Rev: 032609

# DS34S101, DS34S102, DS34S104, DS34S108
# Single/Dual/Quad/Octal TDM-over-Packet Chip

## General Description

These IETF PWE3 SAToP/CESoPSN/TDMoIP/HDLC compliant devices allow up to eight E1, T1 or serial streams or one high-speed E3, T3, STS-1 or serial stream to be transported transparently over IP, MPLS or Ethernet networks. Jitter and wander of recovered clocks conform to G.823/G.824, G.8261, and TDM specifications. TDM data is transported in up to 64 individually configurable bundles. All standards-based TDM-over-packet mapping methods are supported except AAL2. Frame-based serial HDLC data flows are also supported. The high level of integration available with the DS34S10x devices minimizes cost, board space, and time to market.

## Applications

TDM Circuit Extension Over PSN
- Leased-Line Services Over PSN
- TDM Over GPON/EPON
- TDM Over Cable
- TDM Over Wireless

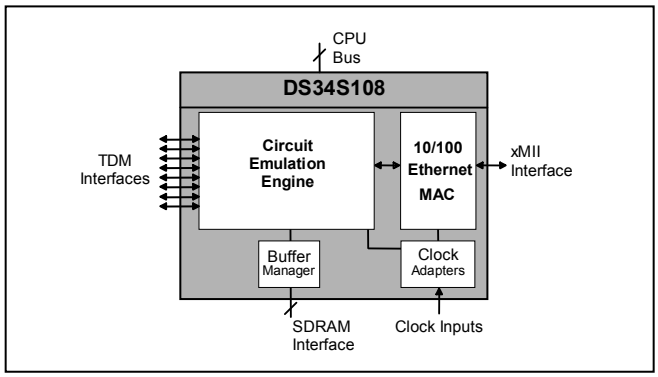
Cellular Backhaul Over PSN
Multiservice Over Unified PSN
HDLC-Based Traffic Transport Over PSN

## Features

- **Transport of E1, T1, E3, T3 or STS-1 TDM or Other CBR Signals Over Packet Networks**
- **Full Support for These Mapping Methods: SAToP, CESoPSN, TDMoIP (AAL1), HDLC, Unstructured, Structured, Structured with CAS**
- **Adaptive Clock Recovery, Common Clock, External Clock and Loopback Timing Modes**
- **On-Chip TDM Clock Recovery Machines, One Per Port, Independently Configurable**
- **Clock Recovery Algorithm Handles Network PDV, Packet Loss, Constant Delay Changes, Frequency Changes and Other Impairments**
- **64 Independent Bundles/Connections**
- **Multiprotocol Encapsulation Supports IPv4, IPv6, UDP, RTP, L2TPv3, MPLS, Metro Ethernet**
- **VLAN Support According to 802.1p and 802.1Q**
- **10/100 Ethernet MAC Supports MII/RMII/SSMII**
- **Selectable 32-Bit, 16-Bit or SPI Processor Bus**
- **Operates from Only Two Clock Signals, One for Clock Recovery and One for Packet Processing**
- **Glueless SDRAM Buffer Management**
- **Low-Power 1.8V Core, 3.3V I/O**

***See detailed feature list in Section 7.***

## Functional Diagram




## Ordering Information

| PART | PORTS | TEMP RANGE | PIN-PACKAGE |
|---|---|---|---|
| **DS34S101**GN | 1 | -40°C to +85°C | 256 TECSBGA |
| DS34S101GN+ | 1 | -40°C to +85°C | 256 TECSBGA |
| **DS34S102**GN | 2 | -40°C to +85°C | 256 TECSBGA |
| DS34S102GN+ | 2 | -40°C to +85°C | 256 TECSBGA |
| **DS34S104**GN | 4 | -40°C to +85°C | 256 TECSBGA |
| DS34S104GN+ | 4 | -40°C to +85°C | 256 TECSBGA |
| **DS34S108**GN | 8 | -40°C to +85°C | 484 HSBGA |
| DS34S108GN+ | 8 | -40°C to +85°C | 484 HSBGA |

*+Denotes a lead(Pb)-free/RoHS-compliant package (explanation).*

## Table of Contents

## List of Figures

## List of Tables

# 1. Introduction

The DS34S101/2/4/8 family of products provide single and multiport TDM-over-packet circuit emulation. Dedicated payload-type engines are included for TDMoIP (AAL1), CESoPSN, SAToP, and HDLC.

Products in the DS34S10x family provide the mapping/demapping ability to enable the transport of TDM data (Nx64kbps, E1, T1, J1, E3, T3, STS-1) over IP, MPLS or Ethernet networks. These products enable service providers to migrate to next generation networks while continuing to provide legacy voice, data and leased-line services. They allow enterprises to transport voice and video over the same IP/Ethernet network that is currently used only for LAN traffic, thereby minimizing network maintenance and operating costs.

Packet-switched networks, such as IP networks, were not designed to transport TDM data and have no inherent clock distribution mechanism. Therefore, when transporting TDM data over packet switched networks, the TDM demapping function needs to accurately reconstruct the TDM service clock(s). The DS34S10x devices perform this important clock recovery task, creating recovered clocks with jitter and wander levels that conform to ITU-T G.823/824 and G.8261, even for networks which introduce significant packet delay variation and packet loss.

The circuit emulation technology in the DS34S10x products that makes this possible is called TDM-over-Packet (TDMoP) and complements VoIP in those cases where VoIP is not applicable or where VoIP price/performance is not sufficient. Most importantly, TDMoP technology provides higher voice quality with lower latency than VoIP. Unlike VoIP, TDMoP can support all applications that run over E1/T1 circuits, not just voice. TDMoP can also provide traditional leased-line services over IP and is transparent to protocols and signaling. Because TDMoP provides an evolutionary, as opposed to revolutionary approach, investment protection is maximized.

# 2. Acronyms and Glossary

## Acronyms

| | |
|---|---|
| AAL1 | ATM Adaptation Layer Type 1 |
| AAL2 | ATM Adaptation Layer Type 2 |
| ATM | Asynchronous Transfer Mode |
| BGA | Ball Grid Array |
| BW | Bandwidth |
| CAS | Channel Associated Signaling |
| CBR | Constant Bit-Rate |
| CCS | Common channel signaling |
| CE | Customer Edge |
| CESoP | Circuit Emulation Service over Packet |
| CESoPSN | Circuit Emulation Services over Packet Switched Network |
| CLAD | Clock Rate Adapter |
| CPE | Customer Premises Equipment |
| CSMA | Carrier Sense Multiple Access |
| CSMA/CD | Carrier Sense Multiple Access with Collision Detection |
| DS0 | Digital Signal Level 0 |
| DS1 | Digital Signal Level 1 |
| DS3 | Digital Signal Level 3 |
| HDLC | High-Level data Link Control |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| JBC | Jitter Buffer Control |
| IWF | Interworking Function |
| LAN | Local Area Network |
| LOF | Loss of Frame (i.e. loss of frame alignment) |
| LOS | Loss of Signal |
| MAC | Media Access Control |
| MEF | Metro Ethernet Forum |
| MFA | MPLS / Frame Relay Alliance (Now called IP/MPLS Forum) |
| MII | Medium Independent Interface |
| MPLS | MULTI PROTOCOL LABEL SWITCHING |
| OC-3 | Optical Carrier Level 3 |
| OCXO | Oven Controlled Crystal Oscillator |
| OFE | Optical Front End |
| OSI | Open Systems Interconnection |
| OSI-RM | Open Systems Interconnection—Reference Model |
| PDH | Plesiochronous Digital Hierarchy |
| PDU | Protocol Data Unit |
| PDV | Packet Delay Variation |
| PE | Provider Edge |
| PRBS | Pseudo-Random Bit Sequence |
| PSN | Packet Switched Network |
| PSTN | Public Switched Telephone Network |
| PWE3 | Pseudo-Wire Emulation Edge-to-Edge |
| QoS | Quality of Service |
| RMII | Reduced Medium Independent Interface |
| Rx or RX | Receive |
| SAR | Segmentation and Reassembly |
| SAToP | Structure-Agnostic TDM over Packet |
| SDH | Synchronous Digital Hierarchy |
| SMII | Serial Media Independent Interface |
| SN | Sequence Number |
| SONET | Synchronous Optical Network |

| | |
|---|---|
| SS7 | Signaling System 7 |
| SSMII | Source Synchronous Serial Media Independent Interface |
| STM-1 | Synchronous Transport Module Level 1 |
| TDM | Time Division Multiplexing |
| TDMoIP | TDM over Internet Protocol |
| TDMoP | TDM over Packet |
| TSA | Timeslot Assigner |
| Tx or TX | Transmit |
| UDP | User Datagram Protocol |
| VoIP | Voice over IP |
| VPLS | Virtual Private LAN Services |
| WAN | Wide Area Network |

### *Glossary*

**bundle** – a virtual path configured at two endpoint TDMoP gateways to carry TDM data over a PSN.

**CLAD** – Clock Rate Adapter, an analog PLL that creates an output clock signal that is phase/frequency locked to an input clock signal of a different frequency. A CLAD is said to “convert” one frequency to another or “adapt” (change) a clock’s rate to be a frequency that is useful to some other block on the chip.

**dword** – a 32-bit (4-byte) unit of information (also known as a doubleword)

# 3. Applicable Standards

**Table 3-1. Applicable Standards**

| SPECIFICATION | SPECIFICATION TITLE |
|---|---|
| **IEEE** | |
| IEEE 802.3 | *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications* (2005) |
| IEEE 1149.1 | *Standard Test Access Port and Boundary-Scan Architecture*, 1990 |
| **IETF** | |
| RFC 4553 | *Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)* (06/2006) |
| RFC 4618 | *Encapsulation Methods for Transport of PPP/High-Level Data Link Control (HDLC) over MPLS Networks* (09/2006) |
| RFC 5086 | *Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)* (12/2007) |
| RFC 5087 | *Time Division Multiplexing over IP (TDMoIP)* (12/2007) |
| **ITU-T** | |
| G.823 | *The Control of Jitter and Wander within Digital Networks which are Based on the 2048kbps Hierarchy* (03/2000) |
| G.824 | *The Control of Jitter and Wander within Digital Networks which are Based on the 1544kbps Hierarchy* (03/2000) |
| G.8261/Y.1361 | *Timing and Synchronization Aspects in Packet Networks* (05/2006) |
| I.363.1 | *B-ISDN ATM Adaptation Layer Specification: Type 1 AAL* (08/1996) |
| I.363.2 | *B-ISDN ATM Adaptation Layer Specification: Type 2 AAL* (11/2000) |
| I.366.2 | *AAL Type 2 Service Specific Convergence Sublayer for Narrow-Band Services* (11/2000) |
| O.151 | *Error Performance Measuring Equipment Operating at the Primary Rate and Above* (1992) |
| O.161 | *In-service Code Violation Monitors for Digital Systems* (1993) |
| Y.1413 | *TDM-MPLS Network Interworking – User Plane Interworking* (03/2004) |
| Y.1414 | *Voice Services–MPLS Network Interworking* (07/2004) |
| Y.1452 | *Voice Trunking over IP Networks* (03/2006) |
| Y.1453 | *TDM-IP Interworking – User Plane Networking* (03/2006) |
| **MEF** | |
| MEF 8 | *Implementation Agreement for the Emulation of PDH Circuits over Metro Ethernet Networks* (10/2004) |
| **MFA** | |
| MFA 4.0 | *TDM Transport over MPLS Using AAL1* (06/2003) |
| MFA 5.0.0 | *I.366.2 Voice Trunking Format over MPLS Implementation Agreement (08/2003)* |
| MFA 8.0.0 | *Emulation of TDM Circuits over MPLS Using Raw Encapsulation – Implementation Agreement* (11/2004) |

# 4. Detailed Description

The DS34S108 is an 8-port TDM-over-Packet (TDMoP) IC. The DS34S104, DS34S102 and DS34S101 have the same functionality as the DS34S108, except they have only 4, 2 or 1 ports, respectively. These sophisticated devices can map and demap multiple E1/T1 data streams or a single E3/T3/STS-1 data stream to and from IP, MPLS or Ethernet networks. A built-in MAC supports connectivity to a single 10/100 Mbps PHY over an MII, RMII or SSMII interface. The DS34S10x devices are controlled through a 16 or 32-bit parallel bus interface or a high-speed SPI serial interface.

The TDM-over-Packet (TDMoP) core is the enabling block for circuit emulation and other network applications. It performs transparent transport of legacy TDM traffic over Packet Switched-Networks (PSN). The TDMoP core implements payload mapping methods such as AAL1 for circuit emulation, HDLC method, structure-agnostic SAToP method, and the structure-aware CESoPSN method.

The AAL1 payload-type machine maps and demaps E1, T1, E3, T3, STS-1 and other serial data flows into and out of IP, MPLS or Ethernet packets, according to the methods described in ITU-T Y.1413, Y.1453, MEF 8, MFA 4.1 and IETF RFC 5087 (TDMoIP). It supports E1/T1 structured mode with or without CAS, using a timeslot size of 8 bits, or unstructured mode (carrying serial interfaces, unframed E1/T1 or E3/T3/STS-1 traffic).

The HDLC payload-type machine maps and demaps HDLC dataflows into and out of IP/MPLS packets according to IETF RFC 4618 (excluding clause 5.3 – PPP) and IETF RFC 5087 (TDMoIP). It supports 2-, 7- and 8-bit timeslot resolution (i.e. 16, 56, and 64 kbps respectively), as well as N × 64 kbps bundles (n=1 to 32). Supported applications of this machine include trunking of HDLC-based traffic (such as Frame Relay) implementing Dynamic Bandwidth Allocation over IP/MPLS networks and HDLC-based signaling interpretation (such as ISDN D-channel signaling termination – BRI or PRI, V5.1/2, or GR-303).

The SAToP payload-type machine maps and demaps unframed E1, T1, E3 or T3 data flows into and out of IP, MPLS or Ethernet packets according to ITU-T Y.1413, Y.1453, MEF 8, MFA 8.0.0 and IETF RFC 4553. It supports E1/T1/E3/T3 with no regard for the TDM structure. If TDM structure exists it is ignored, allowing this to be the simplest mapping/demapping method. The size of the payload is programmable for different services. This emulation suits applications where the provider edges have no need to interpret TDM data or to participate in the TDM signaling. The PSN network must have almost no packet loss and very low packet delay variation (PDV) for this method.

The CESoPSN payload-type machine maps and demaps structured E1, T1, E3 or T3 data flows into and out of IP, MPLS or Ethernet packets with static assignment of timeslots inside a bundle according to ITU-T Y.1413, Y.1453, MEF 8, MFA 8.0.0 and the IETF RFC 5086 (CESoPSN). It supports E1/T1/E3/T3 while taking into account the TDM structure. The level of structure must be chosen for proper payload conversion such as the framing type (i.e. frame or multiframe). This method is less sensitive to PSN impairments but lost packets could still cause service interruption.

# 5. Application Examples

In Figure 5-1, a DS34S10x device is used in each TDMoP gateway to map TDM services into a packet-switched metropolitan network. TDMoP data is carried over various media: fiber, wireless, G/EPON, coax, etc.

*Figure 5-1. TDMoP in a Metropolitan Packet Switched Network*

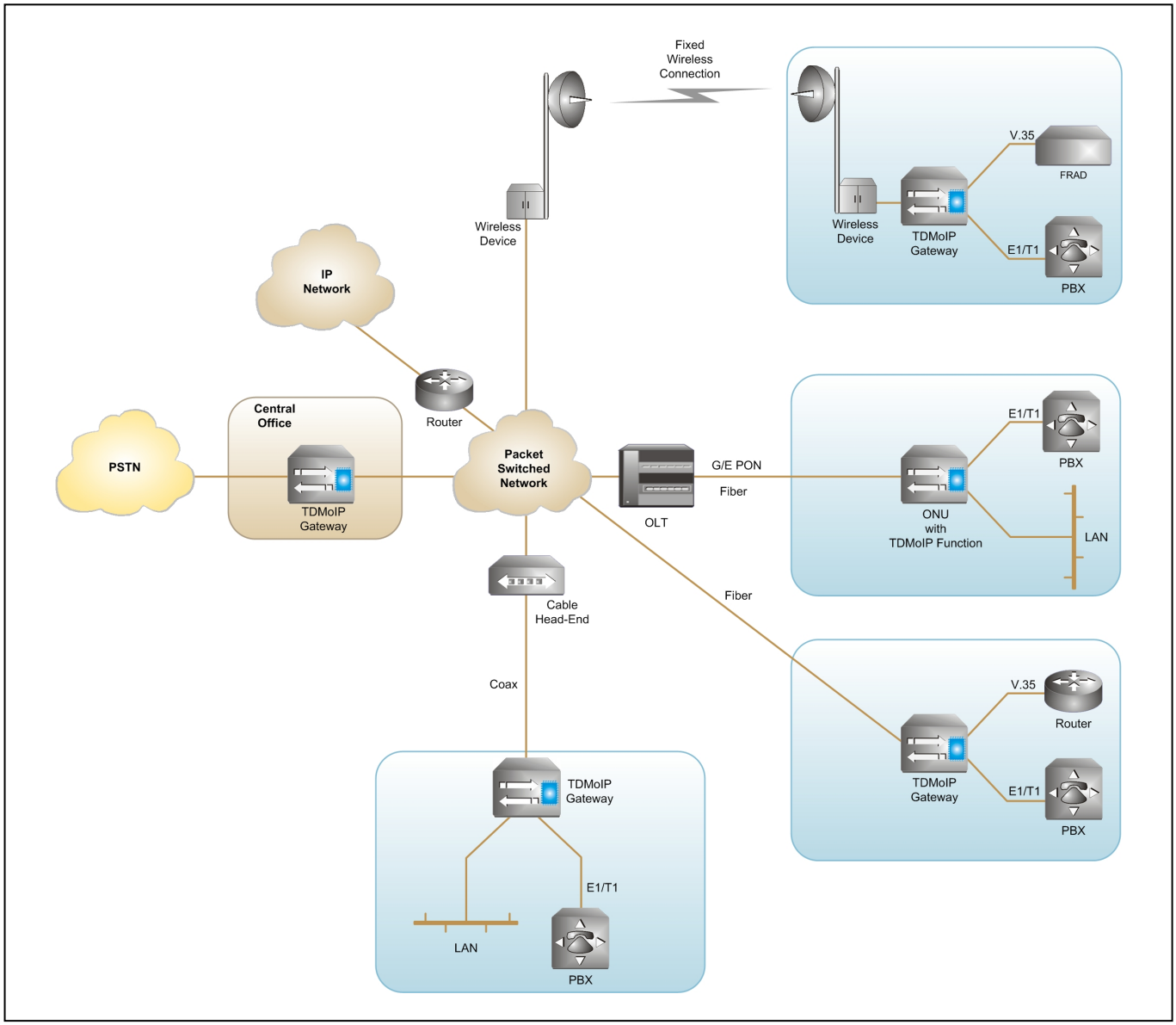
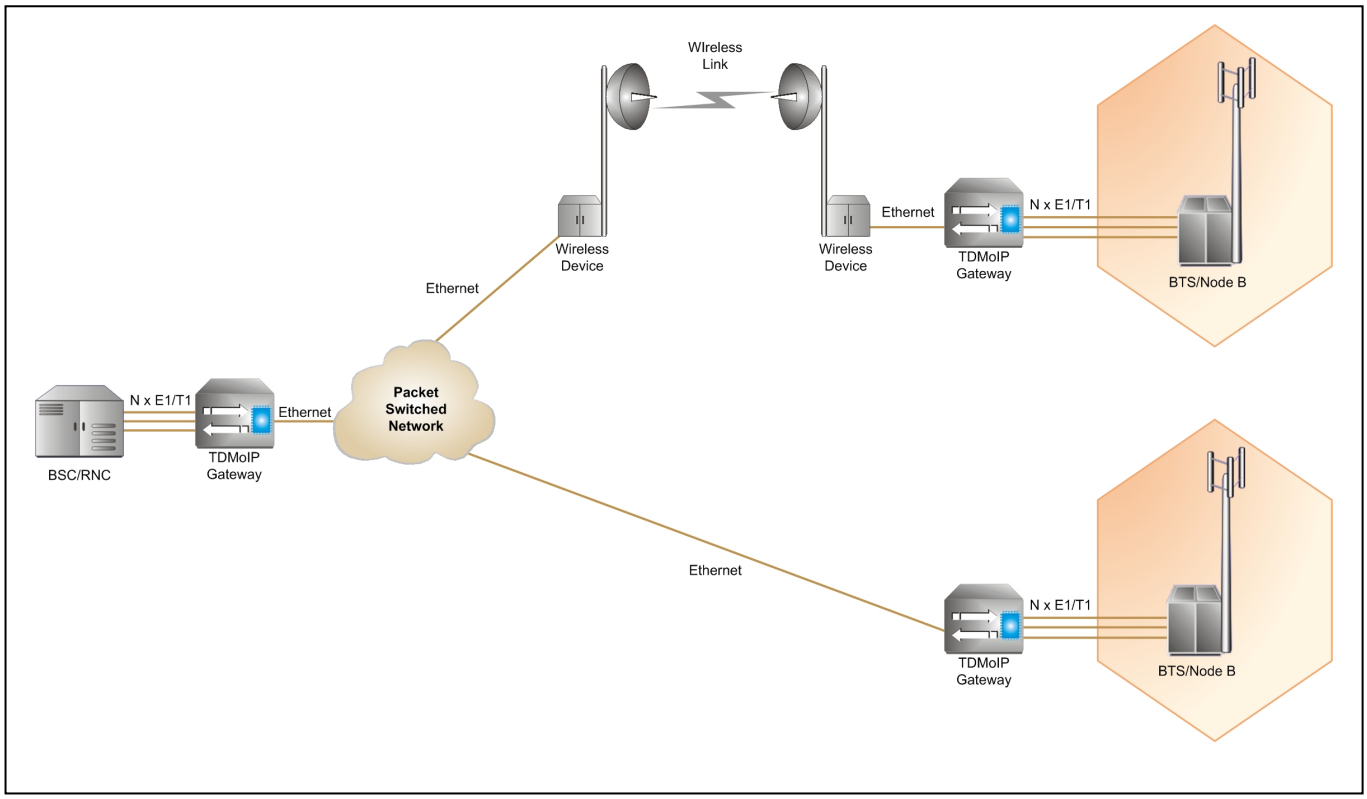


*Figure 5-2. TDMoP in Cellular Backhaul*




## Other Possible Applications

### Point-to-Multipoint TDM Connectivity over IP/Ethernet

The DS34S10x devices support NxDS0 TDMoP connections (known as bundles) with or without CAS (Channel Associated Signaling). There is no need for an external TDM cross-connect, since the packet domain can be used as a virtual cross-connect. Any bundle of timeslots can be directed to another remote location on the packet domain.

### HDLC Transport over IP/MPLS

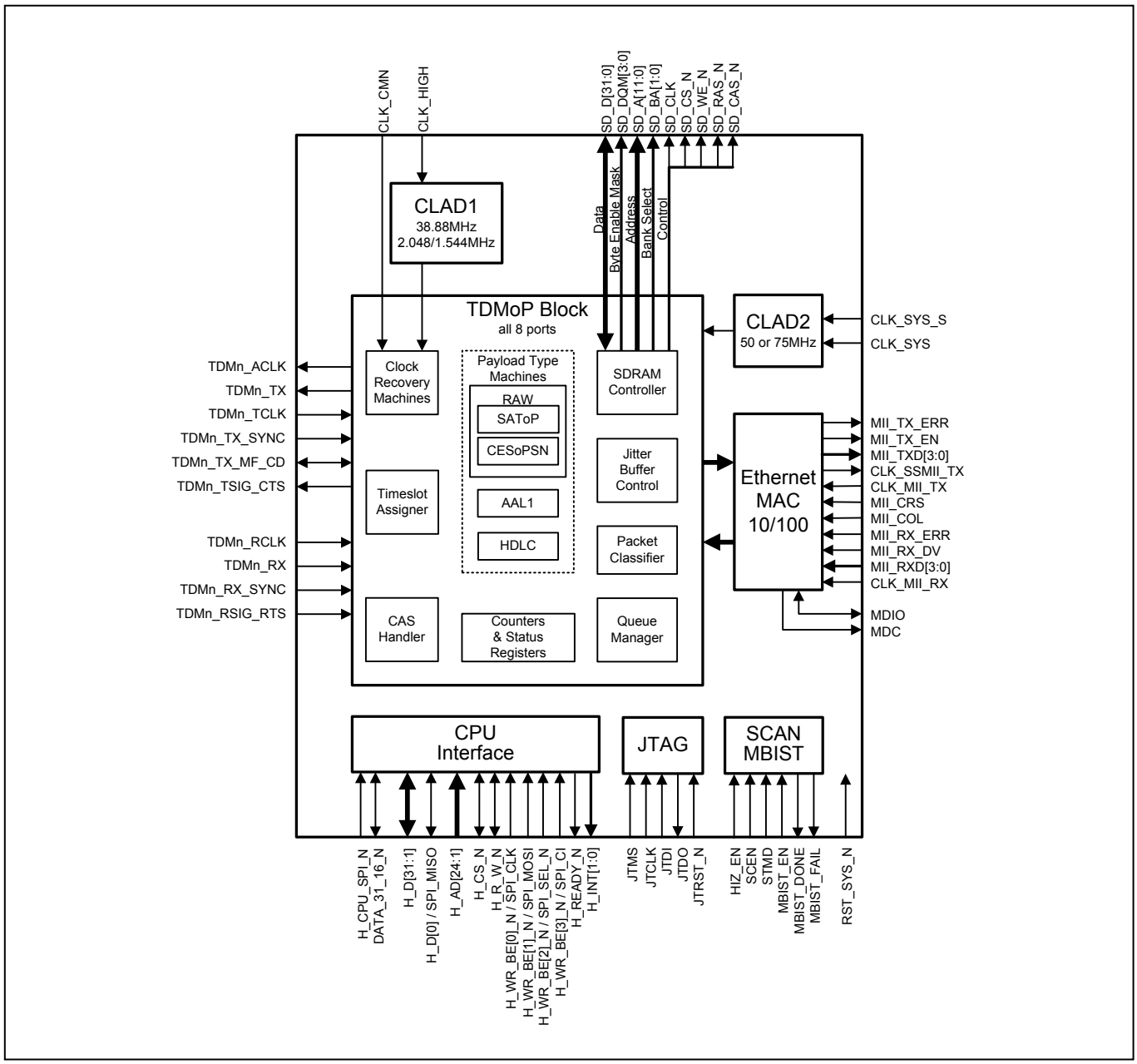
TDM traffic streams often contain HDLC-based control channels and data traffic. These data streams, when transported over a packet domain, should be treated differently than the time-sensitive TDM payload. DS34S10x devices can terminate HDLC channels in the TDM streams and optionally map them into IP/MPLS/Ethernet for transport. All HDLC-based control protocols (ISDN BRI and PRI, SS7 etc.) and all HDLC data traffic can be managed and transported.

### Using a Packet Backplane for Multiservice Concentrators

A communications device with all the above-mentioned capabilities can use a packet-based backplane instead of the more expensive TDM bus option. This enables a cost-effective and future-proof design of communication platforms with full support for both legacy and next-generation services.

# 6. Block Diagram

*Figure 6-1. Top-Level Block Diagram*



# 7. FEATURES

## Global Features

- TDMoP Interfaces
  - DS34S101: 1 E1/T1/serial TDM interface
  - DS34S102: 2 E1/T1/serial TDM interfaces
  - DS34S104: 4 E1/T1/serial TDM interfaces
  - DS34S108: 8 E1/T1/serial TDM interfaces
  - All four devices: optionally 1 high-speed E3/DS3/STS-1 TDM interface
  - All four devices: each interface optionally configurable for serial operation for V.35 and RS530
- Ethernet Interface
  - One 10/100 Mbps port configurable for MII, RMII or SSMII interface format
  - Half or full duplex operation
  - VLAN support according to 802.1p and 802.1Q including stacked tags
  - Fully compatible with IEEE 802.3 standard
- End-to-end TDM synchronization through the IP/MPLS domain by on-chip, per-port TDM clock recovery
- 64 independent bundles/connections, each with its own:
  - Transmit and receive queues
  - Configurable jitter-buffer depth
  - Connection-level redundancy, with traffic duplication option
- Packet loss compensation and handling of misordered packets
- Glueless SDRAM interface
- Complies with MPLS-Frame Relay Alliance Implementation Agreements 4.1, 5.1 and 8.0
- Complies with ITU-T standards Y.1413 and Y.1414.
- Complies with Metro Ethernet Forum 3 and 8
- Complies with IETF RFC 4553 (SAToP), RFC 5086 (CESoPSN) and RFC 5087 (TDMoIP)
- IEEE 1146.1 JTAG boundary scan
- 1.8V and 3.3V Operation with 5.0V tolerant I/O

## Clock Synthesizers

- Clocks to operate the TDMoP clock recovery machines can synthesized from a single clock input (10MHz, 19.44MHz, 38.88MHz or 77.76MHz on the CLK_HIGH pin)
- Clock to operate TDMoP logic and SDRAM interface (50MHz or 75MHz) can be synthesized from a single 25MHz clock on the CLK_SYS pin

## TDM-over-Packet Block

- Enables transport of TDM services (E1, T1, E3, T3, STS-1) or serial data over packet-switched networks
- SAToP payload-type machine maps/demaps unframed E1/T1/E3/T3/STS-1 or serial data flows to/from IP, MPLS or Ethernet packets according to ITU-T Y.1413, Y.1453, MEF 8, MFA 8.0.0 and IETF RFC 4553.
- CESoPSN payload-type machine maps/demaps structured E1/T1 data flows to/from IP, MPLS or Ethernet packets with static assignment of timeslots inside a bundle according to ITU-T Y.1413, Y.1453, MEF 8, MFA 8.0.0 and IETF RFC 5086.
- AAL1 payload-type machine maps/demaps E1/T1/E3/T3/STS-1 or serial data flows to/from IP, MPLS or Ethernet packets according to ITU-T Y.1413, MEF 8, MFA 4.1 and IETF RFC 5087. For E1/T1 it supports structured mode with/without CAS using 8-bit timeslot resolution, while implementing static timeslot allocation. For E1/T1, E3/T3/STS-1 or serial interface it supports unstructured mode.
- HDLC payload-type machine maps/demaps HDLC-based E1/T1/serial flow to/from IP, MPLS or Ethernet packets. It supports 2-, 7- and 8-bit timeslot resolution (i.e. 16, 56, and 64 kbps respectively), as well as N x 64 kbps bundles. This is useful in applications where HDLC-based signaling interpretation is required (such as ISDN D channel signaling termination, V.51/2, or GR-303), or for trunking packet-based applications (such as Frame Relay), according to IETF RFC 4618.

