXD1700 system clock is 40 MHz; this requires that bits 19:18 of port 24 and 25 MACC be set to 10. This value determines the MDC clock frequency to be about 2 MHz and thus enables the CXD1700 to communicate with manageable PHYs (refer to step 3 below). (Also note: if TXCLK from the PHY connected to the port is not active during the hardware reset, do a software reset of the MAC with bit 15 of MACC -- see MACC Table for instructions. This applies to ports 0-23 only.)

MAC1, 10 Mbit ports: 0x00005200
MAC1, 100 Mbit ports: 0x00001500

This sets the interpacket gap time between sequentially transmitted frames to a standard 9.6 µsec on 10 Mbit ports and 0.96 µsec on 100 Mbit ports.

MAC2, 10 Mbit ports: 0x0183037F
MAC2, 100 Mbit ports: 0x00A1537F

This sets the interpacket gap time following a received frame to a standard 9.6 µsec on 10 Mbit ports and 0.96 µsec on 100 Mbit ports; it sets the first stage defer time to one half of the total IPG time. The collision window is set to one slot time (512 bits) and the transmit retry count following a collision is set to 15. Each of the above is either a spec value, or a common practice value.

MAC3 through MAC8, all ports: 0x00000000
These are test registers for CXD1700 internal functions and are not to be written during initialization or read at all other times.

LSA1 and LSA2, all ports: 48-bit station MAC address.
This is the address used for 802.3x pause frames.

MCMD, MADR, MWTD, MRDD, MIND, ports 24 and 25 0x00000000
This clears the registers in preparation for communicating with manageable PHYs.

3. Read Autonegotiation Results. If manageable PHYs are used on any of the ports and the PHYs are enabled to autonegotiate, the autonegotiation occurs on powerup, or whenever the connection to a link partner goes active. The results are stored in the Autonegotiation Link Partner Ability register of the PHY and the PHY then automatically sets its own operating mode. The CPU must read the results from the PHY register and set up PCR0 and MACC registers to match the operating mode of the PHY (on powerup all CXD1700 ports default to half duplex; procedure for changing to full duplex is in Section 4.5). If autonegotiation is disabled or the PHYs are forced into any mode, reading of the Autonegotiation Link Partner Ability register does not need to be performed.

4. Clear Learning Tables. Before turning on ports and processing packets, the three learning tables must be cleared (all 0s written into them). This is done by writing a 1 in ECR8 bit 31 and then monitoring bit 31 of ECR0. When bit 31 or ECR0 returns from 1 to 0, it indicates an learning table is cleared. Repeat this procedure three times to clear all three learning tables.

5. Load PCR Registers of All Ports. There are four PCR registers in each of the 26 ports. These must be initialized one port at a time, therefore writing must be repeated 26 times. It is suggested that PCR registers are written last, so that all other setup is complete before the ports are turned on and the CXD1700 is opened to packet traffic. The following are typical but not required initial settings for the PCR registers.

PCR0, 10 Mbit ports: 0xF9000000
PCR0, 100 Mbit ports: 0xC9000000
This enables the receive and transmit functions of the port, enables cut-thru in the receive and transmit controllers of 10 Mbit ports, but disables it on 100 Mbit ports. The above values place ports in half duplex operation, and enables Per-Packet Collision flow control.

PCR1, all ports: Values in this register are different in different applications and depend on the size of the SDRAM and the desired congestion limits in the Tx queue. Refer to Section 4.6 for suggestions on setting up flow control.

PCR2 and PCR3, all ports: Enter packet buffer starting and ending addresses that agree with the SDRAM memory map.
4.4 Reading/Writing PHY Registers and Link Status Bit

The MII registers are used for reading and writing registers in manageable PHY devices equipped with standard MDIO interface and registers in accordance with IEEE 802.3u. As described in Section 2.5, manageable PHYs can be used on any or all 26 of the CXD1700 ports by daisychaining them on the MDIO lines from port 24 and 25. The PHY devices each must be assigned a unique physical address. Use the following procedures to execute PHY read and write operations.

To do a read of a PHY register, do the first two steps with one write operation (refer to Section 4.2 for CXD1700 register write instructions):

1. Write the 5-bit physical PHY address FIAD in MADR register.
2. Write the 5-bit PHY register address RGAD in MADR register.

Do step 3. with a second write operation:

3. Write a logic 1 in command register MCMD bit position 0 (RSTAT). A rising edge on this register bit causes the read operation to be executed. Monitor the BUSY bit in indicators register MIND to determine when the read operation is complete.
4. Result is loaded into MRDD results register, where it can be read by the CPU.

To do a write, do the first two steps with one write operation:

1. Write the 5-bit physical PHY address FIAD in MADR register.
2. Write the 5-bit PHY register address RGAD in MADR register.

Do step 3. with a second write operation:

3. Write the 16-bit data word (CTLD) to be written into the PHY register into MWTD, bit positions 15:0. The rising edge of the load strobe to this register causes the data to also be written into the specified PHY register.
4. Monitor the BUSY bit in indicators register MIND to determine when the write operation is complete.