### TDMoP TDM Interfaces

- Supports single high-speed E3, T3 or STS-1 interface on port 1 or one (DS34S101), two (DS34S102), four (DS34S104) or eight (DS34S108) E1, T1 or serial interfaces
- For single high-speed E3, T3 or STS-1 interface, AAL1 or SAToP payload type is used
- For E1 or T1 interfaces, the following modes are available:
  - Unframed – E1/T1 pass-through mode (AAL1, SAToP or HDLC payload type)
  - Structured – fractional E1/T1 support (all payloads)
  - Structured with CAS – fractional E1/T1 with CAS support (CESoPSN or AAL1 payload type)
- For serial interfaces, the following modes are available:
  - Arbitrary continuous bit stream (using AAL1 or SAToP payload type)
  - Single-interface high-speed mode on port 1 up to STS-1 rate (51.84 Mbps) using a single bundle/connection.
  - Low-speed mode with each interface operating at N x 64 kbps (N = 1 to 63) with an aggregate rate of 18.6Mbps
  - HDLC-based traffic (such as Frame Relay) at N x 64 kbps (N = 1 to 63) with an aggregate rate of 18.6Mbps).
- All serial interface modes are capable of working with a gapped clock.

### TDMoP Bundles

- 64 independent bundles, each can be assigned to any TDM interface
- Each bundle carries a data stream from one TDM interface over IP/MPLS/Ethernet PSN from TDMoP source device to TDMoP destination device
- Each bundle may be for N x 64kbps, an entire E1, T1, E3, T3 or STS-1, or an arbitrary serial data stream
- Each bundle is unidirectional (but frequently coupled with opposite-direction bundle for bidirectional communication)
- Multiple bundles can be transported between TDMoP devices
- Multiple bundles can be assigned to the same TDM interface
- Each bundle is independently configured with its own:
  - Transmit and receive queues
  - Configurable receive-buffer depth
  - Optional connection-level redundancy (SAToP, AAL1, CESoPSN only).
- Each bundle can be assigned to one of the payload-type machines or to the CPU
- For E1/T1 the device provides internal bundle cross-connect functionality, with DS0 resolution

### TDMoP Clock Recovery

- Sophisticated TDM clock recovery machines, one for each TDM interface, allow end-to-end TDM clock synchronization, despite the packet delay variation of the IP/MPLS/Ethernet network
- The following clock recovery modes are supported:
  - Adaptive clock recovery
  - Common clock (using RTP)
  - External clock
  - Loopback clock
- The clock recovery machines provide both fast frequency acquisition and highly accurate phase tracking:
  - Jitter and wander of the recovered clock are maintained at levels that conform to G.823/G.824 traffic or synchronization interfaces. (For adaptive clock recovery, the recovered clock performance depends on packet network characteristics.)
  - Short-term frequency accuracy (1 second) is better than 16 ppb (using OCXO reference), or 100 ppb (using TCXO reference)
  - Capture range is ±90 ppm
  - Internal synthesizer frequency resolution of 0.5 ppb
  - High resilience to packet loss and misordering, up to 2% without degradation of clock recovery performance

- Robust to sudden significant constant delay changes
- Automatic transition to holdover when link break is detected

### TDMoP Delay Variation Compensation

- Configurable jitter buffers compensate for delay variation introduce by the IP/MPLS/Ethernet network
- Large maximum jitter buffer depths:
  - E1: up to 256 ms
  - T1 unframed: up to 340 ms
  - T1 framed: up to 256 ms
  - T1 framed with CAS: up to 192 ms
  - E3: up to 60 ms
  - T3: up to 45 ms
  - STS-1: up to 40 ms.
- Packet reordering is performed for SAToP and CESoPSN bundles within the range of the jitter buffer
- Packet loss is compensated by inserting either a pre-configured conditioning value or the last received value.

### TDMoP CAS Support

- On-chip CAS handler terminates E1/T1 CAS when using AAL1/CESoPSN in structured-with-CAS mode.
- CPU intervention is not required for CAS handling.

### Test and Diagnostics

- IEEE 1149.1 JTAG support
- MBIST (memory built-in self test)

### CPU Interface

- 32 or 16-bit parallel interface or optional SPI serial interface
- Byte write enable pins for single-byte write resolution
- Hardware reset pin
- Software reset supported
- Software access to device ID and silicon revision
- On-chip SDRAM controller provides access to SDRAM for both the chip and the CPU
- CPU can access transmit and receive buffers in SDRAM used for packets to/from the CPU (ARP, SNMP, etc.)

## 8. OVERVIEW OF MAJOR OPERATIONAL MODES

Globally, the resources of the device can be committed to either one high-speed E3, T3 or STS-1 TDM stream (high-speed mode) or one or more E1, T1 or serial streams (normal low-speed mode). In high-speed mode, the TDM signal is carried using an unstructured AAL1 or SAToP mapping. High-speed mode is enabled by setting General_cfg_reg0.High_speed=1.

In normal, low-speed mode, each port can be configured for E1, T1 or serial (e.g. V.35) operation. Ports configured for E1 or T1 can be further configured for unframed, framed, or multiframed interface. In addition, each port can be configured to have the transmit and receive directions clocked by independent clocks (two-clock mode) or to have both directions clocked by the transmit clock (one-clock mode). All of this configuration is specified in the per-port Port[n]_cfg_reg register.

# 9. PIN DESCRIPTIONS

## 9.1 Short Pin Descriptions

**Table 9-1. Short Pin Descriptions**

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| **TDM Interface** | | |
| TDMn_ACLK | O | TDMoP Recovered Clock Output |
| TDMn_TCLK | Ipu | TDMoP Transmit Clock Input (here transmit means “away from Ethernet MII”) |
| TDMn_TX | O | TDMoP Transmit Data Output |
| TDMn_TX_SYNC | Ipd | TDMoP Transmit Frame Sync Input |
| TDMn_TX_MF_CD | IOpd | TDMoP Transmit Multiframe Sync Input or Carrier Detect Output |
| TDMn_TSIG_CTS | O | TDMoP Transmit Signaling Output or Clear to Send Output |
| TDMn_RCLK | Ipu | TDMoP Receive Clock Input (here receive means “toward Ethernet MII”) |
| TDMn_RX | Ipu | TDMoP Receive Data Input |
| TDMn_RX_SYNC | Ipd | TDMoP Receive Frame/Multiframe Sync Input |
| TDMn_RSIG_RTS | Ipu | TDMoP Receive Signaling Input or Request To Send Input |
| **SDRAM Interface** | | |
| SD_CLK | O | SDRAM Clock |
| SD_D[31:0] | IO | SDRAM Data Bus |
| SD_DQM[3:0] | O | SDRAM Byte Enable Mask |
| SD_A[11:0] | O | SDRAM Address Bus |
| SD_BA[1:0] | O | SDRAM Bank Select Outputs |
| SD_CS_N | O | SDRAM Chip Select (Active Low) |
| SD_WE_N | O | SDRAM Write Enable (Active Low) |
| SD_RAS_N | O | SDRAM Row Address Strobe (Active Low) |
| SD_CAS_N | O | SDRAM Column Address Strobe (Active Low) |
| **Ethernet PHY Interface (MII/RMII/SSMII)** | | |
| CLK_MII_TX | I | MII Transmit Clock Input |
| CLK_SSMII_TX | O | SSMII Transmit Clock Output |
| MII_TXD[3:0] | O | MII Transmit Data Outputs |
| MII_TX_EN | O | MII Transmit Enable Output |
| MII_TX_ERR | O | MII Transmit Error Output |
| CLK_MII_RX | I | MII Receive Clock Input |
| MII_RXD[3:0] | I | MII Receive Data Inputs |
| MII_RX_DV | I | MII Receive Data Valid Input |
| MII_RX_ERR | I | MII Receive Error Input |
| MII_COL | I | MII Collision Input |
| MII_CRS | I | MII Carrier Sense Input |
| MDC | O | PHY Management Clock Output |
| MDIO | IOpu | PHY Management Data Input/Output |
| **Global Clocks** | | |
| CLK_SYS_S | I | System Clock Selection Input |
| CLK_SYS | I | System Clock Input: 25, 50 or 75MHz |
| CLK_CMN | I | Common Clock Input (for common clock mode also known as differential mode) |
| CLK_HIGH | I | Clock High Input (for adaptive clock recovery machines and E1/T1 master clocks) |
| **CPU Interface** | | |
| H_CPU_SPI_N | Ipu | Host Bus Interface (1=Parallel Bus, 0=SPI Bus) |
| DAT_32_16_N | Ipu | Data Bus Width (1=32-bit , 0=16-bit) |
| H_D[31:1] | IO | Host Data Bus |
| H_D[0] / SPI_MISO | IO | Host Data LSb or SPI Data Output |
| H_AD[24:1] | I | Host Address Bus |
| H_CS_N | I | Host Chip Select (Active Low) |
| H_R_W_N / SPI_CP | I | Host Read/Write Control or SPI Clock Phase |
| H_WR_BE0_N / SPI_CLK | I | Host Write Enable Byte 0 (Active Low) or SPI Clock |
| H_WR_BE1_N / SPI_MOSI | I | Host Write Enable Byte 1 (Active Low) or SPI Data Input |

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| H_WR_BE2_N / SPI_SEL_N | I | Host Write Enable Byte 2 or SPI Chip Select (Active Low) |
| H_WR_BE3_N / SPI_CI | I | Host Write Enable Byte 3 (Active Low) or SPI Clock Invert |
| H_READY_N | Oz | Host Ready Output (Active Low) |
| H_INT | O | Host Interrupt Output. |
| **JTAG Interface** | | |
| JTRST_N | Ipu | JTAG Test Reset |
| JTCLK | I | JTAG Test Clock |
| JTMS | Ipu | JTAG Test Mode Select |
| JTDI | Ipu | JTAG Test Data Input |
| JTDO | Oz | JTAG Test Data Output |
| **Reset and Factory Test Pins** | | |
| RST_SYS_N | Ipu | System Reset (Active Low) |
| HIZ_N | I | High Impedance Enable (Active Low) |
| SCEN | Ipd | Used for factory tests. |
| STMD | Ipd | Used for factory tests. |
| MBIST_EN | I | Used for factory tests. |
| MBIST_DONE | O | Used for factory tests. |
| MBIST_FAIL | O | Used for factory tests |
| TEST_CLK | O | Used for factory tests. |
| TST_CLD | I | Used for factory tests. DS34S108 only. |
| **Power and Ground** | | |
| DVDDC | P | 1.8V Core Voltage for TDM-over-Packet Digital logic (17 pins) |
| DVDDIO | P | 3.3V for I/O Pins (16 pins) |
| DVSS | P | Ground for TDM-over-Packet logic and I/O Pins (31 pins) |
| ACVDD1, ACVDD2 | P | 1.8V for CLAD Analog Circuits |
| ACVSS1, ACVSS2 | P | Ground for CLAD Analog Circuits |

**Note 1:** In pin names, the suffix "n" stands for port number: n=1 to 8 for DS34S108; n=1 to 4 for DS34S104; n=2 for DS34S102; n=1 for DS34S101. All pin names ending in "_N" are active low.

**Note 2:** All pins, except power and analog pins, are CMOS/TTL unless otherwise specified in the pin description.

**PIN TYPES**

I = input pin
$I_{PD}$ = input pin with internal 50kΩ pulldown to DVSS
$I_{PU}$ = input pin with internal 50kΩ pullup to DVDDIO
IO = input/output pin
$IO_{PD}$ = input/output pin with internal 50kΩ pulldown to DVSS
$IO_{PU}$ = input/output pin with internal 50kΩ pullup to DVDDIO
O = output pin
$O_Z$ = output pin that can be placed in a high-impedance state
P = power-supply or ground pin

## 9.2 Detailed Pin Descriptions

**Table 9-2. TDM-over-Packet Engine TDM Interface Pins**

In this table, the transmit direction is the packet-to-TDM direction while the receive direction is the TDM-to-packet direction. See Figure 6-1.

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| TDMn_ACLK | O<br>8mA | **TDMoP Recovered Clock Output**<br>The clock recovered by the TDMoP clock recovery machine is output on this pin. TDM1_ACLK (port 1) is used in high speed E3/T3/STS1 mode. |
| TDMn_TCLK | Ipu | **TDMoP Transmit Clock Input**<br>This signal clocks the transmit TDM interface of the TDMoP engine. Depending on the value of Port[n]_cfg_reg:tx_sample, outputs TDMn_TX and TDMn_TSIG_CTS are updated on the either the rising edge (0) or falling edge (1) of TDMn_TCLK. Inputs TDMn_TX_SYNC and TDMn_TX_MF_CD are latched on the opposite edge. See the timing diagrams in Figure 14-8 and Figure 14-9.<br><br>In one-clock mode, TDMn_TCLK also clocks the receive TDM interface of the TDMoP engine. Depending on the value of Port[n]_cfg_reg:tx_sample, outputs TDMn_RX, TDMn_RX_SYNC and TDMn_RSIG_RTS are updated on the either the rising edge (0) or falling edge (1) of TDMn_TCLK.<br><br>Port[n]_cfg_reg.Two_clocks specifies two-clock mode (1) or one-clock mode (0).<br>This pin is only active in external mode (GCR1.MODE=1).<br>Only TDM1_TCLK (port 1) is used in high speed E3/T3/STS1 mode (General_cfg_reg0.High_speed=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_TX | O<br>8mA | **TDMoP Transmit Data Output**<br>Serial data from the TDMoP engine is output on this pin.<br>This signal is clocked by TDMn_TCLK.<br><br>Only TDM1_TX (port 1) is used in high speed E3/T3/STS1 mode (i.e. when General_cfg_reg0.High_speed=1).<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_TX_SYNC | Ipd | **TDMoP Transmit Frame Sync Input**<br>Frame sync information is provided to the TDMoP engine from this pin. In two-clock mode, this signal specifies only transmit frame sync. In one-clock mode, this signal specifies frame sync for both the transmit and receive directions.<br><br>The signal on this pin must pulse high for one TDMn_TCLK cycle when the first bit of a frame is expected to present on the TDMn_TX pin (and the TDMn_RX pin in one-clock mode). This pulse must be repeated every N*125µs where N is a positive integer (example: if N=16, it pulses every 2ms).<br><br>Port[n]_cfg_reg.Two_clocks specifies two-clock mode (1) or one-clock mode (0).<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_TX_MF_CD | IOpd | **TDMoP Transmit Multiframe Sync Input**<br>When the interface type is configured for E1 or T1, multiframe sync is provided to the TDMoP engine from this pin. The signal on this pin must pulse high for one TDMn_TCLK cycle when the first bit the multiframe is expected to be present on the TDMn_TX pin.<br><br>**TDMoP Transmit Carrier Detect Output**<br>When the interface type is configured for serial, the carrier detect function of this pin is active. When Port[n]_cfg_reg.CD_en=1, the state of this pin is controlled by the value stored in Port[n]_cfg_reg.CD. |

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| | | Port[n]_cfg_reg.Int_type=specifies serial (00), E1 (01) or T1 (10).<br>Port[n]_cfg_reg.Int_type=specifies serial (00), E1 (01) or T1 (10) interface type.<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_TSIG_CTS | O<br>8mA | **TDMoP Transmit Signaling Output**<br>When the interface type is configured for E1 or T1, the transmit signaling function of this pin is active. Functional timing is shown in Figure 10-33 and Figure 10-34.<br><br>**TDMoP Clear to Send Output**<br>When the interface type is configured for serial, the clear-to-send function of this pin is active. In this mode, the state of this pin is controlled by the value stored in Port[n]_cfg_reg.CTS.<br><br>Port[n]_cfg_reg.Int_type specifies serial (00), E1 (01) or T1 (10) interface type.<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_RCLK | Ipu | **TDMoP Receive Clock Input**<br>In two-clock mode, this signal clocks the receive TDM interface of the TDMoP engine: TDMn_RX, TDMn_RX_SYNC and TDMn_RSIG_RTS.<br><br>In one-clock mode, this signal is ignored, and the TDMn_TCLK signal clocks both the transmit and receive interfaces of the TDMoP engine.<br><br>Port[n]_cfg_reg.Two_clocks specifies two-clock mode (1) or one-clock mode (0).<br>Port[n]_cfg_reg.Rx_sample specifies latching on the rising (1) or falling (0) edge.<br>TDM1_RCLK (port 1) is used in high speed E3/T3/STS1 mode.<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_RX | Ipu | **TDMoP Receive Data Input**<br>Serial data to the TDMoP engine is input on this pin.<br>In two-clock mode, this signal is clocked by TDMn_RCLK.<br>In one-clock mode, this signal, is clocked by TDMn_TCLK.<br><br>Port[n]_cfg_reg.Two_clocks specifies two-clock mode (1) or one-clock mode (0).<br>TDM1_RX (port 1) is used in high speed E3/T3/STS1 mode.<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_RX_SYNC | Ipd | **TDMoP Receive Frame/Multiframe Sync Input**<br>In two-clock mode, this signal is clocked by TDMn_RCLK and specifies frame or multiframe alignment for the receive interface of the TDMoP engine. The signal on this pin must pulse high for one TDMn_RCLK cycle when the first bit of a frame is present on the TDMn_RX pin. This pulse must be repeated every N*125µs where N is a positive integer (example: if N=16, it pulses every 2ms).<br><br>In one-clock mode, this signal is ignored and TDMn_TX_SYNC specifies frame alignment for both the transmit and receive interfaces of the TDMoP engine.<br><br>Port[n]_cfg_reg.Two_clocks specifies two-clock mode (1) or one-clock mode (0).<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| TDMn_RSIG_RTS | Ipu | **TDMoP Receive Signaling Input**<br>When the interface type is configured for E1 or T1, the transmit signaling function of this pin is active.<br>In two-clock mode, this signal is clocked by TDMn_RCLK.<br>In one-clock mode, this signal, is clocked by TDMn_TCLK. |

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| | | **TDMoP Request To Send Input**<br>When the interface type is configured for serial, the request-to-send function of this pin is active. In this mode, the real-time status of this pin can be read from Port[n]_stat_reg1.RTS_P.<br><br>Port[n]_cfg_reg.Two_clocks specifies two-clock mode (1) or one-clock mode (0).<br>Port[n]_cfg_reg.Int_type specifies serial (00), E1 (01) or T1 (10) interface type.<br>This pin is only active in external mode (GCR1.MODE=1).<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |

**Table 9-3. SDRAM Interface Pins**

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| SD_CLK | O<br>8mA | **SDRAM Clock**<br>All SDRAM interface pins are updated or latched on the rising edge of SD_CLK.<br>See the timing diagrams in Figure 14-6 and Figure 14-7. |
| SD_D[31:0] | IO<br>8mA | **SDRAM Data**<br>MSB is SD_D[31]. |
| SD_DQM[3:0] | O<br>8mA | **SDRAM Byte Enable Mask**<br>SD_DQM[0] is associated with the least significant byte. SD_DQM[3] is associated with the most significant byte. When a SD_DQM pin is high during a write cycle, the associated byte is not written to SDRAM. When a SD_DQM pin is high during a read cycle, the associated byte is not driven out of the SDRAM (the SD_D pins remain high-Z). |
| SD_A[11:0] | O<br>8mA | **SDRAM Address Bus**<br>MSB is SD_A[11]. |
| SD_BA[1:0] | O<br>8mA | **SDRAM Bank Select Outputs**<br>The external SDRAMs used by the device have their memory organized into four banks. These pins specify the bank to be accessed. The bank must be specified on the same SD_CLK edge that the row information is specified on SD_A[11:0]. |
| SD_CS_N | O<br>8mA | **SDRAM Chip Select** (Active Low)<br>Driven low by the device to initiate a memory access (read or write) to the external SDRAM. |
| SD_WE_N | O<br>8mA | **SDRAM Write Enable** (Active Low)<br>Driven low by the device when data is to be written to the external SDRAM. Left high when data is to be read from the external SDRAM. |
| SD_RAS_N | O<br>8mA | **SDRAM Row Address Strobe** (Active Low)<br>Driven low by the device during SD_CLK cycles in which SD_A[11:0] indicates the SDRAM row address. |
| SD_CAS_N | O<br>8mA | **SDRAM Column Address Strobe** (Active Low)<br>Driven low by the device during SD_CLK cycles in which SD_A[11:0] indicates the SDRAM column address. |

**Table 9-4. Ethernet PHY Interface Pins (MII/RMII/SSMII)**

The PHY interface type is configured by General_cfg_reg0.MII_mode_select[1:0]. 00=MII, 01=Reduced MII (RMII), 11=Source Synchronous Serial MII (SSMII). The MII interface is described in IEEE 802.3-2005 Section 22. The RMII interface is described in this document: http://www.national.com/appinfo/networks/files/rmii_1_2.pdf. The Source Synchronous Serial MII is described in this document: ftp://ftp-eng.cisco.com/smii/smii.pdf.

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| CLK_MII_TX | I | **MII Transmit Clock Input**<br>In MII mode a 25MHz clock must be applied to this pin to clock the transmit side of the interface. MII_TXD[3:0], MII_TX_EN and MII_TX_ERR are clocked out of the device on the rising edge of CLK_MII_TX. See the timing diagram in Figure 14-15.<br><br>In RMII mode a 50MHz clock must be applied to this pin to clock the transmit side of the interface. MII_TXD[3:2] and MII_TX_EN are clocked out of the device on the rising edge of CLK_MII_TX. See the timing diagram in Figure 14-17.<br><br>In SSMII mode, a 125MHz clock must be applied to this pin. This clock is the reference for the CLK_SSMII_TX output. |
| CLK_SSMII_TX | O<br>12ma | **SSMII Transmit Clock Output**<br>In SSMII mode, the device provides a 125MHz clock on this pin to clock the transmit side of the interface. MII_TXD[0] (SSMII_TXD) and MII_TXD[1] (SSMII_TX_SYNC) are clocked out of the device on the rising edge of CLK_MII_TX. See the timing diagram in Figure 14-19. This pin is not used in MII and RMII modes. |
| MII_TXD[3:0] | O<br>8mA | **MII Transmit Data Outputs**<br>In MII mode, transmit data is passed to the PHY four bits at a time on MII_TXD[3:0] on the rising edge of CLK_MII_TX. See the timing diagram in Figure 14-15.<br><br>In RMII mode, transmit data is passed to the PHY two bits at a time on MII_TXD[3:2] on the rising edge of CLK_MII_TX while MII_TXD[1:0] are not used. See the timing diagram in Figure 14-17.<br><br>In SSMII mode, transmit data is passed to the PHY one bit at a time on MII_TXD[0] (SSMII_TXD) on the rising edge of CLK_SSMII_TX. MII_TXD[1] (SSMII_TX_SYNC) indicates 10-bit segment alignment of the serial data stream. |
| MII_TX_EN | O<br>8mA | **MII Transmit Enable Output**<br>In MII mode and RMII, this pin serves as the transmit enable output. In SSMII mode this pin is not used. |
| MII_TX_ERR | O<br>8mA | **MI Transmit Error Output**<br>In MII mode this pin serves as the transmit error output. In RMII and SSMII modes this pin is not used. |
| CLK_MII_RX | I | **MII Receive Clock Input**<br>In MII mode a 25MHz clock must be applied to this pin. MII_RXD[3:0], MII_RX_DV, and MII_RX_ERR are clocked into the device on the rising edge of CLK_MII_RX. See the timing diagram in Figure 14-16.<br><br>In RMII mode this pin is not used, and a 50MHz clock applied to CLK_MII_TX provides timing for both transmit and receive sides of the interface. MII_RXD[3:2], MII_RX_DV and MII_RX_ERR are clocked into the device on the rising edge of CLK_MII_TX. See the timing diagram in Figure 14-18.<br><br>In SSMII mode a 125MHz clock from the PHY must be applied to this pin. MII_RXD[0] (SSMII_RXD) and MII_RXD[1] (SSMII_RX_SYNC) are clocked into the device on the rising edge of CLK_MII_RX. See the timing diagram in Figure 14-20. |

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| MII_RXD[3:0] | I | **MII Receive Data Inputs**<br>In MII mode, receive data comes from the PHY four bits at a time on MII_RXD[3:0], on the rising edge of CLK_MII_RX. See the timing diagram in Figure 14-16.<br><br>In RMII mode, receive data comes from the PHY two bits at a time on MII_RXD[3:2] and is latched on the rising edge of CLK_MII_TX. MII_RXD[1:0] are not used. See the timing diagram in Figure 14-18.<br><br>In SSMII mode, received data comes from the PHY one bit at a time on MII_RXD[0] (SSMII_RXD) on the rising edge of CLK_MII_RX. MII_RXD[1] (SSMII_RX_SYNC) indicates 10-bit segment alignment of the serial data stream. |
| MII_RX_DV | I | **MII Receive Data Valid Input**<br>In MII mode, this pin serves as the receive data valid input. In RMII mode, carrier sense and receive data valid alternate on this pin. See the RMII spec for details. In SSMII mode this pin is not used and should be pulled low or high. |
| MII_RX_ERR | I | **MII Receive Error Input**<br>In MII mode and RMII mode, this pin serves as the receive error input. In SSMII mode this pin is not used and should be pulled low or high. |
| MII_COL | I | **MII Collision Input**<br>In MII mode this pin serves as the collision detection input. In RMII mode and SSMII mode this pin is not used and should be pulled low or high. |
| MII_CRS | I | **MII Carrier Sense Input**<br>In MII mode this pin serves as the carrier sense input. In RMII mode and SMII mode this pin is not used and should be pulled low or high. |
| MDC | O<br>8mA | **PHY Management Clock Output**<br>This signal is the clock for the Ethernet PHY management interface, which consists of MDC and MDIO. See the timing diagram in Figure 14-14. |
| MDIO | IOpu<br>8mA | **PHY Management Data Input/Output**<br>This signal is the serial data signal for the Ethernet PHY management interface, which consists of MDC and MDIO. When MDIO is an output, it is updated on the rising edge of MDC. When MDIO is an input, it is latched into the device on the rising edge of MDC. See the timing diagram in Figure 14-14. |

**Table 9-5. Global Clock Pins**

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| CLK_SYS_S | Ipd | **System Clock Selection Input**<br>This pin specifies the frequency of the clock applied to the CLK_SYS pin. See section 10.4.<br>0 = 50 or 75 MHz<br>1 = 25 MHz |
| CLK_SYS | I | **System Clock Input**<br>A 25 MHz, 50 MHz or 75 MHz clock (±50 ppm or better) must be applied to this pin to clock TDM-over-Packet internal circuitry and the SDRAM interface (SD_CLK). When a 25MHz clock is applied, it is internally multiplied by the CLAD2 block to 50MHz or 75MHz as specified by GCR1.SYSCLKS. The CLK_SYS_S pin specifies whether the CLK_SYS signal is 25MHz (and therefore needs to multiplied up) or 50/75MHz (and therefore is used as-is). See section 10.4. |
| CLK_CMN | I | **Common Clock Input**<br>When the TDMoP engine is configured for common clock mode (also known as differential mode), the common clock is applied to this pin. This clock signal has to be a multiple of 8kHz and in the range of 1MHz to 25MHz. The frequency should not be too close to an integer multiple of the service clock frequency. Based on these criteria, the following frequencies are suggested:<br>For systems with access to a common SONET/SDH network, a frequency of 19.44 MHz (2430*8 kHz).<br>For systems with access to a common ATM network, 9.72 MHz (1215*8 kHz) or |

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| | | 19.44 MHz (2430*8 kHz).<br>For systems using GPS, 8.184 MHz (1023*8 kHz).<br>For systems connected by a single hop of 100 Mbit/s Ethernet where it is possible to lock the physical layer clock, 25 MHz (3125*8 kHz).<br>For systems connected by a single hop of Gigabit Ethernet where it is possible to lock to the physical layer clock, 10MHz (1250*8 kHz).<br>When a clock is not needed on this pin, pull it high or low. See section 10.4. |
| CLK_HIGH | I | **Clock High Input**<br>A 10, 19.44, 38.88 or 77.76MHz clock can be applied to this pin. From the CLK_HIGH signal, an on-chip frequency converter block (called a clock adapter or CLAD, in this case CLAD1) produces the 38.88MHz reference clock required by the clock recovery machines in the TDMoP block.<br>GCR1.FREQSEL specifies the frequency of the clock applied to CLK_HIGH.<br>When GCR1.CLK_HIGHD=1, the CLAD disables the 38.88MHz reference clock to the clock recovery machines.<br>When clock recovery is not required (i.e. when none of the recovered clock outputs TDMn_ACLK are used), CLK_HIGH can be held low.<br>The required quality of the CLK_HIGH signal is discussed in section 10.6.3. |

**Table 9-6. CPU Interface Pins**

See the parallel interface timing diagrams in Figure 14-2 and Figure 14-3 and the SPI timing diagrams in Figure 14-4 and Figure 14-5.

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| H_CPU_SPI_N | Ipu | **Host Bus Interface**<br>0 = SPI serial interface<br>1 = Parallel interface |
| DAT_32_16_N | Ipu | **Data Bus Width**<br>0 = 16-bit<br>1 = 32-bit<br>In SPI bus mode this pin is ignored. |
| H_D[31:1] | IO<br>8mA | **Host Data Bus**<br>When the device is configured for a 32-bit parallel interface, H_D[31:0] are the data I/O pins (HD[31] is the MSb). When the device is configured for a 16-bit parallel interface, H_D[15:0] are the data I/O pins (HD[15] is the MSb) and H_D[31:16] are ignored and should be pulled low or high. The DAT_32_16_N pin specifies bus width. In SPI bus mode these pins are ignored. |
| H_D[0] /<br>SPI_MISO | IO<br>8mA | **H_D[0]: Host Data LSb**<br>In parallel interface mode this pin is H_D[0], LSb of the data bus.<br><br>**SPI_MISO: SPI Data Output (Master In Slave Out)**<br>In SPI bus mode this pin is the SPI data output. |
| H_AD[24:1] | I | **Host Address Bus**<br>H_AD[24] is the MSb. When the host data bus is 32 bits (DAT_32_16_N=1), H_AD[1] should be held low. In SPI bus mode these pins are ignored. |
| H_CS_N | I | **Host Chip Select (Active Low)**<br>In parallel interface mode this pin must be asserted (low) to read or write internal registers. In SPI bus mode this pin is ignored. |
| H_R_W_N /<br>SPI_CP | I | **H_R_W_N**: **Host Read/Write Control**<br>In parallel interface mode this pin controls whether an access to internal registers is a read or a write.<br><br>**SPI_CP: SPI Clock Phase**<br>In SPI interface mode this pin specifies SPI clock phase. See the timing diagrams in Figure 14-4 and Figure 14-5 for details.<br>0 = input data is latched on the leading edge of the SCLK pulse; output data is updated on the trailing edge |

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| | | 1 = input data is latched on the trailing edge of the SCLK pulse; output data is updated on the leading edge |
| H_WR_BE0_N / SPI_CLK | I | **H_WR_BE0_N: Host Write Enable Byte 0 (Active Low)**<br>In parallel interface mode during a write access this pin specifies whether or not byte 0 (H_D[7:0]) should be written to the device. This pin is active in both 32-bit and 16-bit modes.<br>0 = write byte 0<br>1 = don't write byte 0<br><br>**SPI_CLK: SPI Clock**<br>In SPI interface mode this pin is the clock for the interface. |
| H_WR_BE1_N / SPI_MOSI | I | **H_WR_BE1_N: Host Write Enable Byte 1 (Active Low)**<br>In parallel interface mode during a write access this pin specifies whether or not byte 1 (H_D[15:8]) should be written to the device. This pin is active in both 32-bit and 16-bit modes.<br>0 = write byte 1<br>1 = don't write byte 1<br><br>**SPI_MOSI: SPI Data Input (Master Out Slave In)**<br>In SPI interface mode this pin is the data input pin for the interface. |
| H_WR_BE2_N / SPI_SEL_N | I | **H_WR_BE2_N: Host Write Enable Byte 2 (Active Low)**<br>In 32-bit parallel interface mode during a write access this pin specifies whether or not byte 2 (H_D[15:8]) should be written to the device. In 16-bit parallel interface mode this pin is ignored and should be pulled high or low.<br>0 = write byte 2<br>1 = don't write byte 2<br><br>**SPI_SEL: SPI Chip Select (Active Low)**<br>In SPI interface mode this pin must be asserted (low) to read or write internal registers. |
| H_WR_BE3_N / SPI_CI | I | **H_WR_BE3_N: Host Write Enable Byte 3 (Active Low)**<br>In 32-bit parallel interface mode during a write access this pin specifies whether or not byte 3 (H_D[15:8]) should be written to the device. In 16-bit parallel interface mode this pin is ignored and should be pulled high or low.<br>0 = write byte 3<br>1 = don't write byte 3<br><br>**SPI_CI: SPI Clock Invert**<br>In SPI interface mode this pin specifies the polarity of the SPI_CLK pin. See the timing diagrams in Figure 14-4 and Figure 14-5 for details.<br>0 = SPI_CLK is normally low and pulses high (leading edge is rising edge)<br>1 = SPI_CLK is normally high and pulses low (leading edge is falling edge) |
| H_READY_N | O<br>8mA | **Host Ready Output (Active Low)**<br>In parallel interface mode the device pulls this pin low during a read or write access to signal that the device is ready for the access to be completed. The host processor should not pull H_CS_N high (inactive) to complete the access until the device has pulled H_READY_N low.<br>This pin requires the use of an external pull-up resistor. The device actively drives this pin high before allowing it to go high-impedance. See Figure 14-2. |
| H_INT | O<br>8mA | **Host Interrupt Output (Active Low)**<br>This pin indicates interrupt requests from the device. When GCR1.IPI=1, H_INT is forced high (inactive). See section 10.9. |

**Table 9-7. JTAG Interface Pins**

See the JTAG interface timing diagram in Figure 14-21.

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| JTRST_N | Ipu | **JTAG Test Reset (Active Low)**<br>This signal is used to asynchronously reset the test access port controller. After power up, JTRST_N must be toggled from low to high. This action sets the device into the JTAG DEVICE ID mode. Pulling JTRST_N low restores normal device operation. If boundary scan is not used, this pin should be held low. |
| JTCLK | I | **JTAG Test Clock**<br>This signal is used to shift data into JTDI on the rising edge and out of JTDO on the falling edge. |
| JTMS | Ipu | **JTAG Test Mode Select**<br>This pin is sampled on the rising edge of JTCLK and is used to place the test access port into the various defined IEEE 1149.1 states. If not used, JTMS should be held high. |
| JTDI | Ipu | **JTAG Test Data Input**<br>Test instructions and data are clocked into this pin on the rising edge of JTCLK. If not used, JTDI can be held low or high (DVDDIO). |
| JTDO | Oz<br>8mA | **JTAG Test Data Output**<br>Test instructions and data are clocked out of this pin on the falling edge of JTCLK. If not used, this pin should be left unconnected. |

**Table 9-8. Reset and Factory Test Pins**

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| RST_SYS_N | Ipu | **System Reset (Active Low)**<br>When this pin is held low the entire device is reset. This pin should be held low (active) for at least 200 µs before going inactive. CLK_SYS and CLK_HIGH should be stable for at least 200 µs before RST_SYS_N goes inactive. See section 10.5 for more information on system resets and block-level resets. |
| HIZ_N | I | **High Impedance Enable (Active Low)**<br>When this signal is low while JTRST_N is low, all of the digital output and bi-directional pins are placed in the high impedance state. For normal operation this signal is high. This is an asynchronous input. |
| SCEN | I | Used during factory test. This pin should be tied to DVSS. |
| STMD | I | Used during factory test. This pin should be tied to DVSS. |
| MBIST_EN | I | Used during factory test. This pin should be tied to DVSS. |
| MBIST_DONE | O | Used during factory test. This pin should be left floating. |
| MBIST_FAIL | O | Used during factory test. This pin should be left floating. |
| TEST_CLK | O | Used during factory test. This pin should be left floating. |
| TST_CLD | I | Used during factory test. This pin should be tied to DVSS. |

**Table 9-9. Power and Ground Pins**

| PIN NAME[1] | TYPE[2] | PIN DESCRIPTION |
|---|---|---|
| DVDDC | P | 1.8V Core Voltage for TDM-over-Packet Digital Logic (17 pins) |
| DVDDIO | P | 3.3V for I/O Pins (16 pins) |
| DVSS | P | Ground for TDM-over-Packet Logic and I/O Pins (31 pins) |
| ACVDD1 | P | 1.8V for CLAD Analog Circuits |
| ACVDD2 | P | 1.8V for CLAD Analog Circuits |
| ACVSS1 | P | Ground for CLAD Analog Circuits |
| ACVSS2 | P | Ground for CLAD Analog Circuits |

# 10. Functional Description

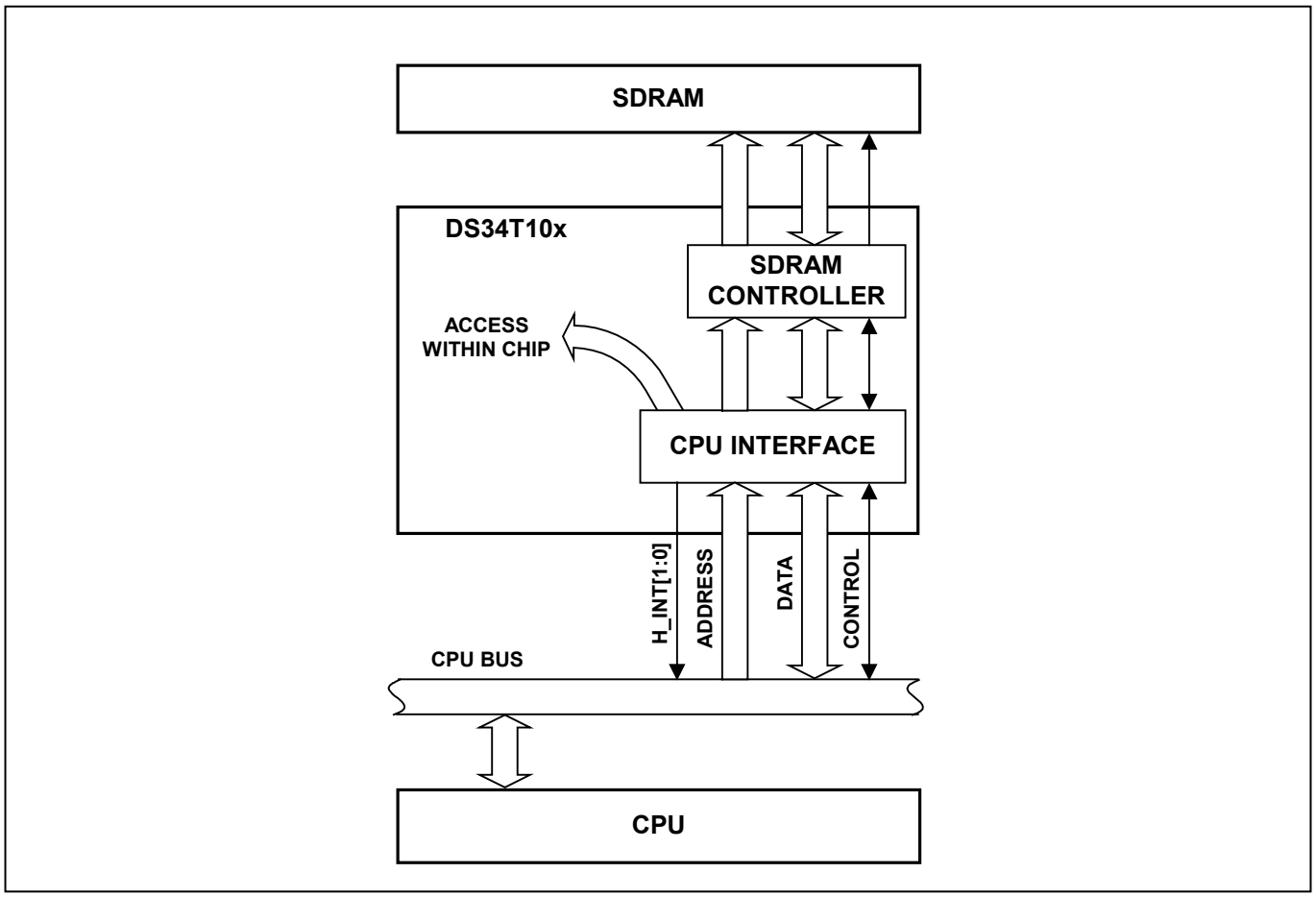
## 10.1 Power-Supply Considerations

Due to the dual-power-supply nature of the device, some I/Os have parasitic diodes between a 1.8V supply and a 3.3V supply. When ramping power supplies up or down, care must be taken to avoid forward-biasing these diodes because it could cause latchup. Two methods are available to prevent this. The first method is to place a Schottky diode external to the device between the 1.8V supply and the 3.3V supply to force the 3.3V supply to be within one parasitic diode drop below the 1.8V supply (i.e. VDD3.3 > VDD1.8 – 0.4V). The second method is to ramp up the 3.3V supply first and then ramp up the 1.8V supply.

## 10.2 CPU Interface

The CPU interface enables an external CPU to configure and control the device and collect statistics from the device. The CPU interface block identifies accesses (read or write) to on-chip registers and to external SDRAM, forwards accesses to the proper place, and replies to the CPU with the requested data during read accesses. See Figure 10-1. AC timing for the CPU interface is specified in section 14.1.

*Figure 10-1. CPU Interface Functional Diagram*




To configure the device for CPU interface mode, the H_CPU_SPI_N pin must be high when the RST_SYS_N (system reset) pin is deasserted. The chip can be configured for 16-bit or 32-bit data bus width by wiring the DAT_32_16_N pin as shown in Table 10-1:

**Table 10-1. CPU Data Bus Widths**

| DAT_32_16_N Value | Data Bus Width | Access to Chip Internal Resources | Access to SDRAM | Data Bus Bits | MSB | H_WR_BE Pins Used |
|---|---|---|---|---|---|---|
| 1 | 32 bits | 32 bit only | 8, 16, 32 bit | H_D[31:0] | H_D[31] | 3:0 |
| 0 | 16 bits | 16 bit only | 8, 16 bit | H_D[15:0] | H_D[15] | 1:0 |

Burst accesses are not supported. The device uses the big-endian byte order, as explained in section 11.1.

The CPU starts an access to the device by asserting the H_CS_N signal (active low), accompanied by the desired read/write state on H_R_W_N, address on H_AD[24:1], write byte enables on the H_WR_BE pins and valid data (for a write access) on the H_D[31:0] pins. In response, the device asserts H_READY_N to indicate that the access has been carried out. The ready assertion indicates that data from the CPU has been written into the device register or external SDRAM (for write access) or that valid data from register/SDRAM is present on the data bus (for read access). In response to H_READY_N assertion, the CPU de-asserts H_CS_N. This causes the chip to de-assert H_READY_N, and thereby finish the CPU access.

In order to make CPU operation more efficient, the device immediately asserts H_READY_N during a write access. On successive accesses (write or read) H_READY_N is asserted only after the previous write has been completed.

In 32-bit bus mode, H_WR_BE0_N through H_WR_BE3_N serve as write byte enable signals, replacing the functionality of H_AD[1:0] in the address bus. In 16-bit bus mode, H_WR_BE0_N and H_WR_BE1_N serve as write byte enables, replacing the functionality of H_AD[0] in the address bus. These signals enable byte-resolution write access to the external SDRAM.

When performing a write access to internal chip resources, all H_WR_BE pins should be asserted since write access to device registers must be done at the full bus width only.

Examples of read and write accesses on 32- and 16-bit buses are shown in the figures below.

*Figure 10-2. Write Access, 32-Bit Bus*

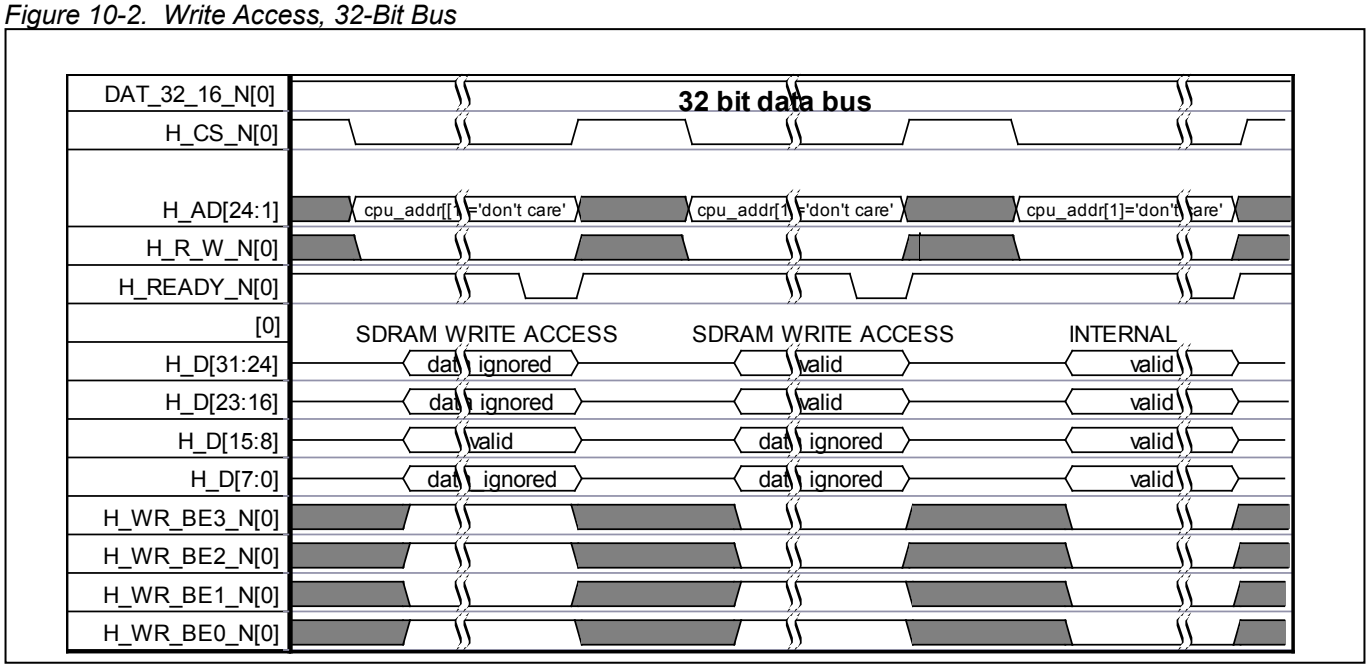



Figure 10-2 shows two write accesses to the SDRAM, one to a byte (at address 2) and the other to a word (at addresses 0 and 1), followed by a write access to the internal chip resources.

The write access to the SDRAM is different than the write access to the chip. The SDRAM can be written with byte resolution using the four byte write enables. In contrast, internal chip resources are always written at full CPU data bus width (32 bits in Figure 10-2). The write byte enable signals should always be asserted when writing to internal device registers.

For 32-bit CPU bus width, H_AD[1] is ignored, since accesses are always on an even 4-byte boundary.

Figure 10-3 shows a read access to the SDRAM followed by a read access to the internal chip resources. Read accesses always occur at CPU data bus width and the H_WR_BE pins are not used (and must be held high). Bytes that are not needed by the CPU can be ignored.

*Figure 10-3. Read Access, 32-Bit Bus*

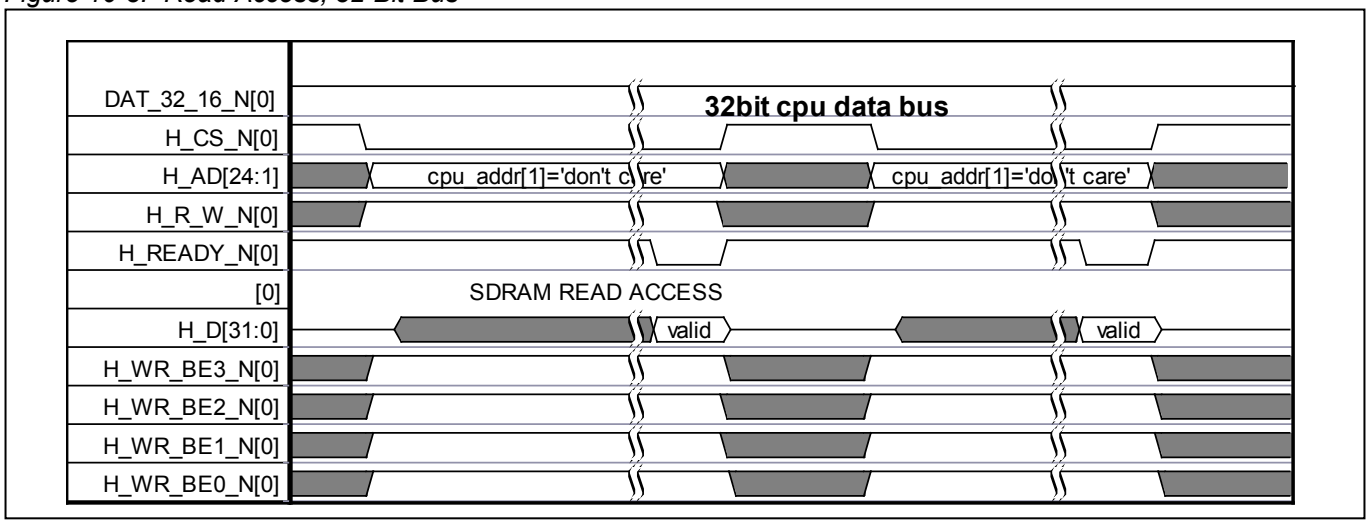



Figure 10-4 shows a write access to the chip followed by a read access in 16-bit bus mode. In this mode the H_AD[1] signal is used because accesses are on an even 2-byte boundary. Write access to the SDRAM can still be at byte resolution, as illustrated in Figure 10-5.

*Figure 10-4. Read/Write Access, 16-Bit Bus*

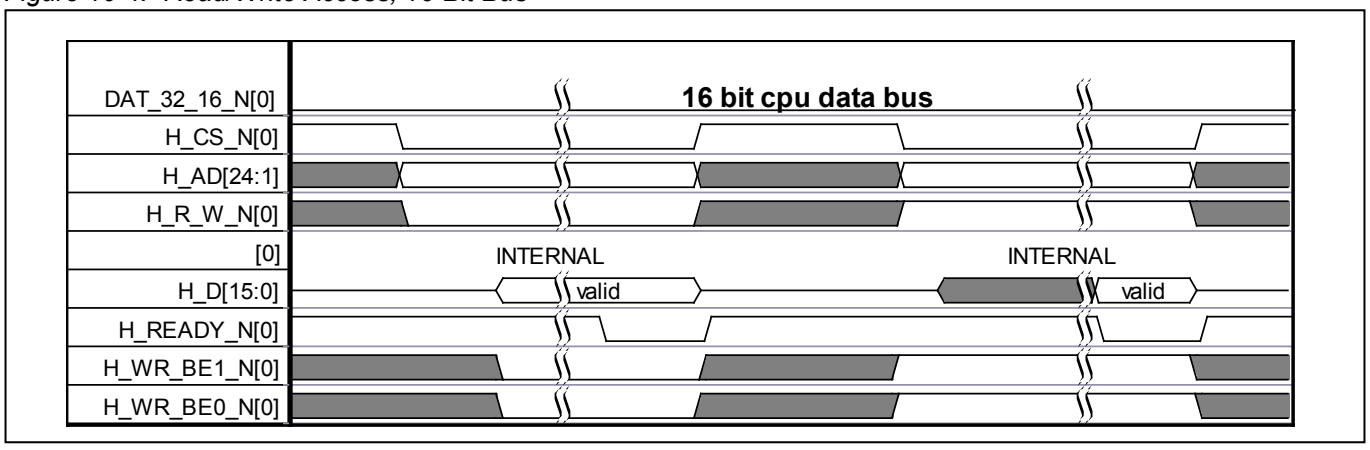
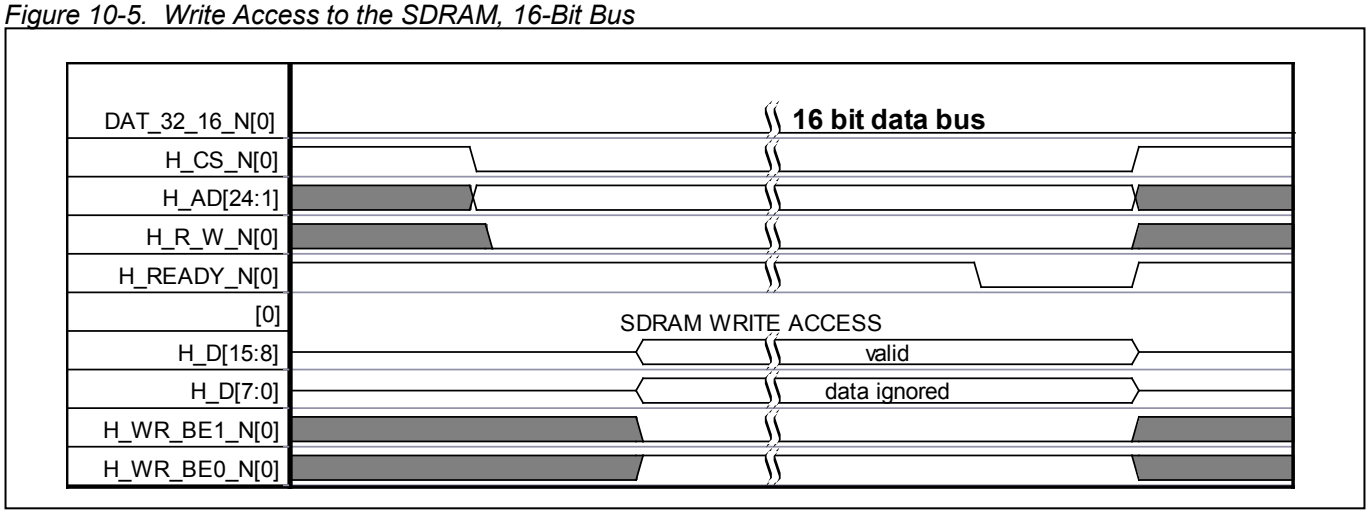
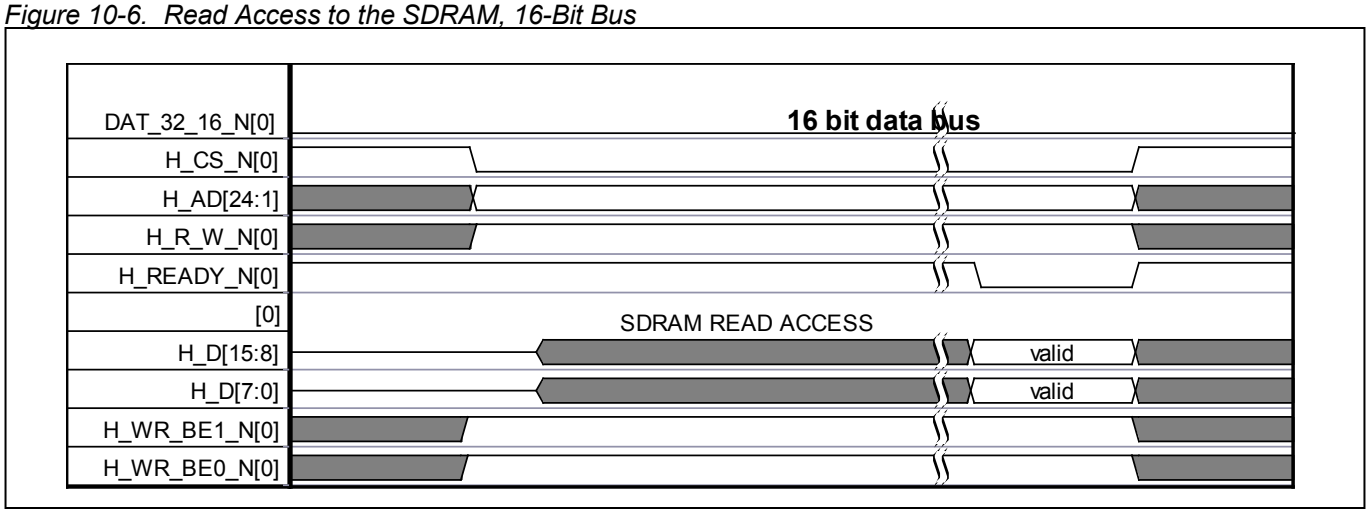


*Figure 10-5. Write Access to the SDRAM, 16-Bit Bus*




In 16-bit bus mode, read accesses to SDRAM are always 16 bits, as in Figure 10-6.

*Figure 10-6. Read Access to the SDRAM, 16-Bit Bus*




## 10.3 SPI Interface

The device optionally can be accessed by an external CPU through a Serial Peripheral Interface (SPI). To configure the device for SPI interface mode, the H_CPU_SPI_N pin must be low when the RST_SYS_N (system reset) pin is deasserted. In SPI mode, some of the parallel CPU bus pins take on an SPI-related function while the rest are disabled. See the CPU interface section of Table 9-1 for details. The device functions as an SPI slave.

### 10.3.1 SPI Operation

The SPI is a 4-wire, full-duplex, synchronous interface. The SPI connects an SPI master (which initiates the data transfer) and an SPI slave.

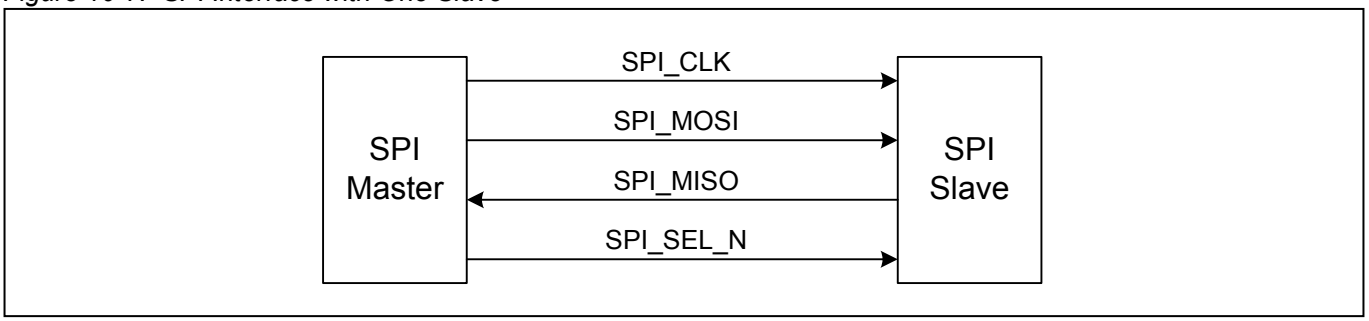
The SPI signal wires are as follows:

- SPI_CLK is the clock for the serial data (gated clock).
- SPI_MOSI is master data output, slave data input.