4.5 Activating Full Duplex Operation

By default CXD1700 comes up after RESET in half duplex on all ports. Any given port may need to be switched to full duplex data communication mode during initialization, as a result of autonegotiation when a link goes active, or because a PHY connected to the port is forced into full duplex by hardwiring. A manageable PHY may also be manually forced to full duplex through the MDIO management interface (refer to Section 2.5.2). If a PHY comes up in full duplex, as a result of any of the foregoing reasons, the PCR and MAC registers must be verified and rewritten as needed to activate the full duplex circuits:

- PCR0 bit 25 — set to 1 This enables the port receive and transmit controllers to operate in full duplex
- PCR0 bits 24:23 — set to 1,1 This changes flow control mode from half to full duplex
- MACC bit 1 — set to 1 This enables the MAC to operate in full duplex
- MACC bit 16 — set to 1 This enables the MAC to receive 802.3x pause frames
- MACC bit 17 — set to 1 This enables the MAC to send out 802.3x pause frames

(Note: it is a good practice to set MACC bits 17:16 to 1 during initialization, regardless of whether the port is to operate in half or full duplex; this setting does not interfere with half duplex operation.)

In addition to the above bits, the flow control parameters should be verified to make sure they match the data communication mode (see Section 4.6)

4.6 Setting Up Flow Control

Flow control is set up for each port individually in PCR0 and PCR1. After setting up backpressure parameters in PCR1, the rest of setup is different, depending on whether the port is operating in half or full duplex.

**Backpressure Parameters:** Set high water mark with bits 30:24 of PCR1. Set low water mark with bits 22:16 of PCR1. Verify that PCR0 bit 21 (backpressure on/off) is set to 0 and bit 22 to desired position. One suggested technique is to set high water mark count at about half of the capacity of the packet buffer, measured in the number of maximum size packets (see Section 4.1). Then, set the low water mark just enough below the high water mark to provide some hysteresis and get smooth operation of the flow control mechanism.

**Half duplex:** Choose the flow control technique with PCR0 bits 24:23. If you choose Per-Port flow control, set the flow control time duration with PCR1 bits 15:0. If you choose Per-Packet flow control, the timer is not involved.

**Full duplex:** Choose the full duplex flow control by setting PCR0 bits 24:23 to 1,1. Set also MACC bits 17:18 to 1,1. Set pause duration with PCR1 bits 15:0.

Note that PCR0 bits 10:0 report the status of congestion, as well as the status of flow control activity in the port.
4.7 Setting Up Collision Window
Any collision detected outside the collision window defined by bits 9:4 of MAC2 register causes the packet being transmitted to be aborted and a late collision event to be registered in the Tx status vector. By definition, a collision window equals one slot time, or 512 bit times. This represents a value of 0x37 in bits 9:4 of MAC2 register, but these bits allow the collision window to be adjusted. The bits are intended only for fine tuning the collision window in the transmit side of the MAC, not for materially extending it.

4.8 Setting Up VLAN
To set up a VLAN, the CPU must write initial values and thereafter manage all three VLAN related tables. All MAC addresses belonging to a VLAN must be written into the Permanent table and be assigned the same index into the VLAN table. The VLAN table must be written with the bit map, listing all ports belonging to that VLAN. For this, the CPU must also implement the linking algorithm that allows MAC addresses with identical lower 16 bits to be linked from the Permanent table into the Permanent Link table. Refer to Section 3.5 for a description of the CXD1700 VLAN mechanism and data formats for the tables.

Any multicast/broadcast packet whose source address is not found in the Permanent table will be forwarded to the CPU for resolution. If a source address-to-VLAN listing is made available to the CPU through network management channels, the CPU can respond by updating the Permanent and VLAN tables and thus learn a new MAC address into the tables. The CPU can then resend the packet and it will be forwarded to the proper CXD1700 ports in the normal manner.

Similar techniques can be used to implement a port-based VLAN. New MAC addresses can be learned into the Permanent and VLAN tables in the same manner. If a port-to-VLAN listing is available to the CPU, it can respond to any packet forwarded to it for resolution by identifying the port on which the packet came in, writing the corresponding VLAN index into the Permanent table, and the port number in the VLAN table map. The CPU can then resending the packet in the normal manner. In a port-based VLAN implementation, every MAC address that comes in on any port assigned to a particular VLAN is entered into the Permanent table with the same VLAN index.

4.9 Setting Up Mirroring
Mirroring is set up in ECR5 and then is turned on or off with bit 26 of ECR0. One or more ports, up to 25, can be mirrored to any other port. When choosing the ports to be mirrored and the port to receive the mirrored data, make sure that the receiving port has enough bandwidth to handle the sum of the expected traffic load. Otherwise, the receiving port will become congested, generate backpressure, and thereby interfere with the traffic that is to be mirrored. Whatever port is chosen as the receiving port, a better practice may be to set its PCR0 bit 21 to 0, so that the port does not generate backpressure and interfere with the traffic to be monitored.

One possible port assignment technique may be to mirror multiple 10 Mbit ports to a 100 Mbit port.