- SPI_MISO is master data input, slave data output.
- SPI_SEL_N is the slave chip select.

The master initiates a data transfer by asserting SPI_SEL_N (low) and generating a sequence of SPI_CLK cycles accompanied by serial data on SPI_MOSI. During read cycles the slave outputs data on SPI_MISO. Each additional slave requires an additional slave chip-select wire. Figure 10-7 illustrates a typical connection between an SPI master and a single SPI slave.

*Figure 10-7. SPI Interface with One Slave*




### 10.3.2 SPI Modes

Two configuration pins define the SPI mode of operation.

- The polarity of SPI_CLK is specified by the SPI_CI (clock invert) input pin.
- The SPI_CP (clock phase) input pin determines whether the first SPI_CLK transition is used to sample the data on SPI_MISO/SPI_MOSI (which requires the first bit to be ready beforehand on these lines) or to updated the data on the SPI_MISO/SPI_MOSI lines. See Figure 10-8 and Figure 10-9.

*Figure 10-8. SPI Interface Timing, SPI_CP=0*

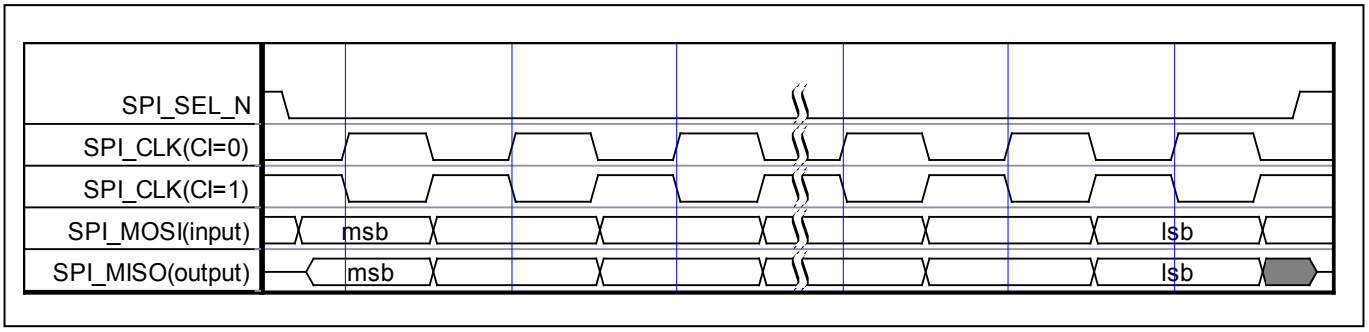
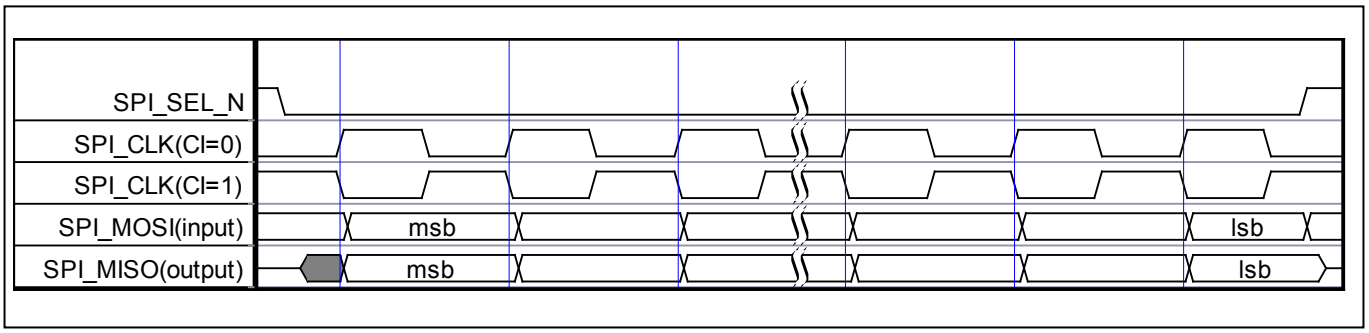



*Figure 10-9. SPI Interface Timing, SPI_CP=1*



### 10.3.3 SPI Signals

In SPI mode, the following CPU bus pins change their functionality and operate as SPI signals.

- Inputs
    - SPI_CLK is shared with H_WR_BE0_N
    - SPI_MOSI is shared with H_WR_BE1_N
    - SPI_SEL_N is shared with H_WR_BE2_N.
- Outputs
    - SPI_MISO is shared with H_D[0].

The SPI configuration is supplied on two external pins as follows:

- SPI_CI (clock invert) is shared with H_WR_BE3_N
- SPI_CP (clock phase) is shared with H_R_W_N.

In the SPI mode the device operates internally in 32-bit mode.

### 10.3.4 SPI Protocol

The external CPU communicates with the device over SPI by issuing commands. There are three command types:

1. Write – performs 32-bit write access
2. Read – performs 32-bit read access
3. Status – verifies that previous access has been finished

The SPI_SEL_N signal must be de-asserted between accesses to the device.

#### 10.3.4.1 Write Command

The SPI write command proceeds as follows:

- The SPI master (CPU) starts a write access by asserting SPI_SEL_N (low).
- Then, during each SPI_CLK cycle a SPI_MOSI data bit is transmitted by the master (CPU), while a SPI_MISO bit is transmitted by the slave (the device).
- The first bit on SPI_MOSI and SPI_MISO is reserved (don’t care).
- The master then transmits two opcode bits on SPI_MOSI. These bits specify a read, write or status command. The value 01b represents a write command. At the same time, the slave transmits the opcode bits of the previous command on SPI_MISO.
- The next four bits the master transmits on SPI_MOSI are byte-enable values: byte_en_3, byte_en_2, byte_en_1, and byte_en_0 which are equivalent to the function of the H_WR_BE3_N to H_WR_BE0_N

signals in CPU bus mode (including being active low). At the same time, the slave transmits the byte enable values of the previous access on SPI_MISO.

- The next bit on SPI_MOSI and SPI_MISO is reserved (don't care).
- The next 24 bits the master transmits on SPI_MOSI are address bits, starting from A24 (MSB) and ending with A1 (LSB). At the same time, the slave transmits the address bits of the previous access on SPI_MISO.
- The next 32 bits the master transmits on SPI_MOSI are 32 bits of data, starting from D31 (MSB) and ending with D0 (LSB). At the same time, the slave transmits 32 don't-care bits on SPI_MISO.
- Finally the master transmits 8 don't care bits on SPI_MOSI. During these clock periods the slave transmits 8 bits on SPI_MISO. The first 7 SPI_MISO bits are don't-care. The 8th bit is a status bit that indicates whether the last access was completed successfully (1) or is still in progress (0). The 0 value indicates that the current operation has not yet completed and that the status command must follow (see section 10.3.4.3).
- The master ends the write access by deasserting SPI_SEL_N.

The total number of SPI_CLK cycles for a write command is 72. This is summarized in Table 10-2.

**Table 10-2. SPI Write Command Sequence**

| Bit Number | SPI_MOSI | SPI_MISO |
|---|---|---|
| 1 | Reserved | Reserved |
| 2–3 | opcode 01 (write) | Previous access opcode |
| 4 | H_WR_BE3_N value | Previous access H_WR_BE3_N value |
| 5 | H_WR_BE2_N value | Previous access H_WR_BE2_N value |
| 6 | H_WR_BE1_N value | Previous access H_WR_BE1_N value |
| 7 | H_WR_BE0_N value | Previous access H_WR_BE0_N value |
| 8 | Reserved | Reserved |
| 9–32 | Address [24 to 1] | Previous access address [24 to 1] |
| 33–64 | Data (32 bits) | Don't care (32 bits) |
| 65–71 | Don't care (7 bits) | Idle (7 bits) |
| 72 | Don't care (1 bit) | Status bit: 1=access has finished, 0=access has not finished |

#### *10.3.4.2 Read Command*

The SPI read command proceeds as follows:

- The SPI master (CPU) starts a write access by asserting SPI_SEL_N (low).
- Then, during each SPI_CLK cycle a SPI_MOSI data bit is transmitted by the master (CPU), while a SPI_MISO bit is transmitted by the slave (the device).
- The first bit on SPI_MOSI and SPI_MISO is reserved (don't care).
- The master then transmits two opcode bits on SPI_MOSI. These bits specify a read, write or status command. The value 10b represents a read command. At the same time, the slave transmits the opcode bits of the previous command on SPI_MISO.
- The next four bits the master transmits on SPI_MOSI are byte-enable values: byte_en_3, byte_en_2, byte_en_1, and byte_en_0 which are equivalent to the function of the H_WR_BE3_N to H_WR_BE0_N

signals in CPU bus mode (including being active low). For a read access, all four of these bits should be 1. At the same time, the slave transmits the byte enable values of the previous access on SPI_MISO.

- The next bit on SPI_MOSI and SPI_MISO is reserved (don't care).
- The next 24 bits the master transmits on SPI_MOSI are address bits, starting from A24 (MSB) and ending with A1 (LSB). At the same time, the slave transmits the address bits of the previous access on SPI_MISO.
- Next the master transmits 8 don't care bits on SPI_MOSI. During these clock periods the slave transmits 8 bits on SPI_MISO. The first 7 SPI_MISO bits are don't-care. The 8th bit is a status bit that indicates whether the current read access was completed successfully (1) or is still in progress (0). Status=0 indicates that the current operation has not yet completed and that the status command must follow (see section 10.3.4.3).
- Status=1 indicates that the current read was completed successfully and 32 bits of data follow on SPI_MISO, starting from D31 (MSB) and ending with D0 (LSB). During these 32 clock cycles, the master transmits 32 don't-care bits on SPI_MOSI.
- Status=0 indicates that the current read was not completed and that the status command must follow (see section 10.3.4.3). During the next 32 clock cycles both the master and the slave must transmit don't-care bits to complete the read command. These 32 bits should be ignored.
- The master ends the write access by deasserting SPI_SEL_N.

The total number of SPI_CLK cycles for a read command is 72.

**Table 10-3. SPI_ Read Command Sequence**

| Bit Number | SPI_MOSI | SPI_MISO |
|---|---|---|
| 1 | Reserved | Reserved |
| 2–3 | opcode 10 (read) | Previous access opcode |
| 4 | 1 | Previous access H_WR_BE3_N value |
| 5 | 1 | Previous access H_WR_BE2_N value |
| 6 | 1 | Previous access H_WR_BE1_N value |
| 7 | 1 | Previous access H_WR_BE0_N value |
| 8 | Reserved | Reserved |
| 9–32 | Address [24 to 1] | Previous access Address [24 to 1] |
| 33–39 | Don't care | Idle (7 bits) |
| 40 | Don't care | Status bit: 1=access has finished, 0=access has not finished |
| 41–72 | Don't care | Data (32 bits) |

#### *10.3.4.3 Status Command*

The status command differs from read or write commands, since it does not initiate an internal access. Usually a status command follows a read or write command that was not completed as described above.

The SPI status command proceeds as follows:

- The SPI master (CPU) starts a status command by asserting SPI_SEL_N (low).
- Then, during each SPI_CLK cycle a SPI_MOSI data bit is transmitted by the master (CPU), while a SPI_MISO bit is transmitted by the slave (the device).

- The first bit on SPI_MOSI and SPI_MISO is reserved (don't care).
- The master then transmits two opcode bits on SPI_MOSI. These bits specify a read, write or status command. The value 00b represents a status command. At the same time, the slave transmits the opcode bits of the previous command on SPI_MISO.
- The master then transmits 4 don't care bits on SPI_MOSI. During these clock periods the slave transmits 4 bits on SPI_MISO. The first 3 SPI_MISO bits are don't-care. The 4th bit is a status bit that indicates whether the last access was completed successfully (1) or is still in progress (0). The 0 value indicates that the last access has not yet completed and that another status command must follow (see section 10.3.4.3).
- Status=1 indicates that the last access was completed successfully. If the last access was a read then 32 bits of data follow on SPI_MISO, starting from D31 (MSB) and ending with D0 (LSB). During these 32 clock cycles, the master transmits 32 don't-care bits on SPI_MOSI. If the last access was a write the during the next 32 clock cycles both the master and the slave must transmit don't-care bits to complete the status command. These 32 bits should be ignored.
- Status=0 indicates that the last access was not completed and that another status command must follow. During the next 32 clock cycles both the master and the slave must transmit don't-care bits to complete the status command. These 32 bits should be ignored.
- The master ends the write access by deasserting SPI_SEL_N.

The total number of SPI_CLK cycles for a status command is 40.

**Table 10-4. SPI Status Command Sequence**

| Bit Number | SPI_MOSI | SPI_MISO |
|---|---|---|
| 1 | Don't care | Don't care |
| 2–3 | opcode 00 (status) | Previous access opcode |
| 4 | Don't care | Don't care |
| 5 | Don't care | Don't care |
| 6 | Don't care | Don't care |
| 7 | Don't care | Don't care |
| 8 | Don't care | Status bit: 1=access has finished, 0=access has not finished |
| 9–40* | Don't care* | Data* |

* only if previous access was a read (previous access opcode = 10b).

## 10.4 Clock Structure

When clock recovery is enabled (Clock_recovery_en=1 in General_cfg_reg0), the clock recovery machines of the TDM-over-packet block require a 38.88MHz clock. This clock can come directly from the CLK_HIGH pin, or the CLAD1 block (see Figure 6-1) can convert a 10MHz, 19.44MHz or 77.76MHz clock on CLK_HIGH to 38.88MHz using an analog PLL. The frequency of CLK_HIGH must be specified in GCR1.FREQSEL.

When common clock (differential) mode is enabled (RTP_timestamp_generation_mode=1 in General_cfg_reg1), the clock recovery block requires a clock on the CLK_CMN pin *in addition to* the clock on the CLK_HIGH pin. See the CLK_CMN pin description for recommendations for the frequency of this clock. Often the same clock signal can be applied to both CLK_CMN and CLK_HIGH, for example 19.44MHz.

When clock recovery is disabled (Clock_recovery_en=0 in General_cfg_reg0), CPU software can disable the 38.88MHz clock output from CLAD1 to save power by setting GCR1.CLK_HIGHD. Clock recovery must be enabled whenever the TDMoP block must recover one or more service clocks from received packets using either adaptive mode or common clock (differential) mode.

The TDM-over-packet block also requires a 50 MHz or 75 MHz clock (±50 ppm or better) to clock its internal circuitry and the SDRAM interface (SD_CLK). When the CLK_SYS_S pin is low, a 50 MHz or 75 MHz clock applied to the CLK_SYS pin is passed directly to the TDMoP block. When the CLK_SYS_S pin is high, a 25 MHz clock on the CLK_SYS pin is internally multiplied by an analog PLL in the CLAD2 block to either 50 MHz or 75 MHz as specified by GCR1.SYSCLKS.

## 10.5 Reset and Power-Down

A hardware reset is issued by forcing the RST_SYS_N pin low. This pin resets the TDM-over-Packet block and the MAC. Note that not all registers are cleared to 0x00 on a reset condition. The register space must be reinitialized to appropriate values after hardware or software reset has occurred. This includes setting reserved locations to 0. Several block-specific resets are also available, as shown in Table 10-5.

**Table 10-5. Reset Functions**

| RESET FUNCTION | LOCATION | COMMENTS |
|---|---|---|
| Hardware Device Reset | RST_SYS_N Pin | Transition to a 200us or more logic 0 level resets the device. CLK_SYS and CLK_HIGH/MCLK are recommended to be stable 200us before transitioning out of reset. |
| Hardware JTAG Reset | JTRST_N Pin | Resets the JTAG test port. |
| Resets TDMoP TX, RX paths | Rst_reg | Used to reset the transmit (TX) and receive (RX) paths of the TDM-over-Packet block. |
| Resets the SDRAM controller | General_cfg_reg0 | The Rst_SDRAM_n bit resets the SDRAM controller. |

## 10.6 TDM-over-Packet Block

### 10.6.1 Packet Formats

To transport TDM data through packet switched networks, the TDM-over-Packet block encapsulates the TDM data into Ethernet packets as depicted in Figure 10-10.

*Figure 10-10. TDM-over-Packet Encapsulation Formats*

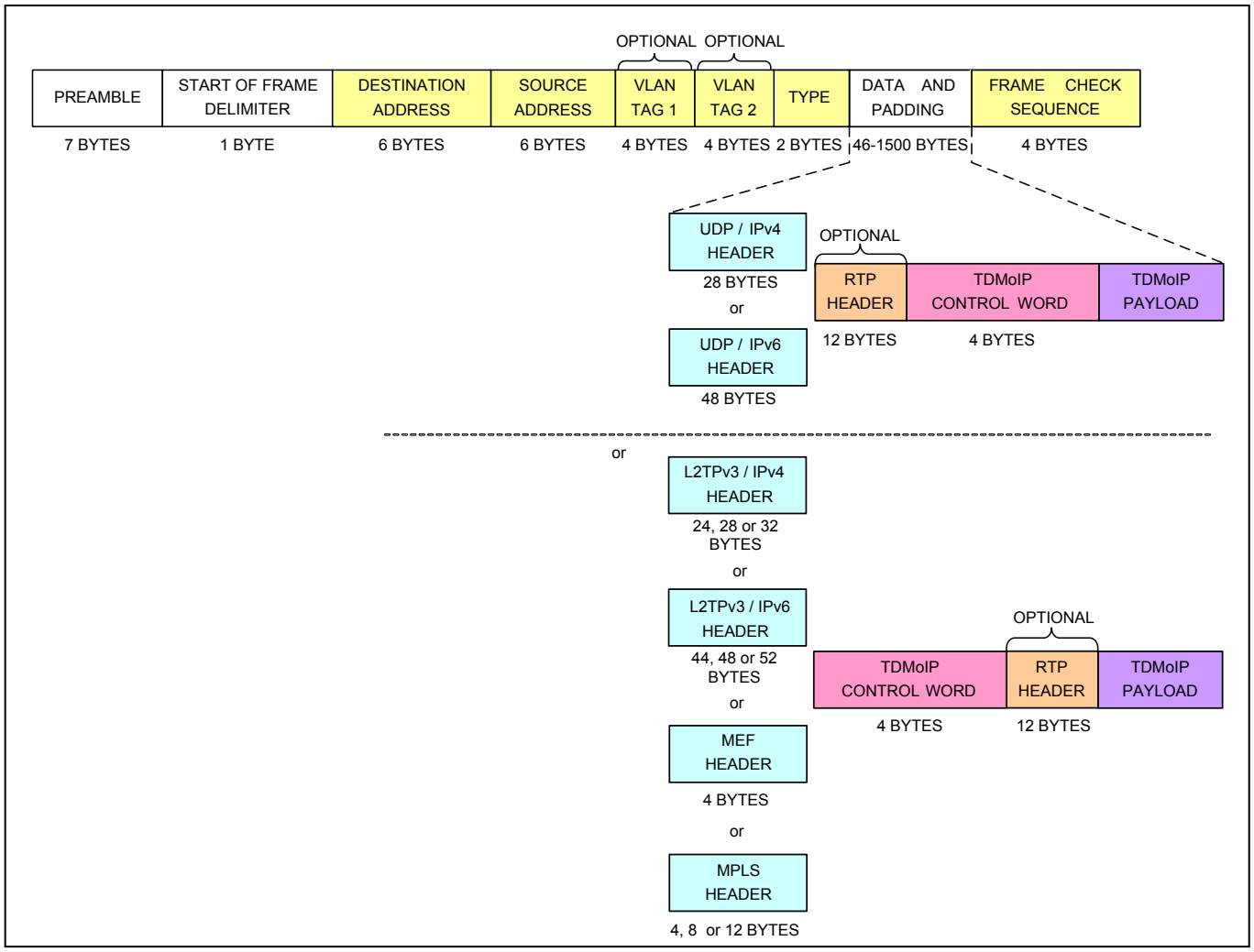



**Table 10-6. Ethernet Packet Fields**

| Field | Description |
|---|---|
| Preamble | A sequence of 56 bits (alternating 1 and 0 values) Gives components in the network time to detect the presence of a signal and synchronize to the incoming bit stream. |
| Start of Frame Delimiter (SFD) | A sequence of 8 bits (10101011) that indicates the start of the packet. |
| Destination Address and Source Address | The Destination Address field identifies the station or stations that are to receive the packet. The Source Address identifies the station that originated the packet. A Destination Address may specify either an individual address destined for a single station, or a multicast address destined for a group of stations. A Destination Address of all 1s refers to all stations on the LAN and is called the broadcast address. |
| Type | Ethertype. The type of payload contained in the Ethernet frame. |
| Data and Padding | This field contains the payload data transferred from the source station to the destination station(s). The maximum size of this field is 1500 bytes. If the payload to be transported is less than 46 bytes, then padding is used to bring the packet size up to the minimum length. A minimum-length Ethernet packet is 64 bytes from the Destination Address field through the Frame Check Sequence. |
| Frame Check Sequence (FCS) | This field contains a 4-byte cyclical redundancy check (CRC) value used for error checking. When a source station assembles a packet, it performs a CRC calculation on all the bits in the packet from the Destination Address through the Pad fields (that is, all fields except the preamble, start frame delimiter, and frame check sequence). The source station stores the calculated value in the FCS field and transmits it as part of the packet. When the packet is received by the destination station, it performs an identical check. If the calculated value does not match the value in the FCS field, the destination station assumes an error has occurred during transmission and discards the packet. |

#### 10.6.1.1 VLAN Tag

As specified in IEEE Standard 802.1q, the twelve-bit VLAN identifier's tagged packets, enables the construction of a maximum of 4,096 distinct VLANs. For cases where this VLAN limit is inadequate VLAN stacking provides a two-level VLAN tag structure, which extends the VLAN ID space to over 16 million VLANs. Each packet may be sent without VLAN tags, with a single VLAN tag or with two VLAN tags (VLAN stacking).

*Figure 10-11. Single VLAN Tag Format*

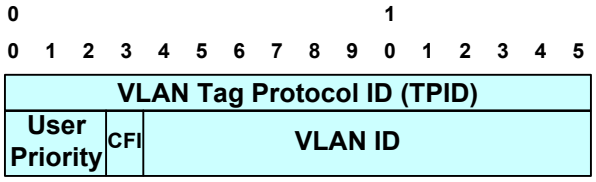
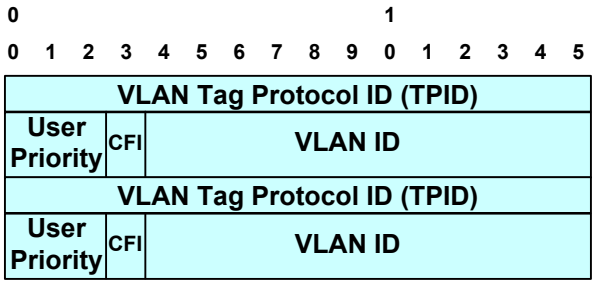



*Figure 10-12. Stacked VLAN Tag Format*




The VLAN tag's Protocol ID (TPID) can be either the typical value of 0x8100 or a value configured in the vlan_2nd_tag_identifier field in Packet_classifier_cfg_reg7.

- The User Priority field is used to assign a priority level to the Ethernet packet.
- The CFI (Canonical Format Indicator) fields indicate the presence of a Router Information Field.
- The VLAN ID, uniquely identifies the VLAN to which the Ethernet packet belongs.

#### 10.6.1.2 UDP/IPv4 Header

*Figure 10-13. UDP/IPv4 Header Format*

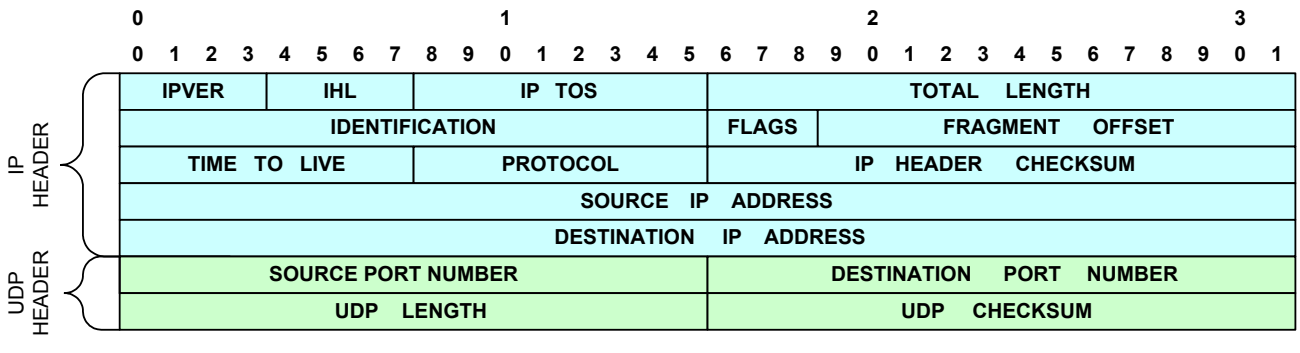


**Table 10-7. IPv4 Header Fields (UDP)**

| Field | Description |
|---|---|
| IPVER | IP version number. IPv4 IPVER=4 |
| IHL | Length in 32-bit words of the IP header, IHL=5 |
| IP TOS | IP type of service |
| Total Length | Length in octets of IP header and data |
| Identification | IP fragmentation identification |
| Flags | IP control flags; must be set to 010 to avoid fragmentation |
| Fragment Offset | Indicates where in the datagram the fragment belongs; not used for TDM-over-Packet |
| Time To Live | IP time-to-live field; datagrams with zero in this field are to be discarded |
| Protocol | Must be set to 0x11 to signify UDP |
| IP Header Checksum | Checksum for the IP header |
| Source IP Address | IP address of the source |
| Destination IP Address | IP address of the destination |

**Table 10-8. UDP Header Fields**

| Field | Description |
|---|---|
| Source Port Number, Destination Port Number | Either the source or the destination port number holds the bundle identifier. The unused field can be set to 0x85E (2142), which is the user port number assigned to TDM-over-Packet by the Internet Assigned Numbers Authority (IANA).<br>For UDP/IP-specific OAM packets, the bundle identifier is all ones. |
| UDP length | Length in octets of UDP header and data |
| UDP checksum | Checksum of UDP/IP header and data. If not computed it must be set to zero. |

#### *10.6.1.3 UDP/IPv6 Header*

*Figure 10-14. UDP/IPv6 Header Format*

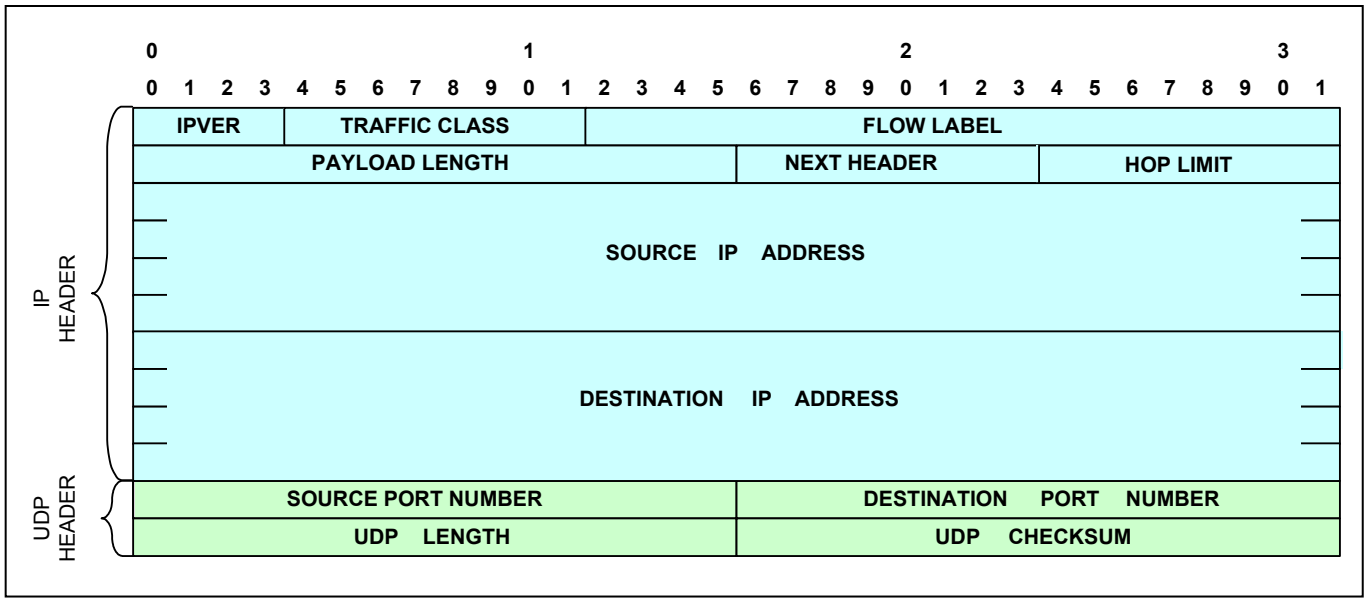


**Table 10-9. IPv6 Header Fields (UDP)**

| Field | Description |
|---|---|
| IPVER | IP version number, for IPv6 IPVER = 6 |
| Traffic Class | An 8-bit field similar to the type of service (ToS) field in IPv4. |
| Flow Label | The 20-bit Flow Label field can be used to tag packets of a specific flow to differentiate the packets at the network layer. |
| Payload Length | Similar to the Total Length field in IPv4. This field indicates the total length of the IP header and data in octets. |
| Next Header | Similar to the Protocol field in IPv4. It determines the type of information following the basic IPv6 header. Must be set to 0x11 to signify UDP. |
| Hop Limit | Similar to the Time to Live field in IPv4. |
| Source IP Address | Similar to the Source Address field in IPv4, except that the field contains a 128-bit source address for IPv6 instead of a 32-bit source address for IPv4. |
| Destination Address | Similar to the Destination Address field in IPv4, except that the field contains a 128-bit destination address for IPv6 instead of a 32-bit destination address for IPv4. |

#### *10.6.1.4 MPLS Header*

*Figure 10-15. MPLS Header Format*

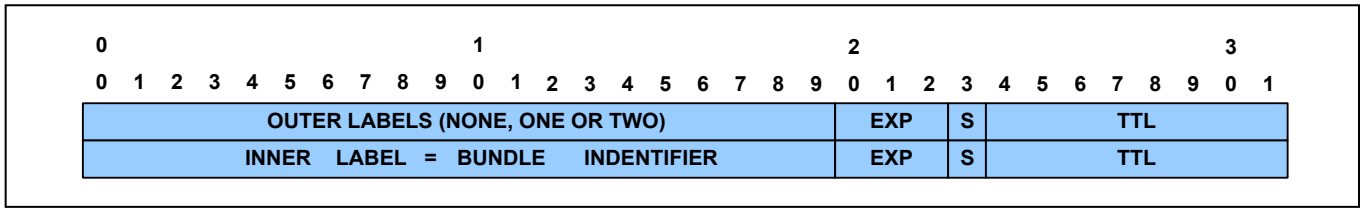



**Table 10-10. MPLS Header Fields**

| Field | Description |
|---|---|
| Outer Labels | MPLS labels, which identify the MPLS LSP, used to tunnel the TDMoMPLS packets through the MPLS network. Also known as tunnel label(s) or transport label(s). The label number can be assigned either manually or using the MPLS control protocol. There can be zero, one or two outer labels. |
| EXP | Experimental field |
| S | Stacking bit: S=1 indicates stack bottom (i.e. the inner label). S=0 for all outer labels. |
| TTL | MPLS time to live |
| Inner Label | MPLS inner label (also known as the PW label or the interworking label) contains the bundle identifier used to multiplex multiple bundles within the same tunnel. It is always be at the bottom of the MPLS label stack, and hence its stacking bit is set (S=1). |

#### *10.6.1.5 MEF Header*

*Figure 10-16. MEF Header Format*

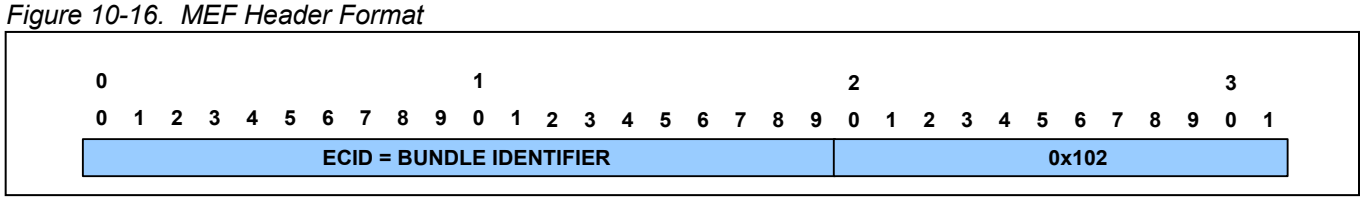



**Table 10-11. MEF Header Fields**

| Field | Description |
|---|---|
| ECID | The Emulated Circuit Identifier (ECID) field. Contains the bundle identifier. |

#### 10.6.1.6 L2TPv3/IPv4 Header

*Figure 10-17. L2TPv3/IPv4 Header Format*

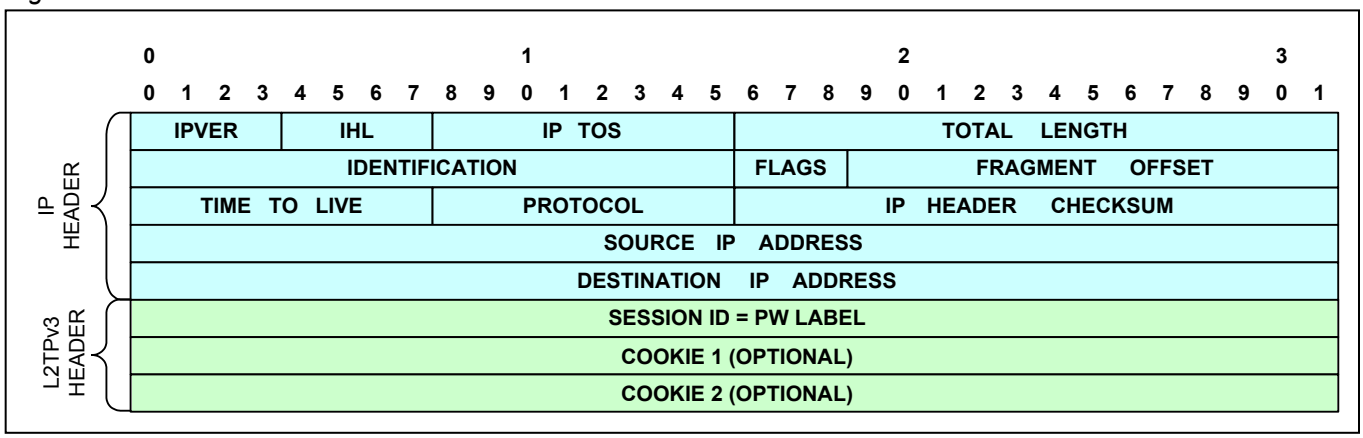



**Table 10-12. IPv4 Header Fields (L2TPv3)**

| Field | Description |
|---|---|
| IPVER | See Table 10-7. |
| IHL | |
| IP TOS | |
| Total Length | |
| Identification | |
| Flags | |
| Fragment Offset | |
| Time To Live | |
| Protocol | Must be set to 0x73 to signify L2TPv3 |
| IP Header Checksum | See Table 10-7. |
| Source IP Address | |
| Destination IP Address | |

**Table 10-13. L2TPv3 Header Fields**

| Field | Description |
|---|---|
| Session ID (32 bits) | Locally significant L2TP session identifier, Contains the bundle identifier. All bundle identifiers are available for use except 0, which is reserved. |
| Cookie (32 or 64 bits) | Optional field that contains a randomly selected value used to validate association of the packet with the expected bundle identifier |

#### 10.6.1.7 L2TPv3/IPv6 Header

*Figure 10-18. L2TPv3/IPv6 Header Format*

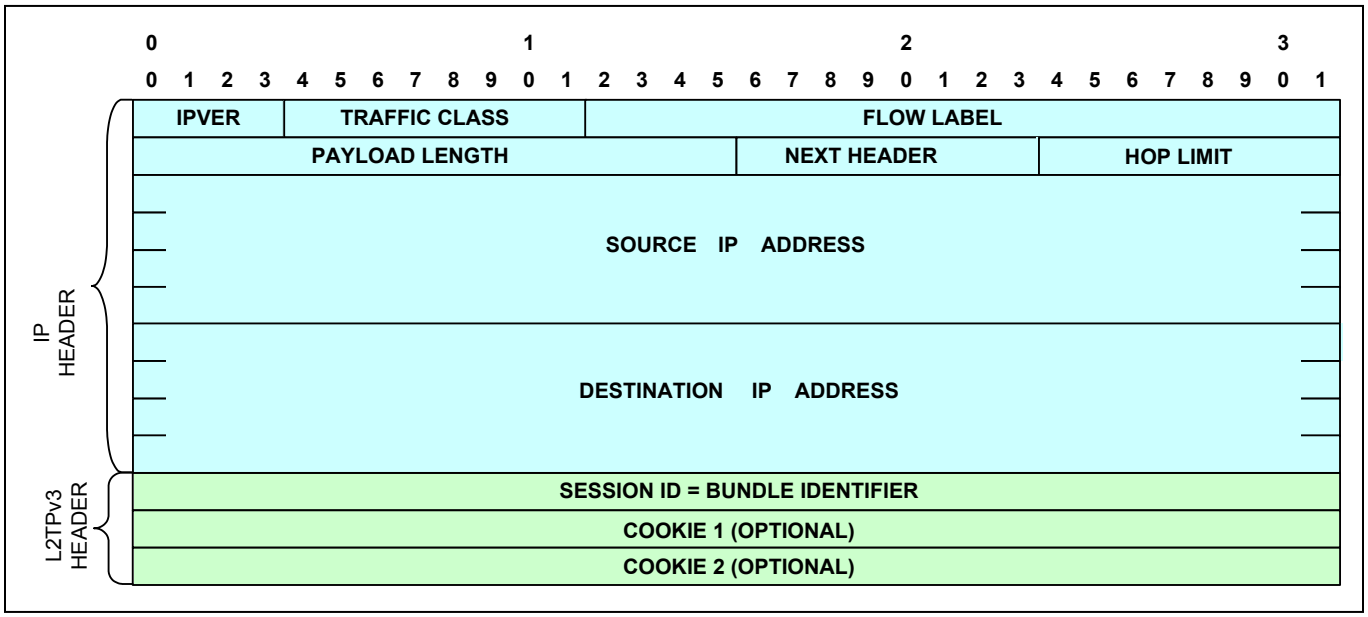



**Table 10-14. IPv6 Header Fields (L2TPv3)**

| Field | Description |
|---|---|
| IPVER | See Table 10-9. |
| Traffic Class | |
| Flow Label | |
| Payload Length | |
| Next Header | Must be set to 0x73 to signify LTPv3 |
| Hop Limit | See Table 10-9. |
| Source Address | |
| Destination Address | |

#### 10.6.1.8 Control Word

*Figure 10-19. Control Word Format*

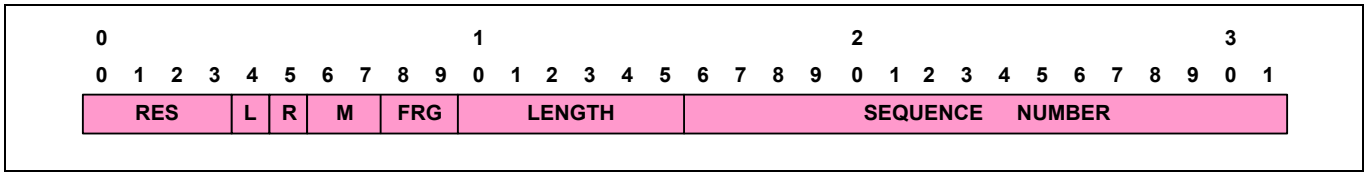



**Table 10-15. Control Word Fields**

| Field | Description |
|---|---|
| RES | Reserved bits.<br>Must be set to zero. |
| L | Local loss of sync failure.<br>This bit is set by CPU software (Port[n]_cfg_reg.Loss) for packets transmitted out the Ethernet port. A set L bit indicates that the source has detected or has been informed of a TDM physical layer fault impacting the data to be transmitted. This bit can be used to indicate physical layer LOS that should trigger AIS generation at the far end. Once set, if the TDM fault is rectified, the L bit must be cleared. |
| R | Remote receive failure.<br>This bit is set by CPU software (Tx_R_bit field in bundle configuration) for packets transmitted out the Ethernet port.. A set R bit indicates that the source is not receiving packets at the Ethernet port, i.e., |

| Field | Description |
|---|---|
| | there is a failure of that direction of the bi-directional connection. This indication can be used to signal congestion or other network related faults. Receiving remote failure indication may trigger fall-back mechanisms for congestion avoidance. The R bit must be set after a preconfigured number of consecutive packets are not received, and must be cleared once packets are once again received. |
| M | Defect Modifier failure.<br>These bits are set by CPU software (Port[n]_cfg_reg.Tx_defect_modifier) for packets transmitted out the Ethernet port.. This field is optional. When used it supplements the L-bit meaning. |
| FRG | Fragmentation field<br>This field is used for fragmenting multiframe structures into multiple packets in case of CESoPSN structured with CAS bundles.<br>The field is used as follows:<br>00 = Indicates that the entire (unfragmented) multiframe structure is carried in a single packet.<br>01 = Indicates the packet carrying the first fragment.<br>10 = Indicates the packet carrying the last fragment.<br>11 = Indicates a packet carrying an intermediate fragment. |
| Length | Length field<br>Includes control word, payload and RTP header (if present) unless it is a UDP/IP packet. It is only used when the total length of these fields is less than 64 bytes. Otherwise, it must be set to zero. |
| Sequence Number | TDM-over-Packet sequence number, defined separately for each bundle and incremented by one for each TDMoP packet sent for that bundle. The initial value of the sequence number is random (unpredictable) for security purposes, and the value is incremented in wrap-around manner separately for each bundle. Used by the receiver to detect packet loss and restore packet sequence.<br>The HDLC payload type machine supports three different modes for this field: always zero, incremented in wrap-around manner or incremented in wrap-around manner, but skips zero value.<br>For OAM packets, it uniquely identifies the message. Its value is unrelated to the sequence number of the TDMoP data packets for the bundle in question. It is incremented in query messages, and replicated without change in replies. |

#### 10.6.1.9 RTP Header

*Figure 10-20. RTP Header Format*

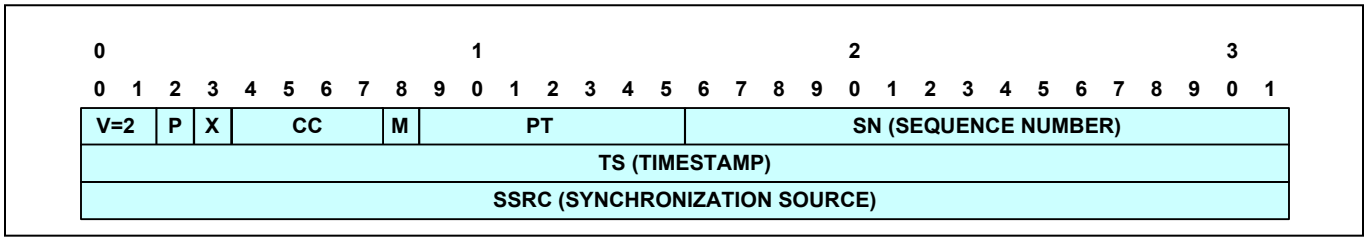



**Table 10-16. RTP Header Fields**

| Field | Description |
|---|---|
| V | RTP version. Must be set to 2. |
| P | Padding bit. Must be set to 0. |
| X | Extension bit. Must be set to 0. |
| CC | CSRC Count. Must be set to 0. |
| M | Marker bit. Must be set to 0. |
| PT | Payload Type. One PT value MUST be allocated from the range of dynamic values for each direction of the bundle. The same PT value MAY be reused for both directions of the bundle, and also reused between different bundles. |
| SN | Sequence number. Identical to the sequence number in the control word. |
| TS | Timestamp. The RTP header can be used in conjunction with the following modes of timestamp generation:<br>Absolute mode: the chip sets timestamps using the clock from the incoming TDM circuit. As a consequence, the timestamps are closely correlated with the sequence numbers. The timestamp is incremented by one every 125 μs.<br>Differential (common clock) mode: The two chips at bundle edges have access to the same high-quality network clock, and this clock source is used for timestamp generation. |

| Field | Description |
|---|---|
| SSRC | Identifies the synchronization source. This identifier should be chosen randomly, with the intent that no two synchronization sources within the same RTP session have the same SSRC identifier. |

#### 10.6.1.10 TDM-over-Packet Payload

This field can contain the following payload types:

- AAL1
- HDLC
- RAW (SAToP or CESoPSN formats)
- OAM (VCCV or UDP/IP-specific).

The AAL1, HDLC and RAW payload type details are provided in sections 10.6.6, 10.6.7 and 10.6.8, respectively. The formats of the OAM payload types are described below.

##### 10.6.1.10.1 VCCV OAM

When using inband performance monitoring, additional OAM packets are sent using the same bundle identifier as the TDM data packets. The OAM packets are identified by having their first nibble (after the PSN specific layers) equal to 0001 and must be separated from TDM data packets before further processing of the control word. The PSN-specific layers are identical to those used to carry the TDM data.

*Figure 10-21. VCCV OAM Packet Format*

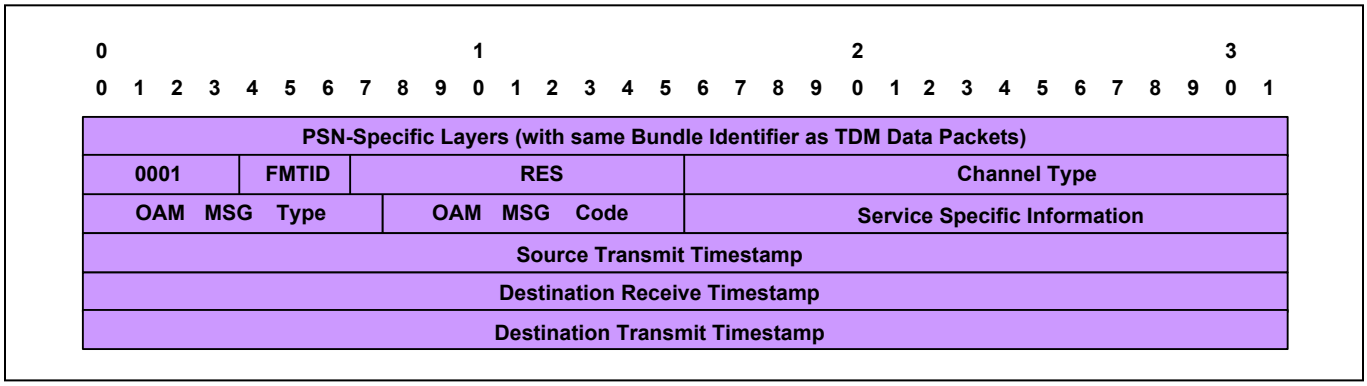



**Table 10-17. VCCV OAM Payload Fields**

| Field | Description |
|---|---|
| FMTID | Must be set to zero |
| RES | Reserved and must be set to zero |
| Channel Type | Must be set to the value allocated by IANA for TDM-over-Packet VCCV OAM |
| OAM Msg Type | See Table 10-18. |
| OAM Msg Code | |
| Source Transmit Timestamp | |
| Destination Receive Timestamp | |
| Destination Transmit Timestamp | |

##### 10.6.1.10.2 UDP/IP-Specific OAM

When using a UDP/IP-Specific OAM, all OAM packet MUST use one of the bundle identifiers preconfigured to indicate OAM (using OAM ID Table). The PSN-specific layers are identical for OAM packets (except for the bundle identifier) to those used to carry the TDM data.

*Figure 10-22. UDP/IP-Specific OAM Packet Format*

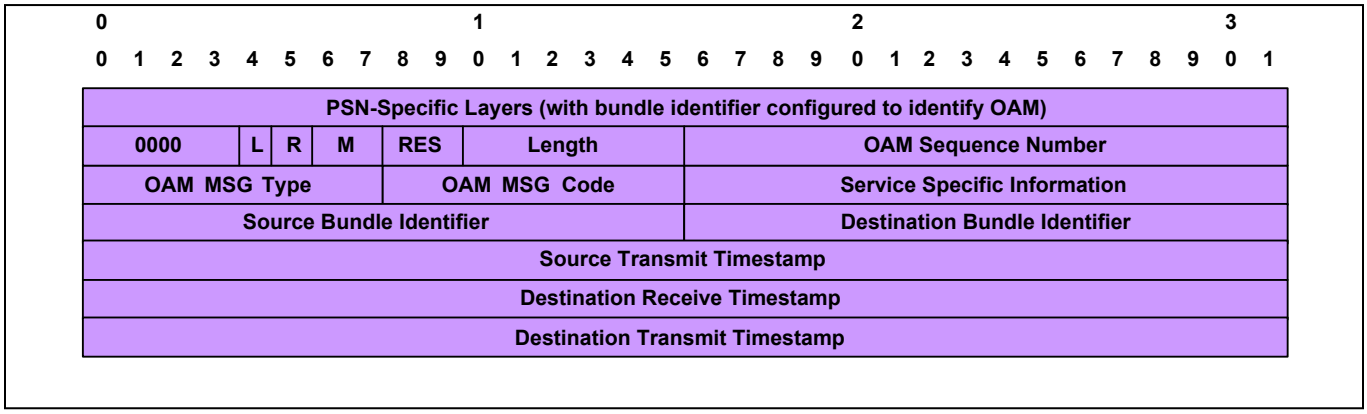



**Table 10-18. UDP/IP-Specific OAM Payload Fields**

| Field | Description |
|---|---|
| L, R, M | Identical to those of the bundle being tested |
| Length | OAM message packet length (in bytes) |
| OAM Sequence Number | Uniquely identifies the message. Its value is unrelated to the sequence number of the TDM data packets for the bundle in question. It is incremented in query messages, and replicated without change in replies. |
| OAM Msg Type | Indicates the function of the message. At present, the following are defined:<br>0: one way connectivity query message<br>8: one way connectivity reply message |
| OAM Msg Code | Information related to the message; its interpretation depends on the message type.<br><br>For OAM Msg Type=0 (connectivity query) messages, the following codes are defined:<br>0: Validate connection<br>1: Do not validate connection<br><br>For OAM Msg Type=8 (connectivity reply) messages, the available codes are:<br>0: Acknowledge valid query<br>1: Invalid query (configuration mismatch) |
| Service Specific Information | Can be used to exchange configuration information between gateways. If not used, it must contain zero. Its interpretation depends on the payload type. At present, the following is defined for AAL1 payloads:<br>Bits 16–23: Number of timeslots being transported, e.g. 24 for full T1<br>Bits 24–31: Number of 48-byte AAL1 PDUs per packet, e.g. 8 when packing 8 AAL1 AAL1 SAR PDUs per packet |
| Source Bundle Identifier | The bundle identifier used for TDM-over-Packet traffic from the source to the destination. |
| Destination Bundle Identifier | The bundle identifier used for TDM-over-Packet traffic from the destination to source. |
| Source Transmit Timestamp | The time the PSN-bound gateway transmitted the query message. This field and the following fields only appear if delay is being measured. The resolution is configurable to 100 μs or 1 μs. |
| Destination Receive Timestamp | The time the destination gateway received the query message. |
| Destination Transmit Timestamp | The time the destination gateway transmitted the reply message. |

For more details about OAM Signaling, see Section 10.6.17.

### 10.6.2 Typical Application

In the application below (Figure 10-23), the device is embedded in a TDMoIP gateway to achieve TDM connectivity over a PSN. The TDM-over-Packet packet formats for both IP and MPLS are shown in Figure 10-24 and Figure 10-25, respectively.

*Figure 10-23. TDM Connectivity over a PSN*

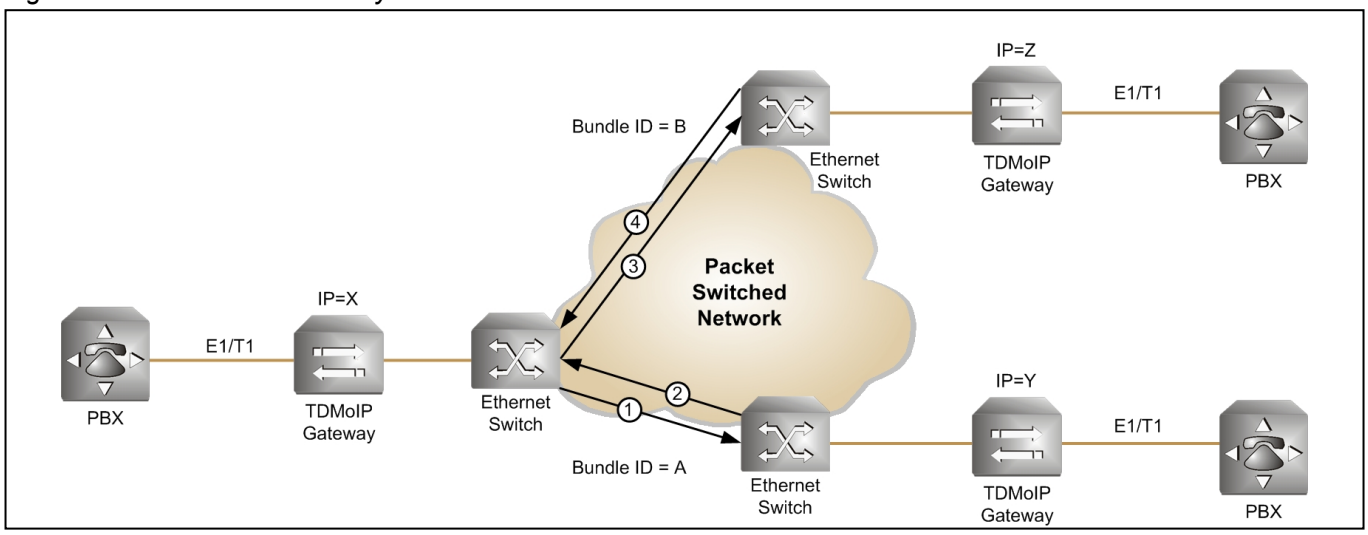
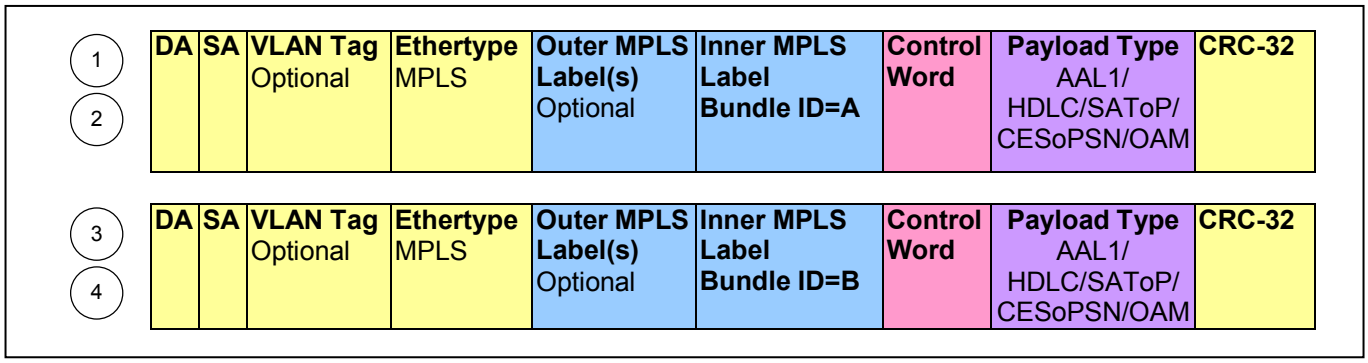



*Figure 10-24. TDMoP Packet Format in a Typical Application*

| | DA | SA | VLAN Tag | Ethertype | IP Header | UDP or L2TPv3 Header | Control Word | Payload Type | CRC-32 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | Optional | IP | Src. IP=X<br>Dst. IP=Y | Bundle ID=A | | AAL1/ HDLC/SAToP/ CESoPSN/ OAM | |
| 2 | | | Optional | IP | Src. IP=Y<br>Dst. IP=X | Bundle ID=A | | AAL1/ HDLC/SAToP/ CESoPSN/OAM | |
| 3 | | | Optional | IP | Src. IP=X<br>Dst. IP=Z | Bundle ID=B | | AAL1/ HDLC/SAToP/ CESoPSN/OAM | |
| 4 | | | Optional | IP | Src. IP=Z<br>Dst. IP=X | Bundle ID=B | | AAL1/ HDLC/SAToP/ CESoPSN/OAM | |

*Figure 10-25. TDMoMPLS Packet Format in a Typical Application*




### 10.6.3 Clock Recovery

The TDM-over-Packet block's innovative clock recovery process is divided into two successive phases. In the acquisition phase, rapid frequency lock is attained. In the tracking phase, frequency lock is sustained and phase is also tracked. During the tracking phase, jitter is attenuated to comply with the relevant telecom standards even for packet-switched networks with relatively large packet delay variation. Packet loss immunity is also significantly improved.

During the acquisition phase, a direct estimation of the frequency discrepancy between the far-end and near-end service clocks continuously drives an internal frequency synthesis device through a band-limited control loop. As a result, frequency acquisition is achieved rapidly (typically less than 10 seconds). The clock recovery capture range is ±90 ppm around the nominal service clock for any supported clock rate.

Once the frequency-monitoring unit has detected a steady frequency lock, the system switches to its tracking phase. During the tracking phase the fill level of the received-packet jitter buffer drives the internal frequency synthesizer through a similar band-limited control loop.

While in the tracking phase, two tasks are performed. First, the far-end service clock frequency is slowly and accurately tracked, while compelling the regenerated near-end service clock to have jitter and wander levels that conform to ITU-T G.823/G.824 requirements, even for networks that introduce high packet delay variation and packet loss. This performance can be attained due to a very efficient jitter attenuation mechanism, combined with a high resolution internal digital PLL ($\Delta f$=0.4 ppb). Second, the received-packet jitter buffer is maintained at its fill level, regardless of the initial frequency discrepancy between the clocks. As a result, the latency added by the mechanism is minimized, while immunity against overflow/underflow events (caused by extreme packet delay variation events) is substantially enhanced.

The TDM-over-Packet block supports two clock recovery modes: common clock (differential) mode and adaptive mode.

The common clock mode is used for applications where the TDMoP gateways at both ends of the PSN path have access to the same high-quality reference clock. This mode makes use of RTP differential mode time-stamps and therefore the RTP header must be present in TDMoP packets when this mode is used. The common reference clock is provided to the chip on the CLK_CMN input pin. The device is configured for common clock mode when Clock_recovery_en=1 in General_cfg_reg0 and RTP_timestamp_generation_mode=1 in General_cfg_reg1.