4.10 Setting Up Spanning Tree Facilities
The spanning tree algorithm utilizes ECR6, ECR7, and ECR8. The first two registers allow the spanning tree algorithm to execute its learning sequences and implement the spanning tree. In the absence of a spanning tree implementation, the aging of tables (writing a 1 to bit 31 of ECR8) is nominally done every 5 minutes, or any other value programmed in the CPU. If the spanning tree algorithm is used, it controls the aging intervals, as necessary to support its learning activity. It may age the tables as often as every 15 seconds.
## 5.0 Hardware Information

### CXD1700 Pinout Summary

<table>
<thead>
<tr>
<th>Label</th>
<th>I/O</th>
<th>Description</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>10 MBPS Pin Descriptions:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXC 0-23</td>
<td>Input</td>
<td>Receive Clock</td>
<td>( V_{IL} = 0.8V )</td>
<td></td>
</tr>
<tr>
<td>RXD 0-23</td>
<td>Input</td>
<td>Receive Data</td>
<td>( V_{IH} = 2.0V )</td>
<td></td>
</tr>
<tr>
<td>CRS 0-23</td>
<td>Input</td>
<td>Carrier Sense Detect</td>
<td>For ( V_{CC} = 3.3 \pm 0.3V )</td>
<td></td>
</tr>
<tr>
<td>COL 0-23</td>
<td>Input</td>
<td>Collision</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXC 0-5</td>
<td>Input</td>
<td>Transmit Clock, four ports to one clock lead</td>
<td></td>
<td>( V_{OH} = 2.4V \pm 300 \mu A )</td>
</tr>
<tr>
<td>TXD 0-23</td>
<td>Output</td>
<td>Transmit data</td>
<td></td>
<td>( V_{OL} = 0.4V \pm 2 mA )</td>
</tr>
<tr>
<td>TX_EN 0-23</td>
<td>Output</td>
<td>Transmit Data Enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MII Pin Descriptions:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXCLK 24-25</td>
<td>Input</td>
<td>Transmit nibble or symbol clock</td>
<td>( V_{IL} = 0.8V )</td>
<td></td>
</tr>
<tr>
<td>TX_ER 24-25</td>
<td>Input</td>
<td>Transmit Error</td>
<td>( V_{IH} = 2.0V )</td>
<td></td>
</tr>
<tr>
<td>MDC 24-25</td>
<td>Output</td>
<td>Management Data Clock</td>
<td>For ( V_{CC} = 3.3 \pm 0.3V )</td>
<td></td>
</tr>
<tr>
<td>MDIO 24-25</td>
<td>I/O</td>
<td>Management Data Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRS 24-25</td>
<td>Input</td>
<td>Carrier Sense</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RX_DV 24-25</td>
<td>Input</td>
<td>Receive Data Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COL 24-25</td>
<td>Input</td>
<td>Collision</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RX_ER 24-25</td>
<td>Output</td>
<td>Receive Error</td>
<td></td>
<td>( V_{OH} = 2.4V \pm 300 \mu A )</td>
</tr>
<tr>
<td>RXD 24-25</td>
<td>Input</td>
<td>Receive Data</td>
<td></td>
<td>( V_{OL} = 0.4V \pm 2 mA )</td>
</tr>
<tr>
<td>RXC 24-25</td>
<td>Input</td>
<td>Receive Clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXD 24-25</td>
<td>Output</td>
<td>Transmit Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX_EN 24-25</td>
<td>Output</td>
<td>Transmit Enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Switch Pin Descriptions:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>Chip Clock at 40 MHz</td>
<td>( V_{IL} = 0.8V )</td>
<td></td>
</tr>
<tr>
<td>RESET#</td>
<td>Input</td>
<td>Chip Initialization</td>
<td>( V_{IH} = 2.0V )</td>
<td></td>
</tr>
<tr>
<td>BREQ_</td>
<td>Input</td>
<td>Bus Request</td>
<td>For ( V_{CC} = 3.3 \pm 0.3V )</td>
<td></td>
</tr>
<tr>
<td>CMDREADY</td>
<td>Input</td>
<td>Mail Call MPU to Switch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGR_T</td>
<td>Output</td>
<td>Bus Grant</td>
<td></td>
<td>( V_{OH} = 2.4V \pm 300 \mu A )</td>
</tr>
<tr>
<td>HDR_STRB</td>
<td>Output</td>
<td>Header Strobe</td>
<td></td>
<td>( V_{OL} = 0.4V \pm 2 mA )</td>
</tr>
<tr>
<td>TASKDONE</td>
<td>Output</td>
<td>Task Finished Switch to CPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ES_CLK</td>
<td>Output</td>
<td>Etherstat Strobe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ES_DATA 3:0</td>
<td>Output</td>
<td>Etherstat Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XCLK</td>
<td>Output</td>
<td>Clock to SDRAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMADDH 3:0</td>
<td>Output</td>
<td>High order memory address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMADDL 11:0</td>
<td>Output</td>
<td>Memory addressing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMICLE</td>
<td>Output</td>
<td>Clock enable</td>
<td>( V_{IL} = 0.8V )</td>
<td></td>
</tr>
<tr>
<td>XMCS_</td>
<td>Output</td>
<td>Memory commands</td>
<td>( V_{IH} = 2.0V )</td>
<td></td>
</tr>
<tr>
<td>XMCAS_</td>
<td>Output</td>
<td>Memory commands</td>
<td>( V_{OH} = 2.4V \pm 2 mA )</td>
<td></td>
</tr>
<tr>
<td>XMRAS_</td>
<td>Output</td>
<td>Memory commands</td>
<td>( V_{OL} = 0.4V \pm 2 mA )</td>
<td></td>
</tr>
<tr>
<td>XMWE_</td>
<td>Output</td>
<td>Memory commands</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMLDQM</td>
<td>Output</td>
<td>Lower byte mask write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMUDQM</td>
<td>Output</td>
<td>Upper byte mask write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XDATA 31:0</td>
<td>I/O</td>
<td>Data bus to SDRAM</td>
<td>( X V_{IL} = 0.8V )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( X V_{IH} = 2.0V )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( Y V_{OH} = 2.4V \pm 2 mA )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( Y V_{OL} = 0.4V \pm 2 mA )</td>
<td></td>
</tr>
</tbody>
</table>
Taped Grid Array Pin Designations & Pinout
<table>
<thead>
<tr>
<th>BALL</th>
<th>SIGNAL</th>
<th>BALL</th>
<th>SIGNAL</th>
<th>BALL</th>
<th>SIGNAL</th>
<th>BALL</th>
<th>SIGNAL</th>
</tr>
</thead>
</table>
| A1   | VSSP   | AC18 | txd2   | D12  | tx_en18| H23  | txd25_1_
| A10  | txd18  | AC19 | rxd2   | AE17 | VSSP   | H24  | txd25_0_
| A11  | crs18  | AC2  | VSSL   | AE18 | crs3   | B15  | col16  
| A12  | rxc18  | AC20 | rxc1   | AE19 | tx_en3 | B16  | rxc16  
| A13  | VDDL_7 | AC21 | VSSP   | AE2  | VSSP   | B17  | rxd15  
| A14  | rxd17  | AC22 | crs0   | AE20 | col2   | B18  | tx_en15 
| A15  | tx_en17| AC23 | rxd0   | AE21 | VDDP_1 | B19  | txd15  
| A16  | txd17  | AC24 | VSSL   | AE22 | rxd1   | B2   | rxc23  
| A17  | crs16  | AC25 | col24  | AE23 | tx_en1 | B20  | rcs14  
| A18  | rxd16  | AC26 | txd24_2_| AE24 | tx_en0 | B21  | txc12_15
| A19  | rxc15  | AC3  | VSSL   | AE25 | rxc0   | B22  | rxd13  
| A2   | txd23  | AC4  | VSSP   | AE26 | rxd24_2_| B23  | VSSP   
| A20  | VSSP   | AC5  | col11  | AE3  | tx_en11| B24  | tx_en12 
| A21  | col14  | AC6  | tx_en10| AE4  | col10  | B25  | VSSP   
| A22  | rxd14  | AC7  | VSSP   | AE5  | rxd10  | B26  | VSSL   
| A23  | VSSL   | AC8  | VSSL   | AE6  | rxc9   | B3   | tx_en23 
| A24  | col13  | AC9  | txd8   | AE7  | tx_en9 | B4   | col22  