The adaptive clock mode is based solely on packet inter-arrival time and therefore can be used for applications where a common reference clock is *not* available to both TDMoP gateways. This mode does not make use of time-stamps and therefore the RTP header is not needed in the TDMoP packets when this mode is used. The device is configured for adaptive clock mode when Clock_recovery_en=1 in General_cfg_reg0 and RTP_timestamp_generation_mode=0 in General_cfg_reg1.

In adaptive mode, for low-speed interfaces (up to 4.6 MHz), an on-chip digital PLL, clocked by a 38.88MHz clock derived from the CLK_HIGH pin, synthesizes the recovered clock frequency. The frequency stability characteristics

of the CLK_HIGH signal depend on the wander requirements of the recovered TDM clock. For applications where the recovered TDM clock must comply with G.823/G.824 requirements for traffic interfaces, typically a TCXO can be use as the source for the CLK_HIGH signal. For applications where the recovered clock must comply with G.823/G.824 requirements for synchronization interfaces, the CLK_HIGH signal typically must come from an OCXO.

In addition to performing clock recovery for up to eight low-speed (typically E1/T1) signals, the device can also be configured in a high-speed mode in which it supports one E3, T3 or STS-1 signal in and out of port 1. In high-speed mode, the on-chip digital PLL synthesizes the recovered clock frequency divided by 10 (for STS-1) or 12 (for E3 or T3). This clock is available on the TDM1_ACLK output pin and can be multiplied by an external PLL to get the recovered clock of the high-speed signal (see section 15.3). High-speed mode is enabled when High_speed=1 in General_cfg_reg0.

For applications where the chip is used only for clock recovery purposes (i.e. data is not forwarded through the chip) the external SDRAM is not needed.

### 10.6.4 Timeslot Assigner (TSA)

The TDM-over-Packet block contains one Timeslot Assigner for each TDM port (framed or multiframed). The TSA is bypassed in high-speed mode (i.e. when High_speed=1 in General_cfg_reg0.) The TSA tables are described in section 11.4.5.

The TSA assigns 2-, 7- or 8-bit wide timeslots to a specific bundle and a specific receive queue. 2-bit timeslots are used for delivering 16K HDLC channels. The 2 bits are located at the first 2 bits (PCM MSbits, HDLC LSbits) of the timeslot. The next 6 bits of the timeslot cannot be assigned. 7-bit timeslots are used for delivering 56kbps HDLC channels. The 7 bits are located at the first 7 bits (PCM MSbits, HDLC LSbits) of the timeslot. The last bit of the timeslot cannot be assigned. The 2-bit and 7-bit timeslots may be assigned only to the HDLC payload type machine. The AAL1 and RAW payload type machines support only 8-bit timeslots. For unframed/Nx64 interfaces all entries must be configured as 8-bit timeslots.

Each port has two TSA tables (banks): one active and the other one shadow. The TSA_int_act_blk status bit in Port[n]_stat_reg1 indicates which bank is currently active. The CPU can only write to the shadow table. After TSA entries are changed in the shadow table the TSA tables should be swapped by changing the TSA_act_blk bit in Port[n]_cfg_reg so that the active table becomes the shadow table and the shadow table becomes the active table. Changes take effect at the next frame sync signal. For an unframed interface the changes take effect up to 256 TDM clock cycles after the TSA_act_blk is changed. After the change occurs, the TSA_int_act_blk bit is updated by the device.

Each table consists of 32 entries, one entry per timeslot. The first entry refers to the first timeslot, i.e. the first 8 bits of the frame (where the frame sync signal indicates start-of-frame). The second entry refers to the second timeslot, i.e. the 8 bits after the first 8 bits, and so on.

The format of a table entry is shown in section 11.4.5. If a port is configured for an unframed signal format, all 32 entries for that port must have the same settings for all fields.

A bundle can only be composed of timeslots from a single TDM port, but timeslots from a TDM port can be assigned to multiple bundles.

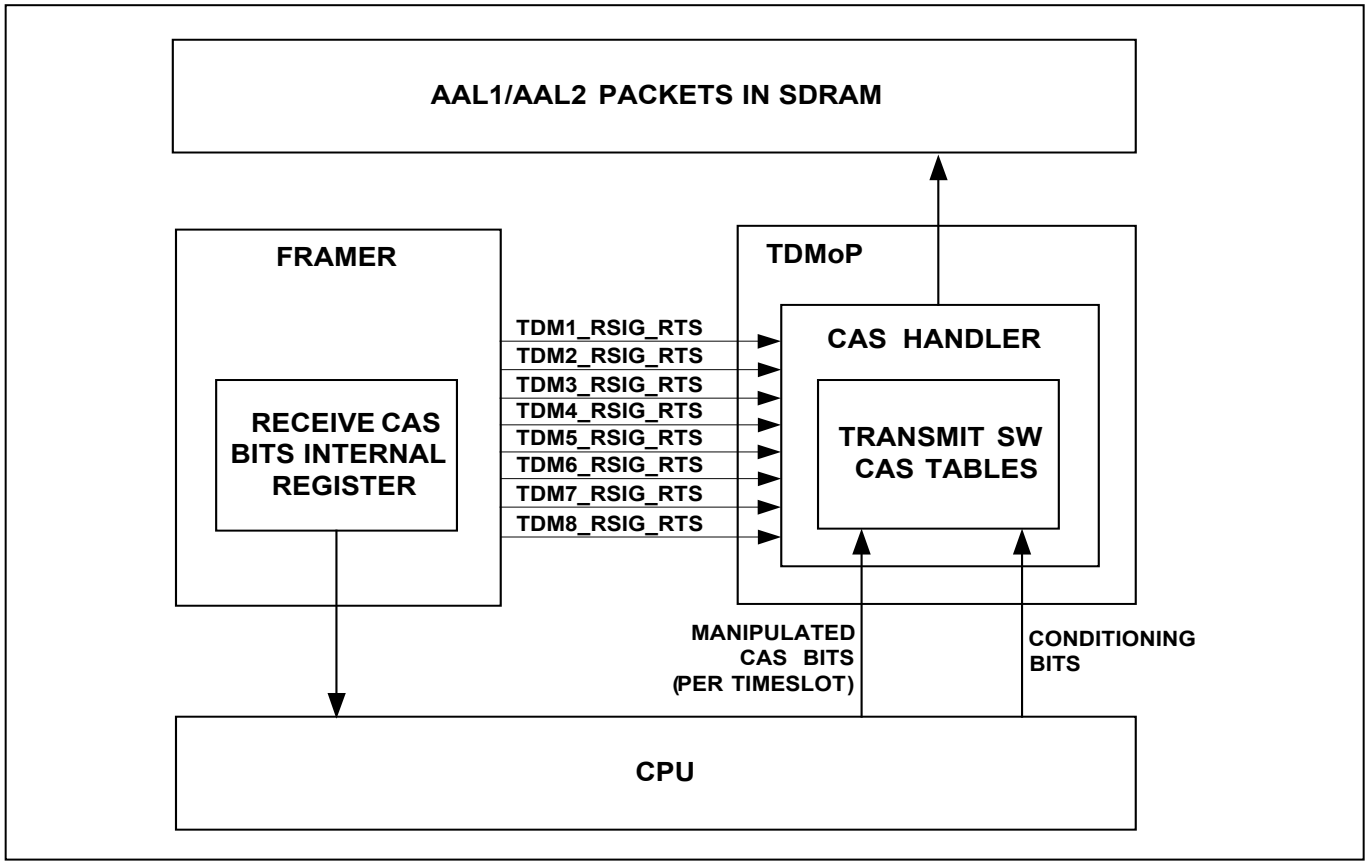
### 10.6.5 CAS Handler

#### 10.6.5.1 CAS Handler, TDM-to-Ethernet Direction

In the TDM-to-Ethernet direction, the CAS handler receives the CAS bits (for structured-with-CAS AAL1/CESoPSN bundles) on the TDMn_RSIG_RTS signal. Depending on the value of the per-bundle Tx_CAS_source configuration bit in the Bundle Configuration Tables, the CAS handler inserts either the CAS bits from the corresponding TDMn_RSIG_RTS signal or the values from the transmit SW CAS tables (section 11.4.9) into the AAL1/CESoPSN packets, in order to deliver the signaling as part of the AAL1/CESoPSN payload packets. See Figure 10-26.

The transmit SW CAS tables may contain conditioning bits set by CPU software during configuration (per timeslot). If CAS bits received on the TDMn_RSIG_RTS signal change, a per-timeslot maskable interrupt is asserted. The Tx_CAS_change registers in the Interrupt Controller indicate which timeslots have changed CAS bits. The Tx_CAS_change_mask registers are available to selectively mask these interrupts. Upon notification that CAS bits have changed, the CPU can read the CAS bits directly from the receive signaling registers of the neighboring E1/T1 framer component, alter them if needed, and write them into the TDMoP block's transmit SW CAS tables.

*Figure 10-26. CAS Transmitted in the TDM-to-Ethernet Direction*




There is a transmit SW CAS table for each TDM port. Each table consists of 4 rows, and each row contains the CAS bits of eight timeslots. For ports configured for E1, timeslots 1–15 and 17–31 are used and timeslots 0 and 16 are meaningless. For ports configured for T1, timeslots 0–23 are used and timeslots 24–31 are meaningless. Ports configured for T1 SF have two copies of A and B CAS bits arranged A, B, A, B. Other port types have one copy of bits A, B, C and D. These cases are illustrated in Figure 10-27 and Figure 10-28.

*Figure 10-27. Transmit SW CAS Table Format for E1 and T1-ESF Interfaces*

| 31 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| ABCD (TS7) | ABCD (TS6) | ABCD (TS5) | ABCD (TS4) | ABCD (TS3) | ABCD (TS2) | ABCD (TS1) | ABCD (TS0) |
| ABCD (TS15) | .. | .. | .. | .. | .. | .. | ABCD (TS8) |
| ABCD (TS23) | .. | .. | .. | .. | .. | .. | ABCD (TS16) |
| ABCD (TS31) | .. | .. | .. | .. | .. | .. | ABCD (TS24) |

*Figure 10-28. Transmit SW CAS Table Format for T1-SF Interfaces*

| 31 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| ABAB (TS7) | ABAB (TS6) | ABAB (TS5) | ABAB (TS4) | ABAB (TS3) | ABAB (TS2) | ABAB (TS1) | ABAB (TS0) |
| ABAB (TS15) | .. | .. | .. | .. | .. | .. | ABAB (TS8) |
| ABAB (TS23) | .. | .. | .. | .. | .. | .. | ABAB (TS16) |

**Table 10-19. CAS – Supported Interface Connections for AAL1 and CESoPSN**

| TDM-to-Packet Interface Format | Packet-to-TDM Interface Format | Transmitted Bits |
|---|---|---|
| E1 MF | E1 MF | CAS bits are transferred as-is. |
| T1 SF | T1 SF | |
| T1 ESF | T1 ESF | |
| T1 ESF | T1 SF | Only A and B bits transferred. |
| T1 SF | T1 ESF | A and B bits transferred. C and D bits sourced from the SF_to_ESF_low_CAS_bits field in Port[n]_cfg_reg. |

For structured-with-CAS bundles connecting two T1 SF/ESF interfaces, the per-bundle Tx_dest_framing bit in the Bundle Configuration Tables indicates the destination interface framing type (SF or ESF).

The figures below shows the location of the CAS bits in the TDMn_RSIG_RTS data stream for each framing mode.

*Figure 10-29. E1 MF Interface RSIG Timing Diagram (two_clocks=1)*

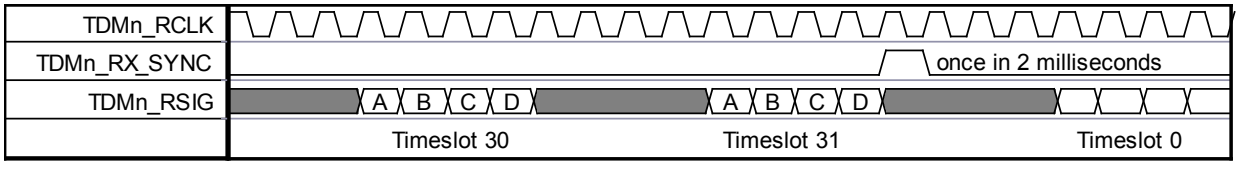


*Figure 10-30. T1 ESF Interface RSIG Timing Diagram (two_clocks=0)*

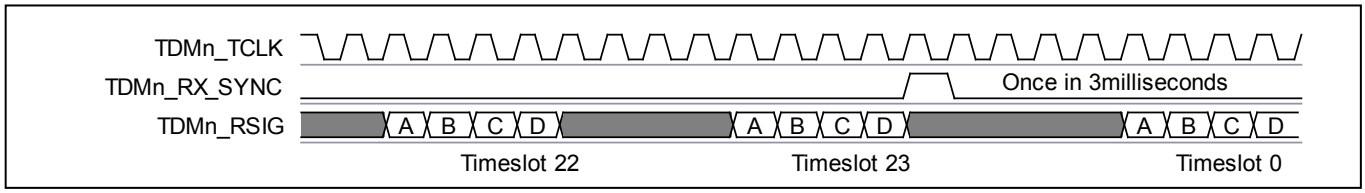
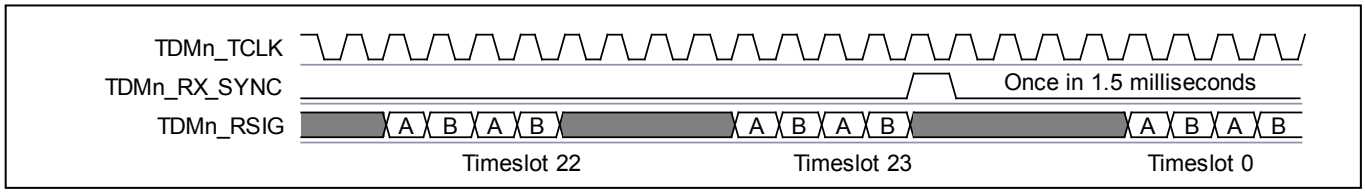



*Figure 10-31. T1 SF Interface RSIG (two_clocks=0) – Timing Diagram*




TDMn_RX_SYNC can be left unconnected or connected to ground if the neighboring E1/T1 framer IC cannot drive it. The TDMoP block has an internal free running counter that generates this signal internally when not driven by an external source. This internally generated multiframe sync signal is synchronized to the TDMn_RX_SYNC input pulse when present.

#### *10.6.5.2 CAS Handler, Ethernet-to-TDM Direction*

In the Ethernet-to-TDM direction, the CAS is received from the incoming packets.

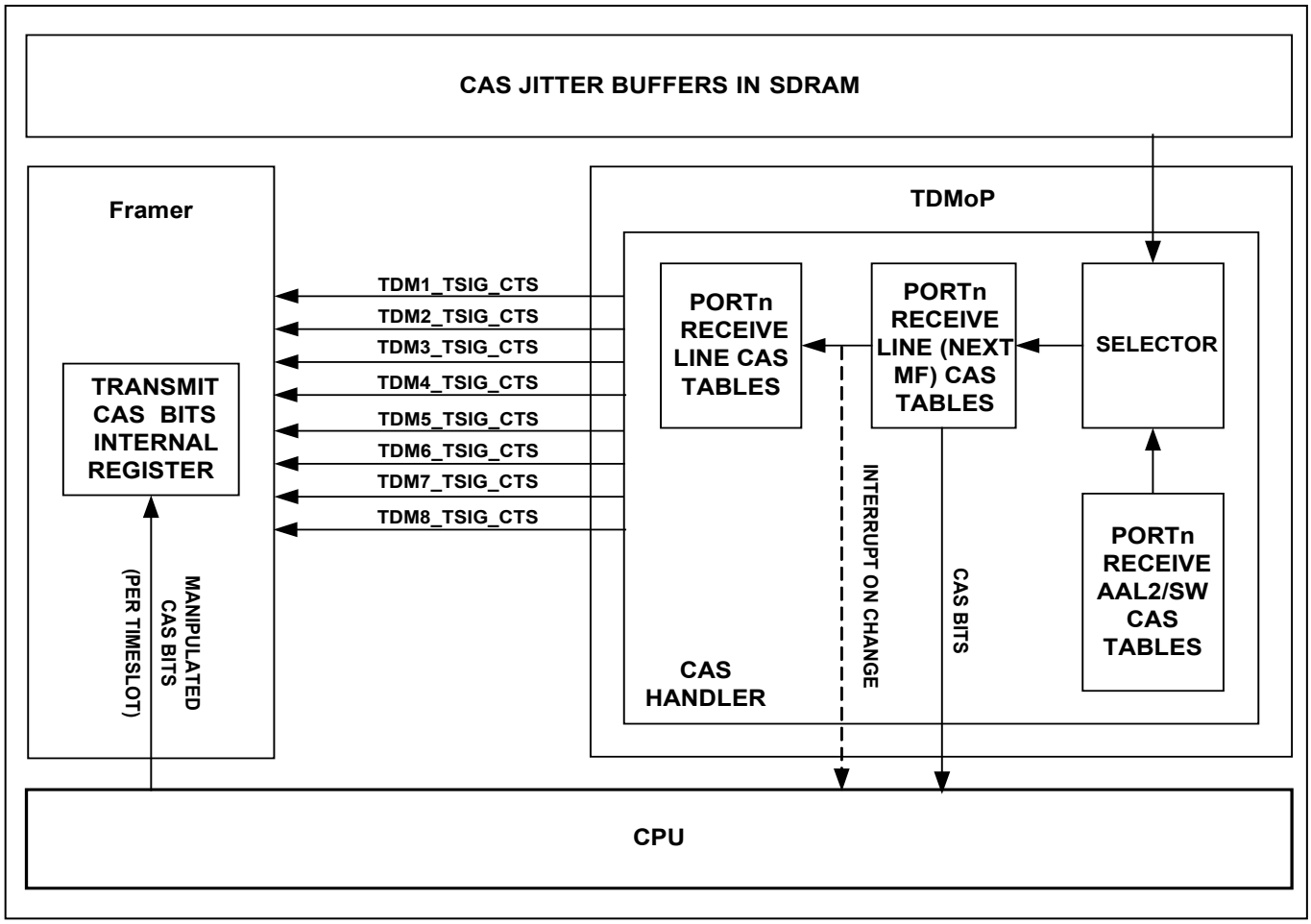
The AAL1/RAW payload type machine extracts the CAS bits from the TDM-over-packet payload and writes them into the CAS jitter buffers in the SDRAM (for structured-with-CAS AAL1/CESoPSN bundles only). The CAS jitter buffers store the CAS information of up to 128 timeslots of the eight ports.

Selectors in the CAS handler send the CAS bits either from the CAS jitter buffers or from the Receive SW CAS tables to the line (next MF) CAS tables (see Figure 10-32). The selectors' decision logic is shown in Table 10-20.

**Table 10-20. CAS Handler Selector Decision Logic**

| Condition | Source of CAS bits Driven on TDMn_TSIG_CTS for this Timeslot |
|---|---|
| Timeslot not assigned or assigned to a bundle which is not an AAL1/CESoPSN structured bundle (Rx_assigned=0 or Structured_type=0 for its TSA entry) | Receive SW CAS tables |
| AAL1 bundle jitter buffer is in underrun state and Rx_CAS_src=1 | |
| Timeslot assigned to an AAL1/CESoPSN structured bundle (Rx_assigned=1 and Structured_type=1 for its TSA entry) | Corresponding CAS jitter buffer in SDRAM (CAS value is the latest received) |
| AAL1/CESoPSN bundle jitter buffer is in underrun state and Rx_CAS_src=0 | |

*Figure 10-32. CAS Transmitted in the Ethernet-to-TDM Direction*




The Receive SW CAS tables contains CAS bits written by CPU software.

Each port's Receive Line CAS table (section 11.4.10) is updated with the CAS bits stored in the Receive Line (Next MF) CAS table when the TDMn_TX_MF_CD signal is asserted to indicate the multiframe boundary. For E1 ports, CAS bits are updated every 2 milliseconds. For T1 SF ports, CAS bits are updated every 1.5 milliseconds. For T1 ESF ports, CAS bits are updated every 3 milliseconds.

There is a Receive Line CAS table for each TDM port. These tables hold the CAS information extracted from received packets and subsequently transmitted on TDMn_TSIG signals. Each table contains 32 rows, and each row holds the CAS bits of one timeslot. Only the first 24 rows are used for T1 interfaces. For E1 and T1 ESF interfaces, each row holds the A, B, C and D bits. For T1 SF interface where only the A and B bits exist, each row holds the A and B bits duplicated i.e. A, B, A, B.

If CAS bits change in the Receive Line CAS table, a per-timeslot interrupt is asserted. The Rx_CAS_change registers in the Interrupt Controller indicate which timeslots have changed CAS bits. Upon notification that CAS bits have changed, CPU software can read the CAS bits from the Receive Line (Next MF) CAS table, manipulate them and then write them directly into the transmit signaling registers of the neighboring E1/T1 framer IC. In this case, the framer should be configured to use the CAS information from its CAS registers and not from its signaling input pin.

The bits in each Receive Line CAS table are transmitted on the TDMn_TSIG signal, as shown in the figures below.

*Figure 10-33. E1 MF Interface TSIG Timing Diagram*

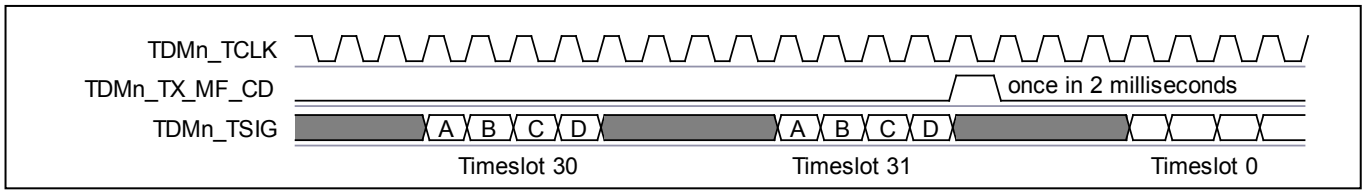



*Figure 10-34. T1 ESF Interface TSIG Timing Diagram*

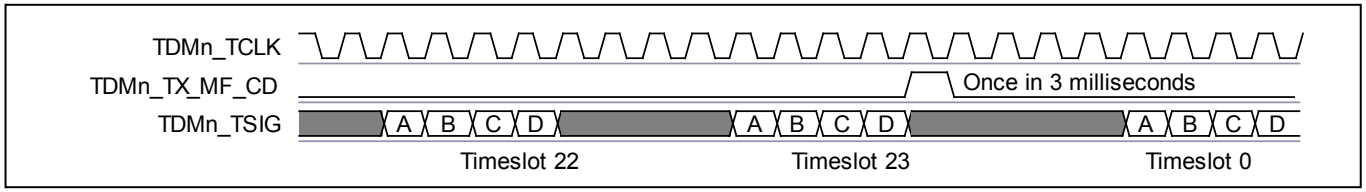
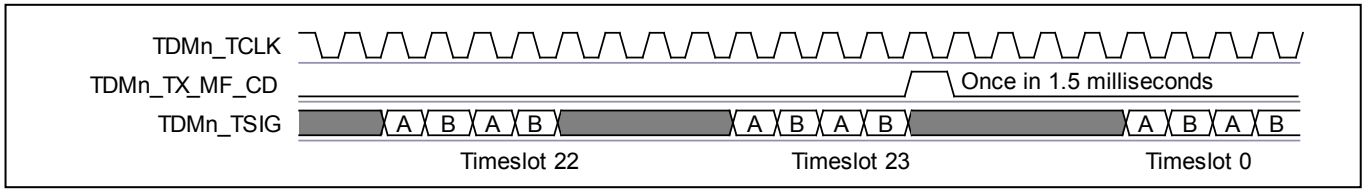



*Figure 10-35. T1 SF Interface TSIG Timing Diagram*




TDMn_TX_MF_CD can be left unconnected or connected to ground if the framer cannot drive it. The TDMoP block has an internal free running counter that generates this signal internally when not driven by external source. This internally generated multiframe sync signal is synchronized to the TDMn_TX_SYNC input pulse when present.

### 10.6.6 AAL1 Payload Type Machine

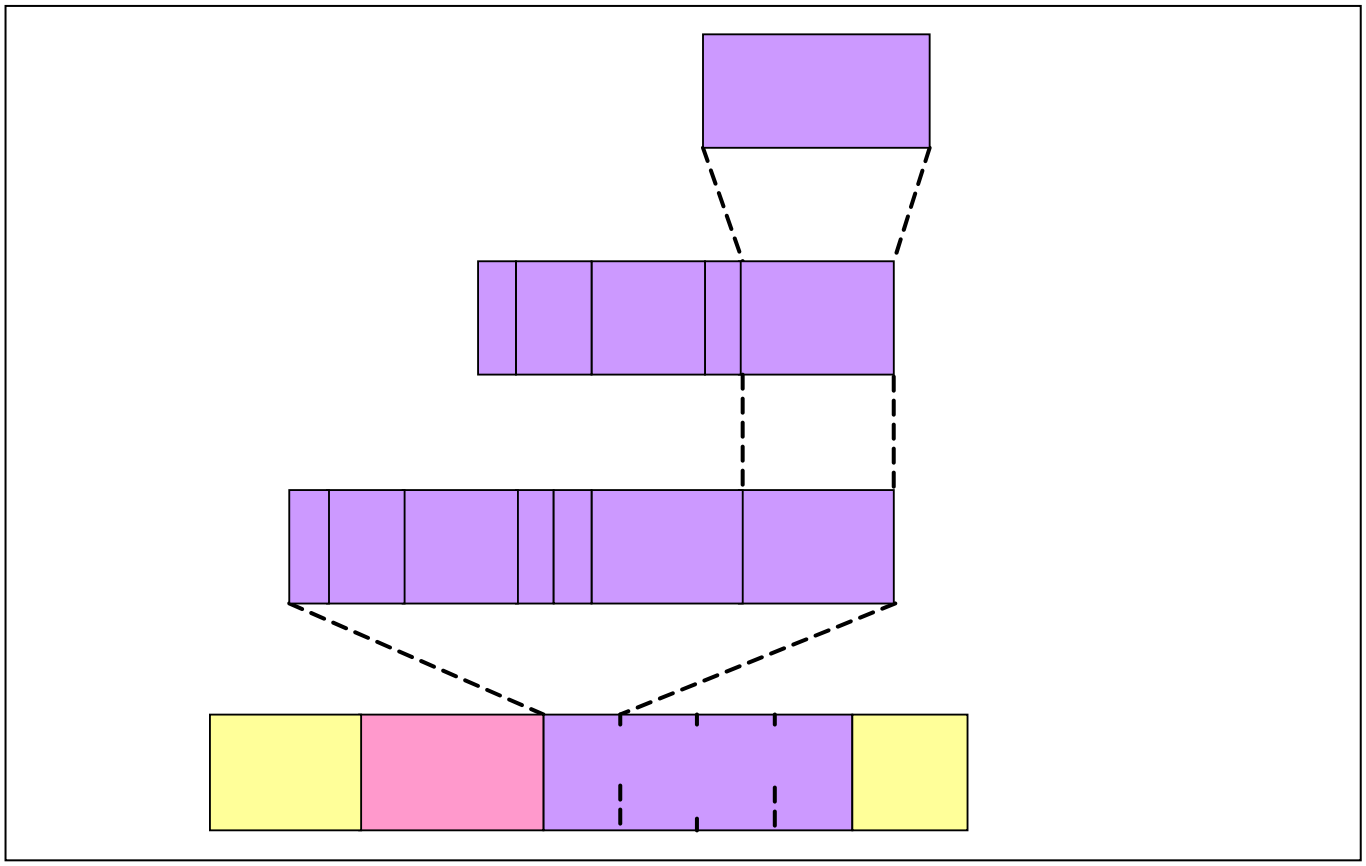
For the prevalent case for which the timeslot allocation is static and no activity detection is performed, the payload can be efficiently encoded using constant bit rate AAL1 adaptation.

The AAL1 payload type machine converts E1, T1, E3, T3, STS-1 or serial data flows into IP, MPLS or Ethernet packets, and vice versa, according to ITU-T Y.1413, Y.1453, MEF 8, MFA 4.1 and IETF RFC 5087 TDMoIP. In this mapping method, data is actually mapped into 48-byte AAL1 SAR PDUs as described in I.361.1 section 2.4.2.

#### 10.6.6.1 TDM-to-Ethernet Direction

In the TDM-to-Ethernet direction, the AAL1 payload type machine concatenates the bundle's timeslots into structures and then slices and maps the structures into 46- or 47-octet AAL1 SAR PDU payloads. After adding the AAL1 SAR PDU header and pointer as needed, the AAL1 SAR PDUs are concatenated and inserted into the payload of the layer 2/layer 3 packet.

*Figure 10-36. AAL1 Mapping, General*



The structure of the AAL1 header is shown in Table 10-21 below.