| A25  | txd12  | AD1  | VSSP   | AE8  | tx_en8 | B5   | txc20_23
| A26  | rxc12  | AD10 | rxd8   | AE9  | crs8   | B6   | crs21  
| A3   | crs22  | AD11 | crs7   | AF1  | rxc11  | B7   | txd20  
| A4   | rxd21  | AD12 | tx_en6 | AF10 | tx_en7 | B8   | col20  
| A5   | tx_en21| AD13 | rxd6   | AF11 | col6   | B9   | rxc20  
| A6   | VSSP   | AD14 | rxd5   | AF12 | crs6   | C1   | VSSP   
| A7   | crs20  | AD15 | txd5   | AF13 | txc4_7 | C10  | rxd19  
| A8   | VDDL_8 | AD16 | rxc4   | AF14 | VDDL_3 | C11  | tx_en19 
| A9   | crs19  | AD17 | rxd3   | AF15 | col5   | C12  | col18  
| AA1  | es_clk | AD18 | txd3   | AF16 | tx_en5 | C13  | txc16_19
| AA2  | VSSL   | AD19 | crs2   | AF17 | col4   | C14  | rxc17  
| AA23 | rxd24_0_| AD2  | VSSL   | AF18 | VDDP_2 | C15  | txd16  
| AA24 | VDDL_9 | AD20 | txc0_3 | AF19 | rxd4   | C16  | VDDL_6 
| AA25 | VDDP_9 | AD21 | crs1   | AF2  | txd11  | C17  | crs15  
| AA26 | rxc24  | AD22 | txd1   | AF20 | col3   | C18  | txd14  
| AA3  | VSSP   | AD23 | col0   | AF21 | tx_en2 | C19  | rxc14  
| AA4  | VDDL_13| AD24 | VSSP   | AF22 | rxc2   | C2   | VSSP   
| AB1  | VSSL   | AD25 | rxd24_3_ | AF23 | VDDL_1 | C20  | VDDL_5 
| AB2  | VSSP   | AD26 | crs24  | AF24 | col1   | C21  | crs13  
| AB23 | VSSL   | AD3  | rxd11  | AF25 | txd0   | C22  | txd13  
| AB24 | rxd24_1_| AD4  | col11  | AF26 | VSSP   | C23  | col12  
| AB25 | txd24_3_| AD5  | txd10  | AF3  | crs10  | C24  | rxc12  
| AB26 | tx_en24| AD6  | rxc10  | AF4  | txc8_11| C25  | VSSL   
| AB3  | VSSL   | AD7  | rxd9   | AF5  | crs9   | C26  | rxd25_1_
| AB4  | VSSP   | AD8  | col9   | AF6  | txd9   | C3   | VSSL   
| AC1  | VSSP   | AD9  | VDDP_4 | AF7  | col8   | C4   | col23  
| AC10 | rxc8   | AE1  | VSSL   | AF8  | rxd7   | C5   | txd22  
| AC11 | rxc7   | AE10 | VDDL_4 | AF9  | VSSL   | C6   | rxc22  
| AC12 | txd6   | AE11 | col7   | B1   | VSSL   | C7   | rxc21  
| AC13 | VDDP_3 | AE12 | txd7   | B10  | rxc19  | C8   | txd21  
| AC14 | rxc5   | AE13 | rxc6   | B11  | txd19  | C9   | tx_en20
<table>
<thead>
<tr>
<th>BALL</th>
<th>SIGNAL</th>
<th>BALL</th>
<th>SIGNAL</th>
<th>BALL</th>
<th>SIGNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>D13</td>
<td>rxd18</td>
<td>H3</td>
<td>VDDP_20</td>
<td>R1</td>
<td>VDDP_17</td>
</tr>
<tr>
<td>D14</td>
<td>crs17</td>
<td>H4</td>
<td>VDDP_21</td>
<td>R2</td>
<td>xmalloc_2_</td>
</tr>
<tr>
<td>D15</td>
<td>tx_en16</td>
<td>J1</td>
<td>VDDP_19</td>
<td>R23</td>
<td>xdata_10_</td>
</tr>
<tr>
<td>D16</td>
<td>VDDP_6</td>
<td>J2</td>
<td>breq_</td>
<td>R24</td>
<td>xdata_11_</td>
</tr>
<tr>
<td>D17</td>
<td>col15</td>
<td>J23</td>
<td>VDDL_12</td>
<td>R25</td>
<td>xdata_9_</td>
</tr>
<tr>
<td>D18</td>
<td>tx_en14</td>
<td>J24</td>
<td>rxc25</td>
<td>R26</td>
<td>xdata_13_</td>
</tr>
<tr>
<td>D19</td>
<td>VDDP_5</td>
<td>J25</td>
<td>rxc25</td>
<td>R3</td>
<td>xmalloc_4_</td>
</tr>
<tr>
<td>D2</td>
<td>VSSL</td>
<td>J26</td>
<td>xdata_28_</td>
<td>R4</td>
<td>xmalloc_3_</td>
</tr>
<tr>
<td>D20</td>
<td>rxc13</td>
<td>J3</td>
<td>hdr_strb</td>
<td>T1</td>
<td>xmalloc_5_</td>
</tr>
<tr>
<td>D21</td>
<td>tx_en13</td>
<td>J4</td>
<td>VDDP_19</td>
<td>T2</td>
<td>xmalloc_0_</td>
</tr>
<tr>
<td>D22</td>
<td>crs12</td>
<td>K1</td>
<td>xmcwe_</td>
<td>T23</td>
<td>xdata_5_</td>
</tr>
<tr>
<td>D23</td>
<td>VSSP</td>
<td>K2</td>
<td>VSSL</td>
<td>T24</td>
<td>xdata_6_</td>
</tr>
<tr>
<td>D24</td>
<td>VSSP</td>
<td>K23</td>
<td>mdio25</td>
<td>T25</td>
<td>xdata_7_</td>
</tr>
<tr>
<td>D25</td>
<td>rxc25</td>
<td>K24</td>
<td>xdata_31_</td>
<td>T26</td>
<td>xdata_12_</td>
</tr>
<tr>
<td>D26</td>
<td>txc25</td>
<td>K25</td>
<td>xdata_29_</td>
<td>T3</td>
<td>VDDL_16</td>
</tr>
<tr>
<td>D3</td>
<td>VSSL</td>
<td>K26</td>
<td>xdata_26_</td>
<td>T4</td>
<td>VDDL_16</td>
</tr>
<tr>
<td>D4</td>
<td>rxd23</td>
<td>K3</td>
<td>clk</td>
<td>U1</td>
<td>xmalloc_1_</td>
</tr>
<tr>
<td>D5</td>
<td>crs23</td>
<td>K4</td>
<td>bgrt_</td>
<td>U2</td>
<td>VDDL_15</td>
</tr>
<tr>
<td>D6</td>
<td>tx_en22</td>
<td>L1</td>
<td>xmcle</td>
<td>U23</td>
<td>VDDP_10</td>
</tr>
<tr>
<td>D7</td>
<td>rxd22</td>
<td>L2</td>
<td>xmudqm</td>
<td>U24</td>
<td>xdata_2_</td>
</tr>
<tr>
<td>D8</td>
<td>col21</td>
<td>L23</td>
<td>VDDP_12</td>
<td>U25</td>
<td>VDDL_10</td>
</tr>
<tr>
<td>D9</td>
<td>VDDP_8</td>
<td>L24</td>
<td>VDDL_11</td>
<td>U26</td>
<td>xdata_8_</td>
</tr>
<tr>
<td>E1</td>
<td>VSSP</td>
<td>L25</td>
<td>xdata_27_</td>
<td>U3</td>
<td>xmalloc_2_</td>
</tr>
<tr>
<td>E2</td>
<td>VSSP</td>
<td>L26</td>
<td>xdata_22_</td>
<td>U4</td>
<td>xmalloc_1_</td>
</tr>
<tr>
<td>E23</td>
<td>VSSP</td>
<td>L3</td>
<td>xmlqdm</td>
<td>V1</td>
<td>VSSP</td>
</tr>
<tr>
<td>E24</td>
<td>VSSL</td>
<td>L4</td>
<td>VDDL_18</td>
<td>V2</td>
<td>xmalloc_0_</td>
</tr>
<tr>
<td>E25</td>
<td>crs25</td>
<td>M1</td>
<td>VSSP</td>
<td>V23</td>
<td>tx_en24</td>
</tr>
<tr>
<td>E26</td>
<td>VDDP_13</td>
<td>M2</td>
<td>xmcas_</td>
<td>V24</td>
<td>rx_en24</td>
</tr>
<tr>
<td>E3</td>
<td>VSSL</td>
<td>M23</td>
<td>xdata_24_</td>
<td>V25</td>
<td>xdata_1_</td>
</tr>
<tr>
<td>E4</td>
<td>VSSP</td>
<td>M24</td>
<td>xdata_23_</td>
<td>V26</td>
<td>xdata_4_</td>
</tr>
<tr>
<td>F1</td>
<td>VSSL</td>
<td>M25</td>
<td>xdata_25_</td>
<td>V3</td>
<td>es_data_1_</td>
</tr>
<tr>
<td>F2</td>
<td>VSSL</td>
<td>M26</td>
<td>xdata_21_</td>
<td>V4</td>
<td>es_data_0_</td>
</tr>
<tr>
<td>F23</td>
<td>rxc25</td>
<td>M3</td>
<td>xmcas_</td>
<td>W1</td>
<td>xmalloc_3_</td>
</tr>
<tr>
<td>F24</td>
<td>rxc25_0</td>
<td>M4</td>
<td>xmras_</td>
<td>W2</td>
<td>es_data_2_</td>
</tr>
<tr>
<td>F25</td>
<td>txc25</td>
<td>N1</td>
<td>xmalloc_10_</td>
<td>W23</td>
<td>txc24_0_</td>
</tr>
<tr>
<td>F26</td>
<td>mdc25</td>
<td>N2</td>
<td>VDDP_18</td>
<td>W24</td>
<td>mdc24</td>
</tr>
<tr>
<td>F3</td>
<td>VSSP</td>
<td>N23</td>
<td>xdata_19_</td>
<td>W25</td>
<td>mdio24</td>
</tr>
<tr>
<td>F4</td>
<td>VSSL</td>
<td>N24</td>
<td>VDDP_11</td>
<td>W26</td>
<td>xdata_3_</td>
</tr>
<tr>
<td>G1</td>
<td>tasksstompup</td>
<td>N25</td>
<td>xdata_20_</td>
<td>W3</td>
<td>VDDP_15</td>
</tr>
<tr>
<td>G2</td>
<td>VSSP</td>
<td>N26</td>
<td>xdata_18_</td>
<td>W4</td>
<td>VDDP_14</td>
</tr>
<tr>
<td>G23</td>
<td>col25</td>
<td>N3</td>
<td>xmalloc_11_</td>
<td>Y1</td>
<td>es_data_3_</td>
</tr>
<tr>
<td>G24</td>
<td>txc25</td>
<td>N4</td>
<td>VDDL_17</td>
<td>Y2</td>
<td>reset_</td>
</tr>
<tr>
<td>G25</td>
<td>tx_en25</td>
<td>P1</td>
<td>xmalloc_9_</td>
<td>Y23</td>
<td>txc24</td>
</tr>
<tr>
<td>G26</td>
<td>tx_en25</td>
<td>P2</td>
<td>xmalloc_6_</td>
<td>Y24</td>
<td>txc24_1_</td>
</tr>
<tr>
<td>G3</td>
<td>VSSL</td>
<td>P23</td>
<td>xdata_15_</td>
<td>Y25</td>
<td>txc24</td>
</tr>
</tbody>
</table>
Electrical Specifications

Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Item</th>
<th>Symbol</th>
<th>Rating</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>VCC</td>
<td>-1.0 to +5 V</td>
<td>V</td>
</tr>
<tr>
<td>Input Voltage</td>
<td>VIN</td>
<td>-0.5 to + VCC + 0.5 V</td>
<td>V</td>
</tr>
<tr>
<td>Power Dissipation</td>
<td>PDmax</td>
<td>6 W</td>
<td></td>
</tr>
<tr>
<td>Ambient(Operating) Temperature</td>
<td>TOPR</td>
<td>0 to +70 °C</td>
<td>°C</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>TSTG</td>
<td>-65 to +150 °C</td>
<td>°C</td>
</tr>
</tbody>
</table>

D. C. Characteristics

<table>
<thead>
<tr>
<th>Item</th>
<th>Symbol</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>VCC</td>
<td>3.15</td>
<td>3.45</td>
<td>V</td>
</tr>
<tr>
<td>Input low voltage</td>
<td>VIL</td>
<td>-0.3</td>
<td>+0.8</td>
<td>V</td>
</tr>
<tr>
<td>Input high voltage</td>
<td>VIH</td>
<td>2.2</td>
<td>VCC + 0.3 V</td>
<td>V</td>
</tr>
<tr>
<td>Output low voltage @ 2.0 mA</td>
<td>VOL</td>
<td>0.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output high voltage @-0.4 mA</td>
<td>VOH</td>
<td>2.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Input leakage current</td>
<td>ILI</td>
<td>-1</td>
<td>1 µA</td>
<td></td>
</tr>
<tr>
<td>Output leakage current</td>
<td>ILO</td>
<td>-1</td>
<td>1 µA</td>
<td></td>
</tr>
<tr>
<td>Capacitance: PQFP</td>
<td>Cin</td>
<td>9 pF</td>
<td>7 pF</td>
<td>pF</td>
</tr>
<tr>
<td>Clock pin</td>
<td></td>
<td>9 pF</td>
<td>9 pF</td>
<td></td>
</tr>
<tr>
<td>Other input pins</td>
<td></td>
<td>9 pF</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Output pins</td>
<td></td>
<td>9 pF</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>I/O pins</td>
<td></td>
<td>9 pF</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

Power Consumption

Package Thermal Resistance

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Air Flow (Meters/Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>ØJA (°C/W)</td>
<td></td>
</tr>
<tr>
<td>ØJC (°C/W)</td>
<td></td>
</tr>
</tbody>
</table>
### 6.0 Timing Diagrams

All timing specifications in this section are based on the following operating conditions:
System Clock Frequency: 40 MHz
VCC = 3.3V ±5%
Temperature: 0 to 70 °C

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLK Low Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLK Fall Time</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLK Rise Time</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BREQ_ Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BREQ_ Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMDREADY Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMDREADY Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t8</td>
<td>TASKDONE Valid</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>t9</td>
<td>BGRT_ Valid</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>t10a</td>
<td>RESET_ # Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t10b</td>
<td>RESET_ Hold</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

![Clock Waveform](image)

**a. Clock Waveform**

Figure 6.1. Timing Diagram of System Clock  (Continued next page)
b. Setup and Hold Timing

c. Valid Delay Timing

Figure 6.1. Timing Diagram of System Clock (Continued)
### CPU to CXD1700 Command/Status Timing

(Figure 6.2)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_7a</td>
<td>CMDREADY Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t_7b</td>
<td>CMDREADY Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t_8</td>
<td>TASKDONE Valid</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Note 1:
The CMDREADY signal is required to be valid ONLY one clock long with the specified Setup and Hold time.

The extended logic level shown ONLY as an example by the user’s external logic for implementing a detecting mechanism to see whether there is a pending command.

---

### CPU Bus Request/Grant Timing

(Figure 6.3)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_6a</td>
<td>BREQ_ Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t_6b</td>
<td>BREQ_ Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t_9</td>
<td>BGRT_ Valid</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

---

**Figure 6.2. Timing Diagram for CMDRDY and TASKDONE**

**Figure 6.3. Timing Diagram for CPU Bus Request and Grant Signals**
Figure 6.4. Timing Diagram for Transfer of Header Data to Bus Bridge Logic
# SDRAM Write Cycle Timing

(Figure 6.5)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t61</td>
<td>XCLK Period</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>t62a</td>
<td>XMADDL[11:0] Setup</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t62b</td>
<td>XMADDL[3:0] Hold</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>t63a</td>
<td>XMADDH[3:0] Setup</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t63b</td>
<td>XMADDH[3:0] Hold</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>t64a</td>
<td>XMCS_, XMWE_, XMCAS_, XMRAS_, XMCLE, XMLDQM, XMUDQM Setup</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t64b</td>
<td>XMCS_, XMWE_, XMCAS_, XMRAS_, XMCLE, XMLDQM, XMUDQM Hold</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Above timing values are for signals output from the switch engine to the memory interface. The signals are referenced to XCLK. XCLK is derived from CLK and output from the switch engine.