**Table 10-21. AAL1 Header Fields**

| Field | Length (bits) | Description |
|---|---|---|
| C | 1 | Indicates if there is a pointer in the second octet of the AAL1 SAR PDU. When set, a pointer exists. |
| SN | 3 | AAL1 SAR PDU sequence number |
| CRC | 3 | Cyclic redundancy code on C and SN |
| P | 1 | Even parity bit on C, SN and CRC or the even byte parity LSB for the sequence number octet (P format AAL1 SAR PDUs only) |
| E | 1 | (P format AAL1 SAR PDUs only) Even byte parity MSB for pointer octet |
| Pointer | 7 | (P format AAL1 SAR PDUs only) Indicates the next structure boundary. It is always located at the first possible position in the sequence number cycle in which a structure boundary occurs. The pointer indicates one of 93 octets (46 octets of the current AAL1 SAR PDU + 47 octets of the next AAL1 SAR PDU). P=0 indicates that the first octet of the current AAL1 SAR PDU's payload is the first octet of the structure. P=93 indicates that the last octet of the next AAL1 SAR PDU is the final octet of the structure. |

The AAL1 block supports the following bundle types:

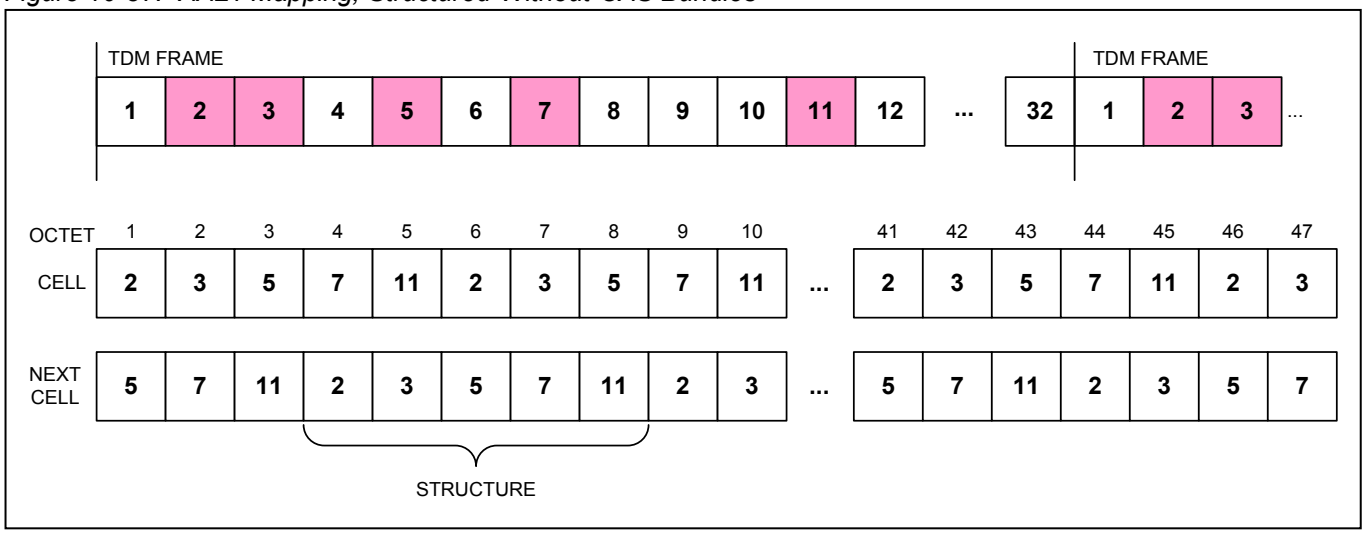
- Unstructured
- Structured
- Structured-with-CAS.

**Unstructured bundles**, for E1/T1 interfaces, support rates of N × 64 kbps, where N is the number of timeslots configured to be assigned to a bundle. Unstructured bundles may also carry traffic of the whole low-speed interface (up to 4.6 Mbps), E1/T1 interface (2.048Mbps/1.544 Mbps) and high-speed interface (up to 51.84 Mbps). The AAL1

SAR PDU payload contains 47 octets (376 bits) of TDM data without regard to frame alignment or timeslot byte alignment. All AAL1 SAR PDUs are non-P format for unstructured bundles.

**Structured bundles**, for E1/T1 interfaces, support rates of N × 64 kbps, where N is the number of timeslots configured to be assigned to a bundle. For this format, the N timeslots from one E1/T1 frame are sequentially mapped into an N-octet structure. This N-octet structure is then mapped into the AAL1 SAR PDU payload, octet-aligned. This process is repeated until all octets of the AAL1 SAR PDU payload are filled. The last octet of the payload may contain a timeslot other than the last timeslot of the structure. The remaining timeslots of the structure are mapped into the next AAL1 SAR PDU payload in the same manner and the process continues. This is illustrated in Figure 10-37.

*Figure 10-37. AAL1 Mapping, Structured-Without-CAS Bundles*




With this mapping each AAL1 SAR PDU can start with a different timeslot. To enable the far end TDMoP function to identify the start of a structure, a pointer to it is sent periodically in one of the even-numbered AAL1 SAR PDUs of every SN cycle. When this pointer is sent, a P-format AAL1 SAR PDU is used. In a P-format AAL1 SAR PDU the first byte of the payload contains the pointer while the last 46 bytes contain payload.

**Structured-with-CAS bundles**, for E1/T1 interfaces, support rates of N × 64 kbps, where N is the number of timeslots configured to be assigned to a bundle. This mapping is similar to the structured-without-CAS mapping described above except that the structure is an entire E1/T1 multiframe of the N timeslots assigned to the bundle, and a CAS signaling substructure is appended to the end of the structure. The addition of CAS only affects the structure arrangement and contents. CAS data of one timeslot is 4 bits long, meaning one octet can contain CAS data of 2 timeslots. Bundles containing an odd number of timeslots need a padding of 4 zeroes in the last CAS octet. For example, a 3-timeslot bundle of an E1 frame with CAS yields the following structure octet sequence: TS1, TS2, TS3 repeated 16 times (a whole E1 multiframe) and then CAS1+CAS2, CAS3+padding.

#### *10.6.6.2 Ethernet-to-TDM Direction*

In the Ethernet-to-TDM direction, AAL1 SAR PDUs of a bundle are being received only after the synchronization process. The synchronization process includes packet SN synchronization, AAL1 SAR PDU SN synchronization, and pointer synchronization. AAL1 SAR PDUs with CRC or parity errors in their header are discarded. Pointer mismatch imposes jitter buffer under-run and bundle resynchronization. AAL1 SAR PDU header errors or pointer errors may be ignored depending on per-bundle configuration. Missing AAL1 SAR PDUs are detected and restored in the jitter buffer.

### 10.6.7 HDLC Payload Type Machine

Handling HDLC in TDM-over-Packet ensures efficient transport of CCS (common channel signaling, such as SS7), embedded in the TDM stream or other HDLC-based traffic, such as Frame Relay, according to IETF RFC 4618 (excluding clause 5.3 – PPP) and RFC 5087 (TDMoIP).

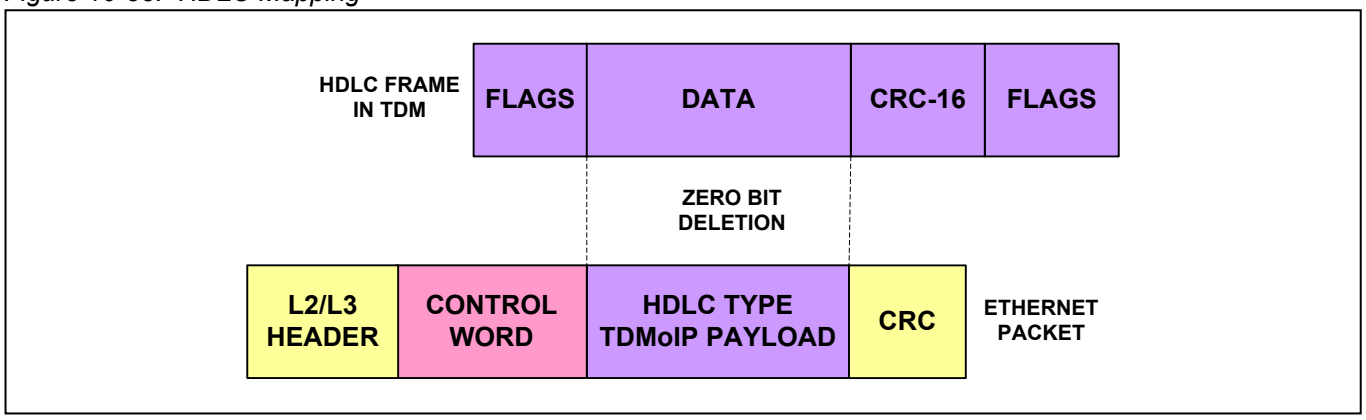
For an E1 interface, each bundle supports the rates of 16 kbps or N × 64 kbps, where N is the number of timeslots configured to be assigned to a bundle (between 1 to 32). For an T1 interface, each bundle supports the rates of 16 kbps, 56 kbps (not supported for T1 SF interface), full T1 (1.544 Mbps) or N × 64 kbps, where N varies from 1 to 24.

In the TDM-to-Ethernet direction, the HDLC block monitors flags until a frame is detected. It removes bit stuffing, collects the contents of the frame and checks the correctness of the CRC, alignment and frame length. Valid frame length is anything greater than 2 bytes and less than Tx_max_frame_size in HDLC_Bundle[n]_cfg[95:64].

Erroneous frames are discarded. Good frames are mapped as-is into the payload of the configured layer 2/3 packet type (without the CRC, flags or transparency zero-insertions).

In the Ethernet-to-TDM direction, when a packet is received, its CRC is calculated, and the original HDLC frame reconstituted (flags are added, bit stuffing is performed, and CRC is added).

*Figure 10-38. HDLC Mapping*



### 10.6.8 RAW Payload Type Machine

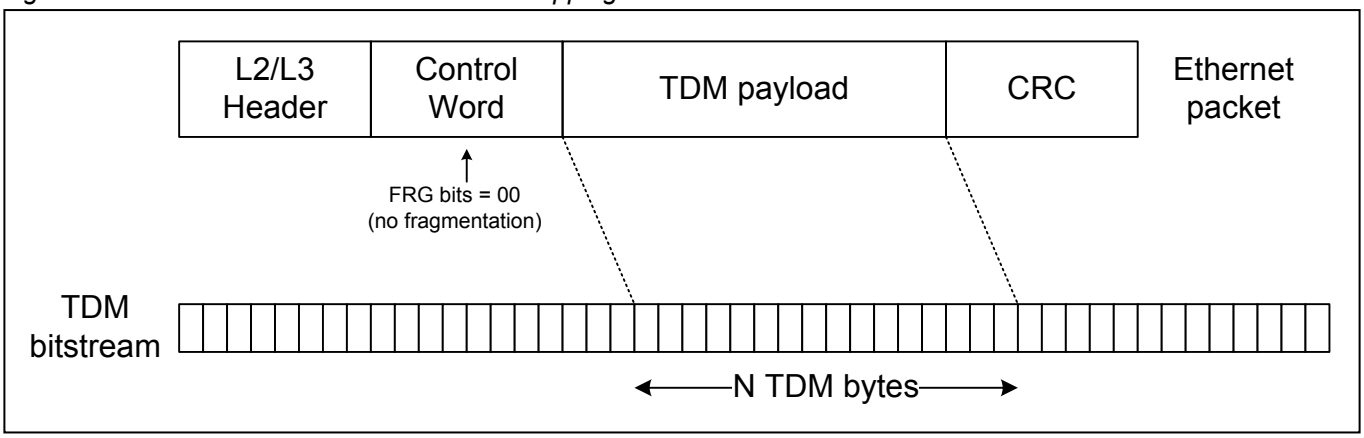
The RAW payload type machine support the following bundle types:

- **Unstructured**
  According to ITU-T Y.1413, Y.1453, MEF 8, MFA 8.0.0 and IETF RFC 4553 (SAToP).
- **Structured without CAS**
  According to ITU-T Y.1413, Y.1453, MEF 8, MFA 8.0.0 and IETF RFC 5086 (CESoPSN).
- **Structured with CAS**
  According to ITU-T Y.1413, Y.1453, MEF 8, MFA 8.0.0 and IETF RFC 5086 (CESoPSN).

#### 10.6.8.1 Unstructured

Unstructured bundles usually carry the data of a whole TDM port. This port may be low-speed such as an E1, T1 or Nx64k bit stream or high-speed such as an E3, T3 or STS1 signal. In an unstructured bundle, the packet payload is comprised of N bytes of the TDM stream without regard for byte or frame alignment. In the receiving device, the TDM data is extracted from the packet payload and inserted as a bit stream into the jitter buffer, from which it is then extracted and sent to the TDM port.

*Figure 10-39. SAToP Unstructured Packet Mapping*




The packetization delay of an unstructured (SAToP) bundle is: T = N x 8 x the bit time of the TDM interface.

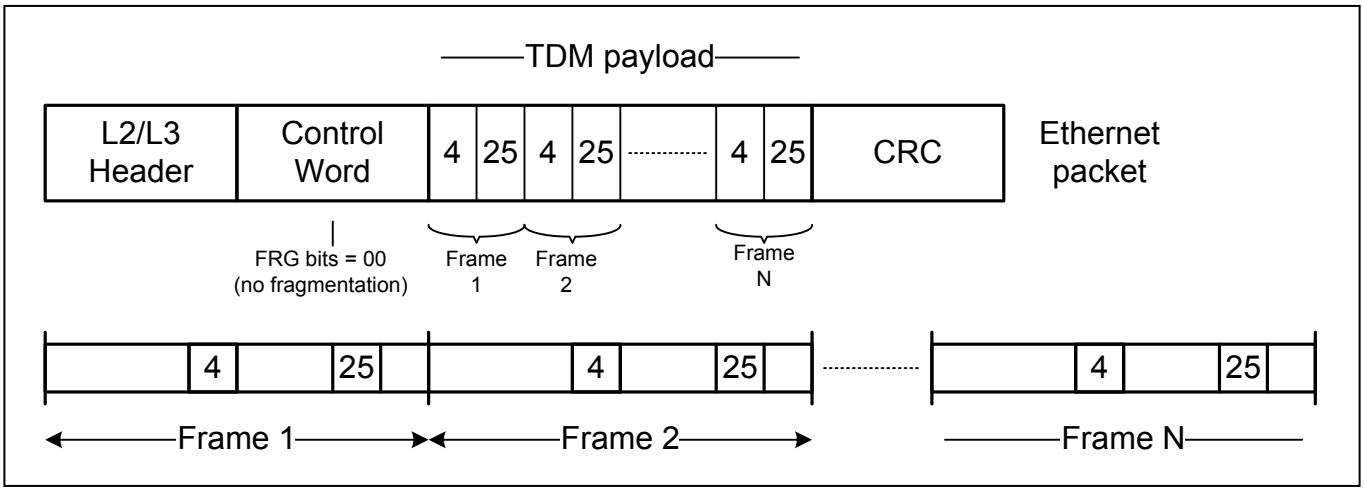
The minimum packetization time of an Ethernet packet for an unstructured (SAToP) bundle is as follows:

- 60 μs for high speed mode
- 125 μs for low speed mode

#### 10.6.8.2 Structured without CAS

In a structured-without-CAS bundle, the packet payload is comprised of the assigned timeslots from N TDM frames as illustrated in Figure 10-40.

*Figure 10-40. CESoPSN Structured-Without-CAS Mapping*



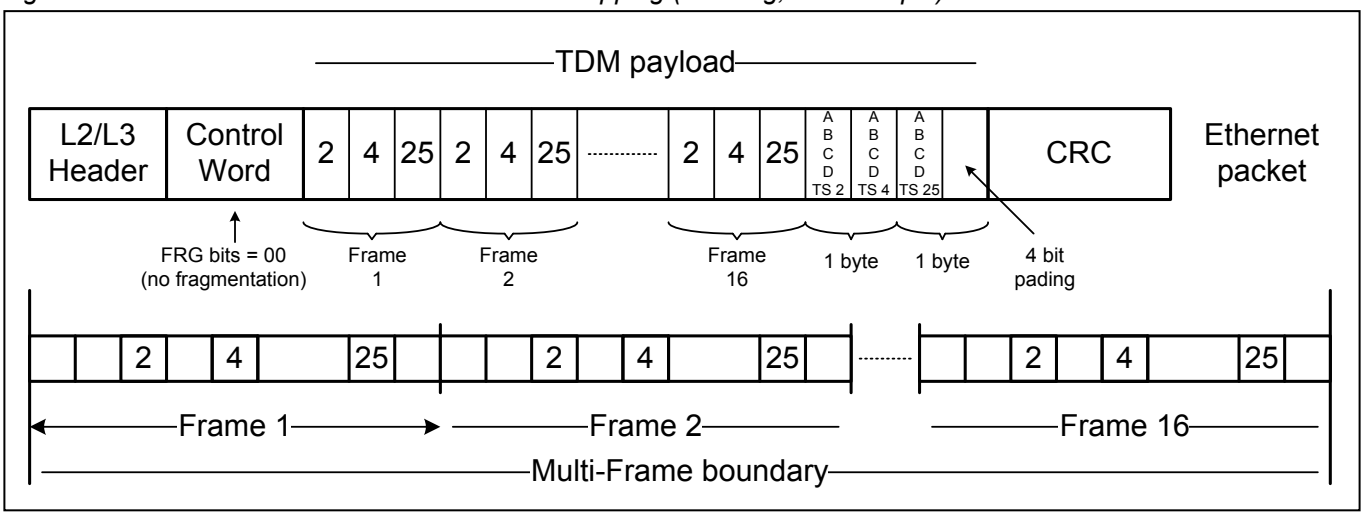

The packetization delay of a CESoPSN structured-without-CAS bundle is: T = N x 125 μs (i.e. N x the frame rate)

The minimum packetization time of an Ethernet packet for a structured (with or without CAS) bundle is 125 μs.

#### *10.6.8.3 Structured with CAS (without Fragmentation)*

In a structured-with-CAS bundle, the packet payload is comprised of the assigned timeslots from all the TDM frames in a multiframe (e.g. 16 frames for E1) followed by the CAS signaling substructure, which contains the CAS info for the assigned timeslots.

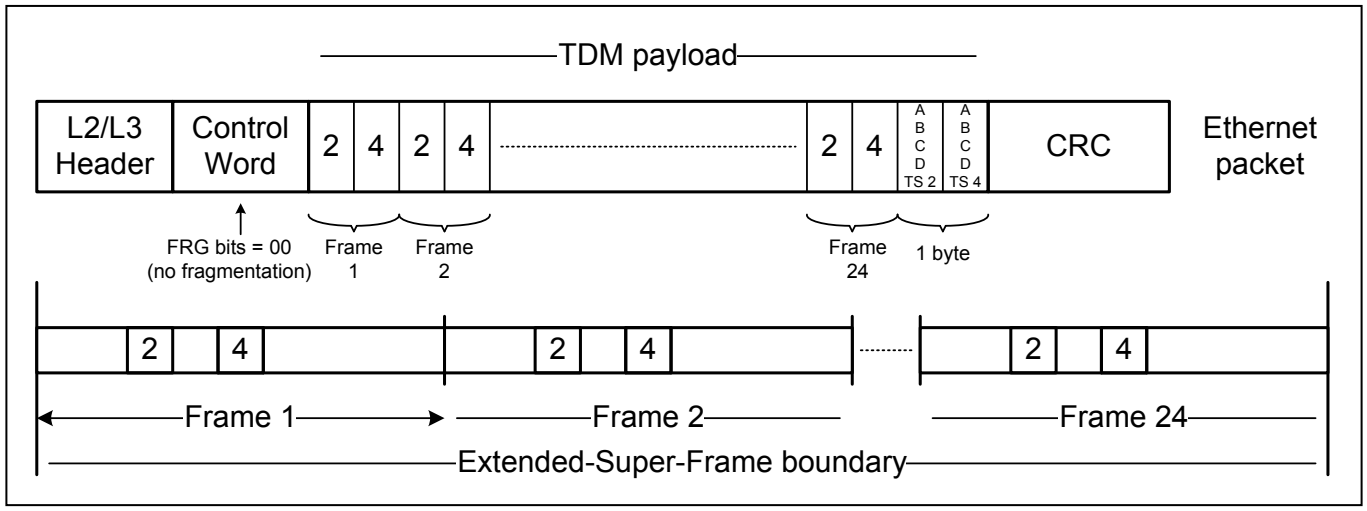
*Figure 10-41. CESoPSN Structured-With-CAS Mapping (No Frag, E1 Example)*




The minimum packetization time of an Ethernet packet for a structured (with or without CAS) bundle is 125 μs.

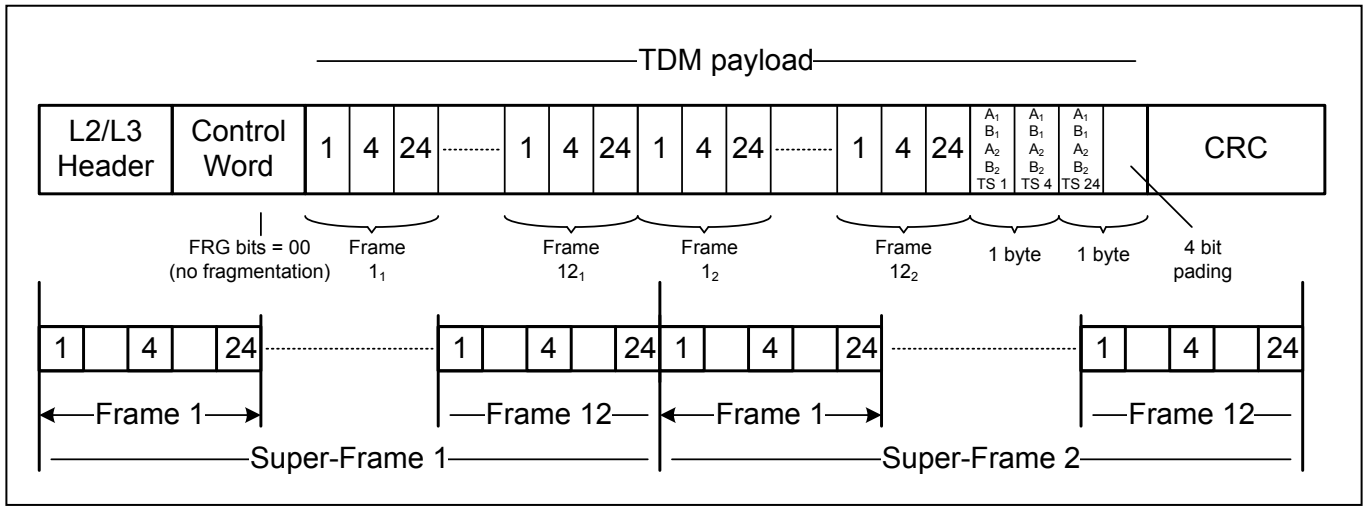
The minimum TDM payload of an Ethernet packet for a structured (with or without CAS) bundle is 8 bytes.

*Figure 10-42. CESoPSN Structured-With-CAS Mapping (No Frag, T1-ESF Example)*




In T1 SF, the multiframe structure is composed of 2 superframes resulting total of 24 TDM frames. The CAS info at the end of the structure contains the CAS info of the 2 corresponding superframes as well.

*Figure 10-43. CESoPSN Structured-With-CAS Mapping (No Frag, T1-SF Example)*




The packetization delay of a CESoPSN structured-with-CAS bundle (not fragmented) is as follows:

- Multiframed E1: T = 2 ms
- T1 SF, ESF: T = 3 ms

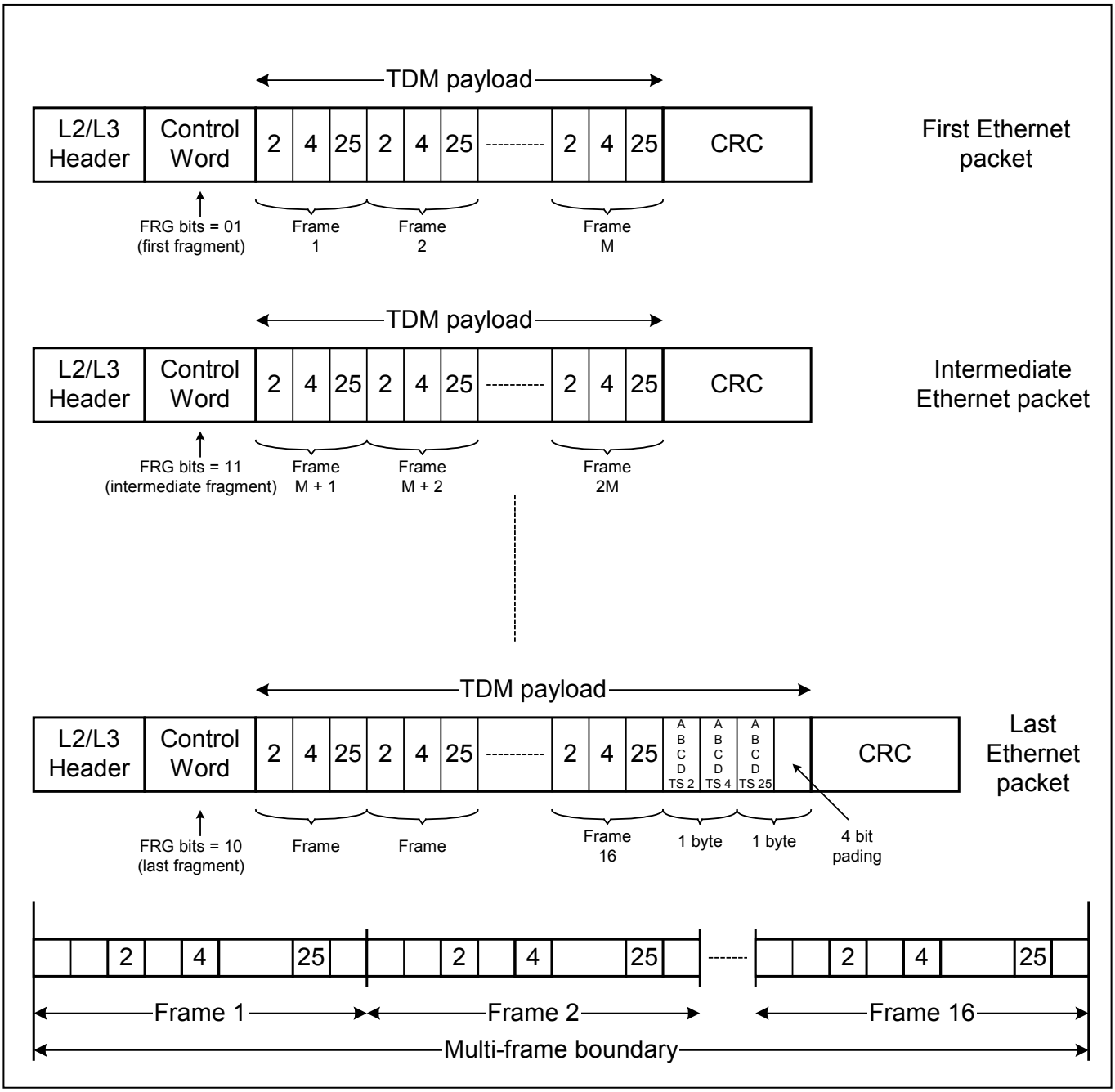
#### *10.6.8.4 Structured-with-CAS (with Fragmentation)*

In order to reduce the packetization delay of structured-with-CAS bundle, the CESoPSN standard supports the option of fragmentation. In this mode, the multiframe data structure is fragmented among several packets. Each packet contains M TDM frames of the assigned timeslots. The last packet also contains the entire multiframe CAS substructure. Because of that, there is limited number of allowed "M" values:

- For multiframed E1: M = 1, 2, 4, 8, 16 (16 means single packet with no fragmentation)
- For T1 SF: M = 1, 2, 3, 4, 6, 8, 12, 24 (24 means single packet with no fragmentation)
- For T1 ESF: M = 1, 2, 3, 4, 6, 8, 12, 24 (24 means single packet with no fragmentation)

The packetization delay of a CESoPSN structured-with-CAS bundle (with fragmentation) is: T = M x 125 μs.

*Figure 10-44. CESoPSN Structured-With-CAS Mapping (Frag, E1 Example)*



### 10.6.9 SDRAM and SDRAM Controller

The device requires an external SDRAM for its operation. The following describes how the TDMoP block and the CPU use the SDRAM:

The TDMoP block accesses these sections of the SDRAM:

- Transmit buffers section
  This area stores outgoing packets created by the payload-type machines. It is a 1-Mbyte area with base address specified by the Tx_buf_base_add field in General_cfg_reg1. The actual amount of SDRAM used in the transmit buffers section depends on the number of open bundles and the number of buffers assigned to each bundle.
- Jitter buffer data section
  This area stores incoming TDM data after it has been extracted from received packets by the payload-type machines. It is a 2-Mbyte area with base address specified by the JBC_data_base_add field in General_cfg_reg1. The actual amount of the SDRAM used in the jitter buffer data section depends on the configuration (most applications allocate only 0.5 Mbyte).
- Jitter buffer signaling section:
  This area stores incoming TDM signaling information after it has been extracted from received packets by the payload-type machines. It is a 32-kbyte area, with base address specified by the JBC_sig_base_add field in General_cfg_reg1. This section is used only when structured-with-CAS bundles have been opened.

The CPU uses the SDRAM as follows:

- The CPU may utilize the sections of SDRAM not used by the TDMoP block in order to send/receive packets through the CPU queues/pools.
- The CPU accesses the transmit buffers section in order to initialize the buffer headers before opening a bundle.

The built-in SDRAM controller allows glueless connection to an external SDRAM (the TDMoP block supplies the SDRAM clock). Supported SDRAM devices are listed in section 15.6.

The TDMoP block typically uses from 1.5 to 3 MB of SDRAM space, depending on configuration. The CPU may use the rest of the memory.

The supported resolutions of CPU access to the SDRAM are shown below.

**Table 10-22. SDRAM Access Resolution**

| Data Bus Width | Access to SDRAM |
|---|---|
| 32 bits | 8, 16, 32 bit |
| 16 bits | 8, 16 bit |

Prior to operation, the SDRAM controller configuration bits (see the General_cfg_reg0 register) must be configured. First, the CPU must set the configuration bits while maintaining the Rst_SDRAM_n bit low (0). Then, it should deassert the Rst_SDRAM_n bit. The Rst_SDRAM_n bit must not be changed during operation.