---

**Figure 6.5.** Timing Diagram for SDRAM Burst Write Cycle
SDRAM Read Cycle Timing (Figure 6.6)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t61</td>
<td>XCLK Period</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td>t62a</td>
<td>XMADDL[11:0] Setup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t62b</td>
<td>XMADDL[3:0] Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t63a</td>
<td>XMADDH[3:0] Setup</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>t63b</td>
<td>XMADDH[3:0] Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t64a</td>
<td>XMCS_, XMWE_, XMCAS_, XMRAS_, XMCLE,</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XMLDQM, XMUDQM Setup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t64b</td>
<td>XMCS_, XMWE_, XMCAS_, XMRAS_, XMCLE,</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XMLDQM, XMUDQM Hold</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Above timing values are for signals output from the switch engine to the memory interface. The signals are referenced to XCLK. XCLK is derived from CLK and output from the switch engine.

Figure 6.6. Timing Diagram for SDRAM Burst Read Cycle
Data Receive Timing, 10Mbit Port  

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t21</td>
<td>RXC Period</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t22</td>
<td>RXC High Time</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>t23</td>
<td>RXC Low Time</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>t24a</td>
<td>RXC Fall Time</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>t24b</td>
<td>RXC Rise Time</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t25a</td>
<td>CRS Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t25b</td>
<td>CRS Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t26a</td>
<td>COL Setup</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>t26b</td>
<td>COL Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t27</td>
<td>RXD Valid</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Above timing values are with respect to the receive clock.

![Timing Diagram](image)

Figure 6.7. Timing Diagram for Data Receive on 10 Mbit Ports
### Data Transmit Timing, 10Mbit Port

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t11</td>
<td>TXC Period – 10 MHz clocks</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>t12</td>
<td>TXC High Time</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>t13</td>
<td>TXC Low Time</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>t14a</td>
<td>TXC Fall Time</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t14b</td>
<td>TXC Rise Time</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t15</td>
<td>TX_EN Valid</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>t16</td>
<td>TXD Valid</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Above timing values are with respect to the transmit clock.

![Timing Diagram for Data Transmit on 10 Mbit Ports](image)

Figure 6.8. Timing Diagram for Data Transmit on 10 Mbit Ports
## Data Receive Timing, 100Mbit Port

(Figure 6.9)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_{41}</td>
<td>RXC Period – 25 MHz Clocks</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>t_{41}</td>
<td>RXC High Time</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>t_{43}</td>
<td>RXC Low Time</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>t_{44a}</td>
<td>RXC Fall Time</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t_{44b}</td>
<td>RXC Rise Time</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>t_{45a}</td>
<td>CRS Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t_{45b}</td>
<td>CRS Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t_{46a}</td>
<td>COL Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t_{46b}</td>
<td>COL Hold</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t_{47}</td>
<td>RXD[3:0] Valid</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>t_{48a}</td>
<td>RX_DV Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t_{48b}</td>
<td>RX_DV Hold</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>t_{49a}</td>
<td>RX_ER Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t_{49b}</td>
<td>RX_ER Hold</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Above timing values are with respect to the receive clock.

Figure 6.9. Timing Diagram for Data Receive on 100 Mbit Ports
## Data Transmit Timing, 100Mbit Port

(Figure 6.10)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t31</td>
<td>TXC Period – 25 MHz Clocks</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>t32</td>
<td>TXC High Time</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>t33</td>
<td>TXC Low Time</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>t34a</td>
<td>TXC Fall Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t34b</td>
<td>TXC Rise Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t35</td>
<td>TX_EN Valid</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>t36</td>
<td>TXD[3:0] Valid</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>t37a</td>
<td>TX_ER Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>t37b</td>
<td>TX_ER Hold</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Above timing values are with respect to the transmit clock.

![Timing Diagram for Data Transmit on 100 Mbit Ports](image)

Figure 6.10. Timing Diagram for Data Transmit on 100 Mbit Ports
Management Statistics Interface Timing (Figure 6.11)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_{61}</td>
<td>ESTAT_STRB</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>t_{62}</td>
<td>ESTAT_OUT[3:0] Setup</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>t_{62}</td>
<td>ESTAT_OUT[3:0] Hold</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Above timing values are for the output signals from the Switch engine to the external statistics interface. These signals are referenced to ESTAT_STRB. ESTAT_STRB is derived from CLK and output from the switch engine.

Figure 6.11. Timing Diagram for Management Statistics Data Output
### MDIO Input/Output Timing (Figure 6.12)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min(ns)</th>
<th>Max(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{50a}$</td>
<td>MDIO Setup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$t_{50b}$</td>
<td>MDIO Hold</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$t_{51}$</td>
<td>MDIO Valid</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Above timing values are for the MDIO lines of both 100 Mbit ports. They timings are with respect to the MDC clock.

![Timing Diagram for MDIO Input and Output](image)

**Figure 6.12. Timing Diagram for MDIO Input and Output**
Figure 6.13. Timing Diagram for MDIO Write Operation in a PHY

<table>
<thead>
<tr>
<th>MDC</th>
<th>MDIO (STA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 0 1</td>
</tr>
<tr>
<td></td>
<td>value(5 bits) value(5 bits) 1 0 value(16 bits)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bus state</th>
<th>Idle</th>
<th>Start</th>
<th>Opcode</th>
<th>PHY Address</th>
<th>Register Addr</th>
<th>TA</th>
<th>Register Data</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PHY Address</th>
<th>Register Addr</th>
<th>Register data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 6.14. Timing Diagram for MDIO Read Operation in a PHY