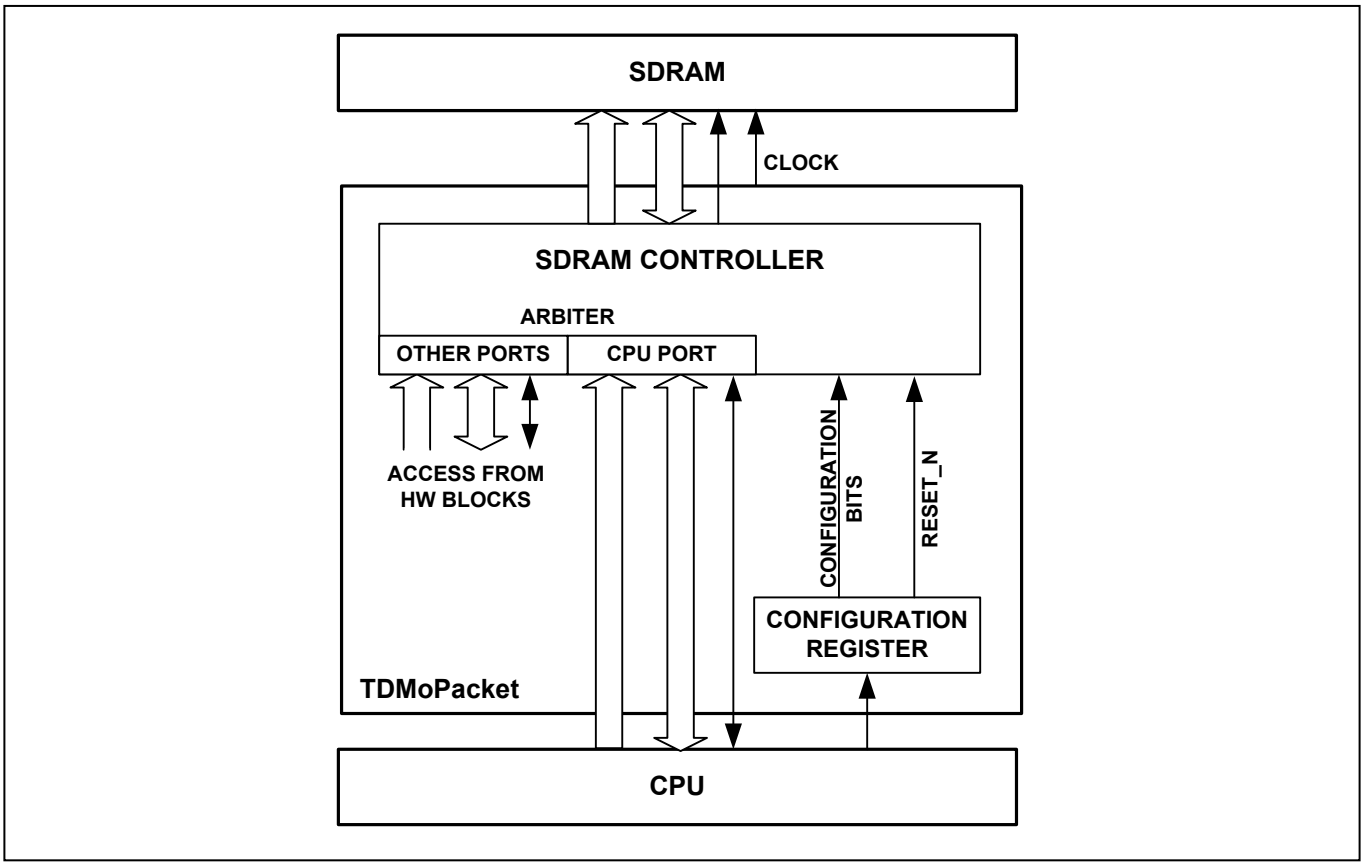
The SDRAM Controller operates at either 50 or 75 MHz with the following CAS latency options:

**Table 10-23. SDRAM CAS Latency vs. Frequency**

| Frequency [MHz] | CAS Latency [clock cycles] |
|---|---|
| 50 | 2 |
| 75 | 2 or 3 |

During operation, the controller's arbiter receives access requests from various internal hardware blocks and the CPU and grants access permissions based on predefined priorities. The controller automatically refreshes the external SDRAM approximately once every 15 μs.

*Figure 10-45. SDRAM Access through the SDRAM Controller*

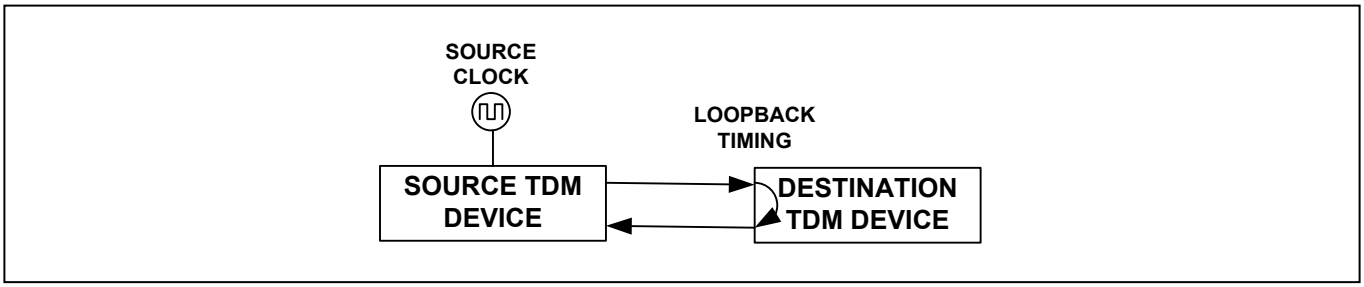



### 10.6.10 Jitter Buffer Control (JBC)

#### 10.6.10.1 Jitter Buffer Application

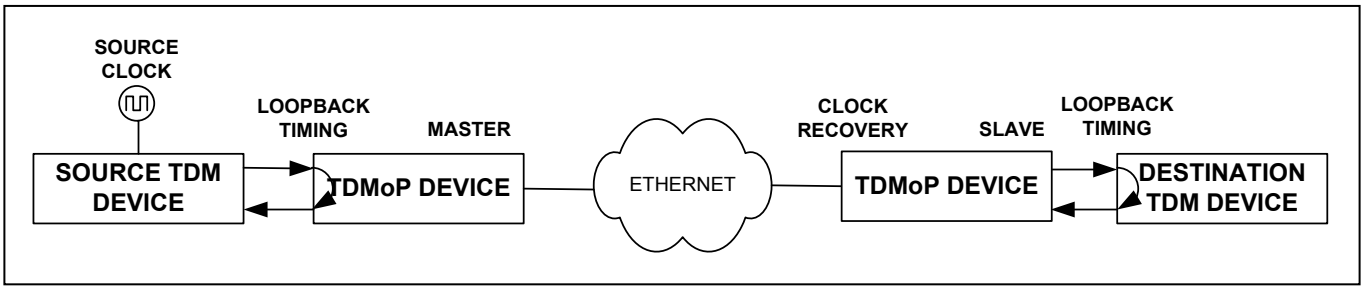
Routinely in TDM networks, destination TDM devices derive a clock from the incoming TDM signal and use it for transmitting data as depicted in Figure 10-46. This is called loopback timing.

*Figure 10-46. Loop Timing in TDM Networks*




When replacing the physical TDM connection with an IP/MPLS network and two TDM-over-Packet devices as shown in Figure 10-47 below, the receiving TDM-over-Packet device (slave) receives packets with variable delays (packet delay variation). After processing, the slave TDMoP device should send TDM data to the destination TDM device at the same clock rate at which the TDM data was originally sent by the source TDM device. To achieve this, the device works in clock recovery mode to reconstruct the source TDM clock to allow the destination TDM device to still work in loopback timing mode.

*Figure 10-47. Timing in TDM-over-Packet*




The jitter buffer, located in the SDRAM, has two main roles:

- Compensate for packet delay variation
- Provide fill level information as the independent variable used by the clock recovery machines to reconstruct the TDM clock on a slave TDMoP device.

The data enters the buffer at a variable rate determined by packet arrival times and leaves it at a constant TDM rate. In clock recovery mode, the amount of data in the jitter buffer (the "fill level") steers the clock recovery mechanism.

#### *10.6.10.2 Jitter Buffer Configuration*

Separate areas are allocated in the external SDRAM for TDM data and for signaling, as described in section 10.6.9.

In low-speed mode (High_speed=0 in General_cfg_reg0) both data and signaling areas are divided into eight identical sections, one for each E1/T1/Nx64 interface. These section are further divided as follows:

- In E1/T1 structured mode, each per-port data section contains the data of 32 timeslots for E1 or 24 timeslots for T1 (a total of 32*8=256 timeslots for all eight interfaces). Each E1/T1 timeslot is allocated a maximum of 4 kB of space (128kB per interface and a total of 1024 kB for all eight interfaces).
- Each signaling section is divided into multiframe sectors, with each sector containing the signaling nibbles of up to 32 timeslots (total of 64 kB for all 8 interfaces).
- In serial interface mode or E1/T1 unstructured mode, there is no per-timeslot allocation. The jitter buffer is divided into eight identical sections, one for each interface (each section is 512 kB for HDLC bundles or 128 kB for other bundle types).

In high-speed mode (E3, T3, STS-1), the jitter buffer is arranged as one large buffer without division into sections (total of 512 kB).

The Jitter Buffer maximum depth in time units (seconds) is calculated according to the following formula:

$$\frac{1}{2} \times \textit{Buffer area per interface} \times \frac{8}{\textit{Rate}}$$

where:

| | | |
|---|---|---|
| ½ | = | Two halves of the buffer |
| Buffer area per interface | = | 512 kB for a single high-speed interface or 128 kB for a low-speed interface |
| 8 | = | Number of bits per byte |
| Rate | = | Transmission rate (e.g., 2.048 Mbps) |

For T1 structured-with-CAS, multiply the above formula by 0.75.

The jitter buffer depth is defined by the Rx_max_buff_size parameter found in the Bundle Configuration Tables. When the jitter buffer level reaches the value of Rx_max_buff_size, an overrun situation is declared.

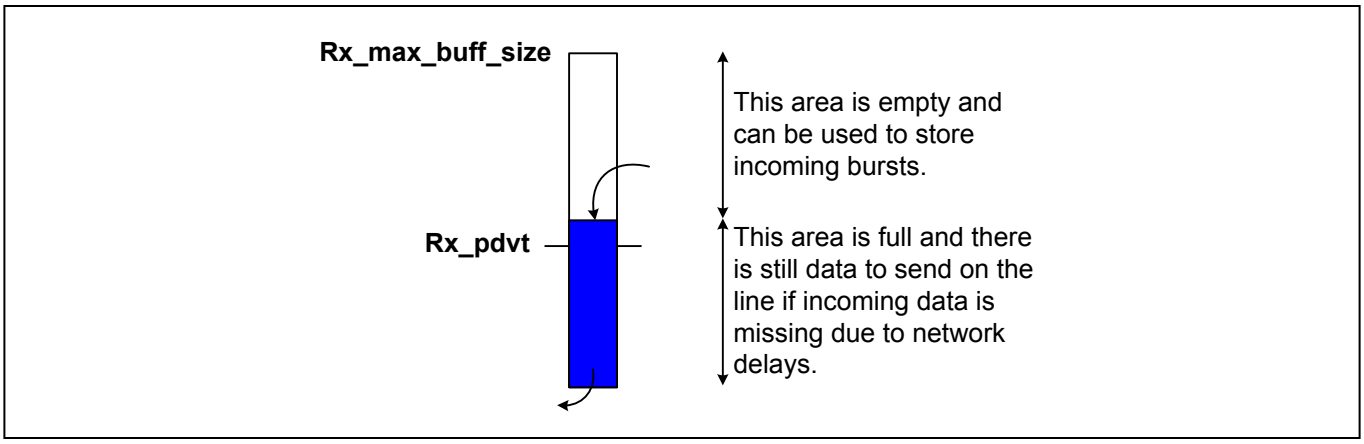
The Rx_PDVT parameter (also found in the Bundle Configuration Tables) defines the amount of data to be stored in the jitter buffer to compensate for network delay variation. This parameter has two implications:

- Rx_PDVT defines the chip's immunity to the Ethernet network delay variation.
- The data arriving from the network is delayed by Rx_PDVT before it is read out of the jitter buffer and transmitted on the TDM pins.

Rx_PDVT must be smaller than Rx_max_buff_size. Also, the difference between Rx_max_buff_size and Rx_PDVT must be larger than the time that it takes to create a packet (otherwise an overrun may occur when the packet arrives). Typically, the recommended value for Rx_max_buff_size is 2* Rx_PDVT + PCT (packet creation time). This provides equal immunity for both delayed and bursty packets.

Configuring the jitter buffer parameters correctly avoids underrun and overrun situations. Underrun occurs when the jitter buffer becomes empty (the rate data is entering the buffer is slower than the rate data is leaving). When an underrun occurs the TDMoP block transmits conditioning data instead of actual data towards the TDM interface. The conditioning data is specified by the Receive SW Conditioning Octet Select table for TDM data and the location specified by Rx_CAS_src (SDRAM or Receive SW CAS) for signaling. Overrun occurs when the jitter buffer is full and there is no room for new data to enter (the rate data is leaving the buffer is slower than the rate data is entering). Underrun and overrun require special treatment from the TDMoP hardware, depending on the bundle type.

*Figure 10-48. Jitter Buffer Parameters*




The JBC uses a 64 by 32 bit Bundle Timeslot Table to identify the assigned timeslots of each active bundle. The index to the table is the bundle number. The CPU must configure each active bundle entry (setting a bit means that the corresponding timeslot is assigned to this bundle). For unstructured bundles, the whole bundle entry (all 32 bits) must be set.

Jitter buffer statistics are stored in a 256-entry table called the Jitter Buffer Status Table. Each TDM port has 32 dedicated entries, one per timeslot. This table stores the statistics of the active jitter buffer for each active bundle. A configurable parameter called Jitter_buffer_index located in the timeslot assignment tables (section 11.4.5) points to the entry in the Jitter Buffer Status Table where the associated jitter buffer statistics are stored. The value of the Jitter_buffer_index should be set as follows:

- For AAL1/HDLC/RAW structured bundles: the Jitter_buffer_index value is the number of the lowest timeslot in the bundle. For example, if the bundle consists of timeslots 2, 4, 17 on port 3, Jitter_buffer_index=0x2.
- For unstructured bundles the Jitter_buffer_index value is 0x0.

#### 10.6.10.3 Jitter Buffer Status and Statistics

The CPU accesses the Jitter Buffer Status Table using the Jitter_buffer_index as described above. The status table contains the current jitter buffer status (such as, the current jitter buffer level and its current state (OK, underrun or overrun).

The status table also contains two variables, Minimal_level and Maximal_level, which report the minimum and maximum fill levels of the jitter buffer since the last time the two fields were read (available for AAL1/RAW bundles only). These variables provide information about network packet delay variation. For example, using these values, the CPU can calculate the margins from the top (Rx_max_buff_size) and the bottom of the jitter buffer. If there is margin, CPU software may want to reduce Rx_PDVT to reduce the latency added by the jitter buffer to the incoming TDM data.

#### 10.6.10.4 Jitter Buffer Response to Packet Loss and Misordering

The payload-type machines detect that a packet was lost by sequence number error in AAL1/RAW. If a packet is lost, conditioning data (specified by the receive software conditioning registers in section 11.4.12) is inserted into the jitter buffer in place of the lost data to maintain bit integrity (i.e. the number of bits that are inserted into the jitter buffer must equal the number of bits that were transmitted by the far end).

If a packet is misordered in a RAW bundle (for example, the packet with the sequence number N arrives after the packet with sequence number N+1) it is reordered by the RAW payload-type machine, and its data is inserted into the appropriate location in the jitter buffer, assuming that the data in this location has not been transmitted to the TDM port yet.

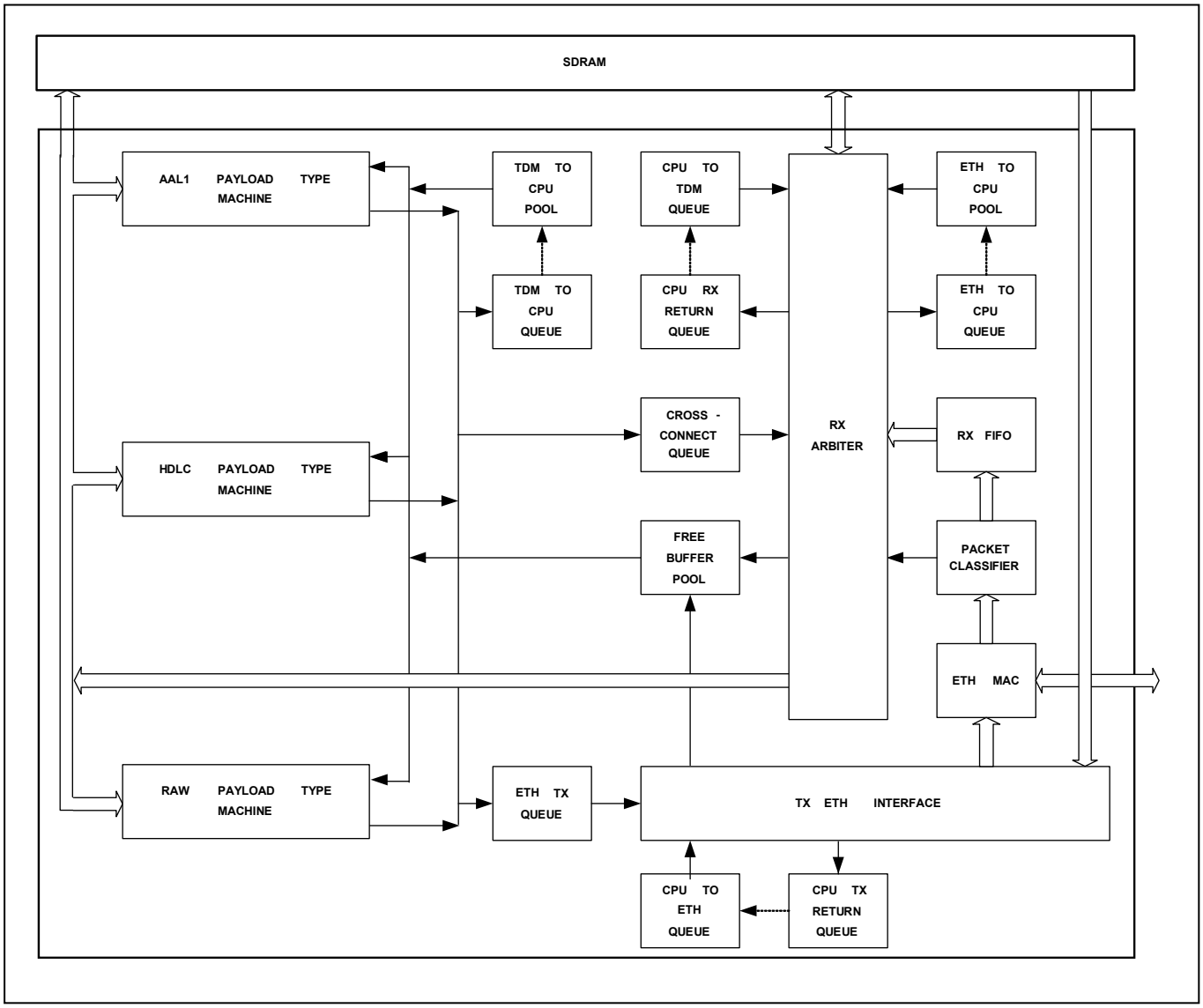
### 10.6.11 Queue Manager

Data flows through the TDMoP block in the following directions:

- TDM to Ethernet (implemented in HW)
- Ethernet to TDM (implemented in HW)
- TDM to TDM (cross-connect, implemented in HW)
- TDM to CPU
- CPU to TDM
- CPU to Ethernet
- Ethernet to CPU.

These data flows are illustrated in Figure 10-49. Each data flow is described in a subsection below.

*Figure 10-49. TDM-over-Packet Data Flow Diagram*




### *10.6.11.1 Buffer Descriptor*

Data is transferred between the Ethernet MAC, internal payload-type machines and the external CPU by means of buffers in the SDRAM. Payload data is stored in 2 kB SDRAM buffers along with a buffer descriptor located in the buffer's first dwords. The buffer pointers are managed inside the TDMoP block and are stored in queues, pools, and other internal blocks. Queues store pointers to SDRAM buffers containing packet data to be processed, while pools store pointers to empty buffers. The pointers are passed from one block to another. Only the block owning the pointer can access the associated buffer.

The size of the buffer descriptor size depends on the internal path it is used for:

TDM → TDM, TDM → CPU and CPU → TDM: One dword
TDM → ETH, CPU → ETH and ETH → TDM: Two dwords
ETH → CPU: Three dwords

The fields of the buffer descriptor dwords are described in the sections below.

#### 10.6.11.2 Buffer Descriptor First Dword

Used for all paths. Located at offset 0x0 from the start of the buffer.

**Table 10-24. Buffer Descriptor First Dword Fields (Used for all Paths)**

| Bits | Data Element | Description |
|---|---|---|
| [31] | MPLS/MEF/L2TIPV3 or UDP/IP-specific OAM | For ETH → TDM and for CPU → TDM indicates that the buffer holds a packet with MPLS / MEF / L2TPv3 Ethertype. For ETH → CPU indicates that the buffer holds a UDP/IP-specific OAM packet. |
| [30] | RST | RX Reset command (the bundle is in reset process).<br>For ETH→TDM and for CPU→TDM: used by the Packet Classifier or by the CPU to inform the next blocks in flow that the bundle was reset. The buffer contains no real data. |
| [29:27] | Buffer contents | 000: Backwards-compatible (experimental) format packet going to the AAL1 payload-type machine<br>001: Standard format packet going to the AAL1 payload-type machine<br>010: Reserved<br>011: Non-TDMoP/MPLS packet (this buffer isn't assigned to any bundle)<br>100: Standard format packet going to the HDLC payload-type machine<br>101: Reserved<br>110: Standard format packet going to the RAW payload-type machine<br>111: Backwards-compatible (experimental) format packet going to the HDLC payload-type machine |
| [26:16] | Length/Rst_Ts | Packet Length or Payload Length<br>For TDM→CPU, TDM→TDM, CPU→TDM and ETH→TDM: payload length in bytes (received bytes + control word if present + RTP header bytes in case of MPLS/MEF packet using RTP and control word)<br>For TDM→ETH, ETH→CPU and CPU→ETH: packet length in bytes, without CRC<br>For Buffer Contents =101: total length of packets concatenated in the buffer, in bytes<br>For RST packets: the reset timeslot number<br><br>*Note: Length must be less than 1951 bytes.*<br>*Note: Offset and Length sum must be less than 2000 bytes.* |
| [15] | Reserved | Must be set to zero. |
| [14:8] | Offset | For ETH→CPU, TDM→ETH and CPU→ETH: offset in bytes from start of buffer to start of packet<br>For ETH→TDM, TDM→CPU, CPU→TDM: offset in bytes from start of buffer to start of payload or to the control word if present<br>For TDM→TDM: bits 13-8 hold the internal bundle number from which the buffer has been transmitted<br><br>For CPU→ETH, when Buffer Content (above) is different than 011, must be calculated as follows: tx_payload_offset – header_length<br><br>*Note: Offset and Length sum must be less than 2000 bytes.*<br>*Note: header_length is the number of bytes from start of packet to the control word (or to start of the payload if control word is not used).* |
| [7] | HW/SW Type | The pool the buffer has been extracted from and should be returned to.<br>0: HW buffers pool<br>1: SW buffers pool<br>For packets coming from Ethernet:<br>0: destination = payload-type machines<br>1: destination = CPU |
| [6] | RTP | For ETH→TDM, ETH→CPU, TDM→TDM and CPU→ TDM indicates whether the packet includes an RTP header. |
| [5:0] | Bundle number | For TDM→ TDM: destination internal bundle number.<br>For any other bundle: packet internal bundle number |

#### 10.6.11.3 Buffer Descriptor Second Dword

Located at offset 0x4 from the start of the buffer.

##### 10.6.11.3.1 TDM → ETH and CPU → ETH Packets

**Table 10-25. Buffer Descriptor Second Dword Fields (TDM → ETH and CPU → ETH)**

| Bits | Data Element | Description |
|---|---|---|
| 31:15 | Reserved | Must be set to zero. |
| 14 | Stamp | Indicates whether the packet should be time-stamped. Valid only for OAM and for non-TDMoP packets. Otherwise ignored. |
| 13:7 | Ts_offset | Indicates the number of dwords from start of buffer to timestamp location. Valid only for OAM and for non-TDMoP packets where Stamp bit is set above. |
| 6:0 | Hdr2_length | The second header length in bytes not including control word or RTP header (The offset to the second header from start of the buffer is 0x782). Limited to 122 bytes and valid only for AAL1, CESoPSN and SAToP bundles where the Protection_mode setting of the bundle equals to “11” or “10”. |

##### 10.6.11.3.2 ETH → CPU Packets

**Table 10-26. Buffer Descriptor Second Dword Fields (ETH → CPU)**

| Bits | Data Element | Description |
|---|---|---|
| 31:30 | Reserved | Must be set to zero. |
| 29 | Ipv6 | IP packet with IP VER = 6 |
| 28 | Ipv4 | IP packet with IP VER = 4 |
| 27 | MEF_OAM | MEF OAM packet, i.e. Ethertype equal to Mef_oam_ether_type setting |
| 26 | VCCV_OAM | VCCV OAM packet |
| 25:24 | No. of MPLS labels | Number of MPLS labels. Equal to “11” for packet with more than 3 labels. |
| 23 | 802.3 | 802.3 packet |
| 22 | Ethernet | Ethernet packet |
| 21 | Reserved | Must be set to zero. |
| 20 | L2TPv3/IP | L2TPv3/IP packet |
| 19 | Two_VLAN tag | Packet with two VLAN tags |
| 18 | VLAN tag | Packet with one/two VLAN tags |
| 17 | UDP/IP | UDP/IP packet |
| 16 | IP | IP packet (with any IP VER) |
| 15 | MEF | MEF packet, i.e. Ethertype equal to Mef_ether_type setting |
| 14 | MPLS | MPLS packet, i.e. packet’s Ethertype equal to 0x8847 or 0x8848 |
| 13:11 | Reserved | |
| 10 | Mpls_over_3_lbls | MPLS packet with more than 3 labels |
| 9 | Unicast_not_mine | Unicast packet with destination address different than MAC addresses |
| 8 | cpu_dst_eth_type | Packet with Ethertype equal to CPU_dest_ether_type setting |
| 7 | OAM | OAM packet |
| 6 | bndl_num_not_exist | A TDM-over-Packet/MPLS/MEF packet destined to the chip but with a bundle identifier that does not match any of one of the chip’s OAM bundle numbers or one of the bundle identifiers assigned to the chip’s internal bundles. |
| 5 | not_tdmoip | UDP/IP packet with destination/source UDP port number different than TDMoIP_port_num1 and TDMoIP_port_num2 |
| 4 | ip_not_udp_l2tpv3 | IP packet with protocol different than UDP or L2TPv3 |
| 3 | arp_chip_ip | ARP packet with destination IP address equal to one of the chip’s IPv4 addresses |
| 2 | unknown_eth_type | A packet with Ethertype different than IP, MPLS, ARP, MEF, MEF OAM or CPU Ethertypes. |
| 1 | not_chip_ip | IP packet with destination IP address different than the chip’s IP addresses |
| 0 | arp_not_chip_ip | ARP packet with destination IP address different than the chip’s IP addresses |

#### 10.6.11.4 Buffer Descriptor Third Dword

Used for ETH → CPU packets. Located at offset 0x8 from start of the buffer.

**Table 10-27. Buffer Descriptor Third Dword Fields (ETH → CPU)**

| Bits | Data Element | Description |
|---|---|---|
| 31:0 | Timestamp | 32 bits timestamp latched by the packet classifier upon packet reception. Timestamp resolution is 100 μs or 1 μs as specified by the OAM_timestamp_resolution field in General_cfg_reg0. |

#### 10.6.11.5 RX Arbiter

The RX arbiter constantly checks for available packets in the Rx FIFO, the CPU-to-TDM queue and the cross-connect queue. It can do one of the following:

- Pass a packet from the Rx FIFO to the payload-type machines
- Pass a packet from the Rx FIFO to the external SDRAM and insert its pointer into the ETH-to-CPU queue
- Extract a pointer from the cross-connect queue and pass a packet from the external SDRAM into the payload-type machines
- Extract a pointer from the CPU-to-TDM queue and pass a packet from the external SDRAM into the payload-type machines.

In general, the Rx arbiter handles packets according to the following priorities:

1. Cross-connect queue
2. Rx FIFO (i.e., packets that arrive from the Ethernet port)
3. CPU-to-TDM queue.

The Rx_fifo_priority_lvl field in General_cfg_reg0 specifies a priority level for the Rx FIFO. Whenever the fill level of the Rx FIFO is above this threshold, the Rx FIFO becomes the highest priority for the Rx arbiter rather than the Cross-connect queue until the fill level of the Rx FIFO drops below the threshold.

#### 10.6.11.6 TX Ethernet Interface

The TX Ethernet interface first checks the Ethernet TX queue. If the queue is not empty, it extracts a pointer, passes the buffer data from the SDRAM to the Ethernet MAC, and returns the pointer to the free buffer pool. If the TX Ethernet queue is empty, the TX Ethernet Interface checks the status of the CPU-to-Ethernet queue. If the queue is not empty, it extracts a pointer, transfers buffer data to the Ethernet MAC, and returns the buffer to the CPU TX Return queue.

#### 10.6.11.7 Free Buffer Pool

The free buffer pool mechanism explained below is used for the TDM-to-Ethernet and TDM-to-TDM flows.

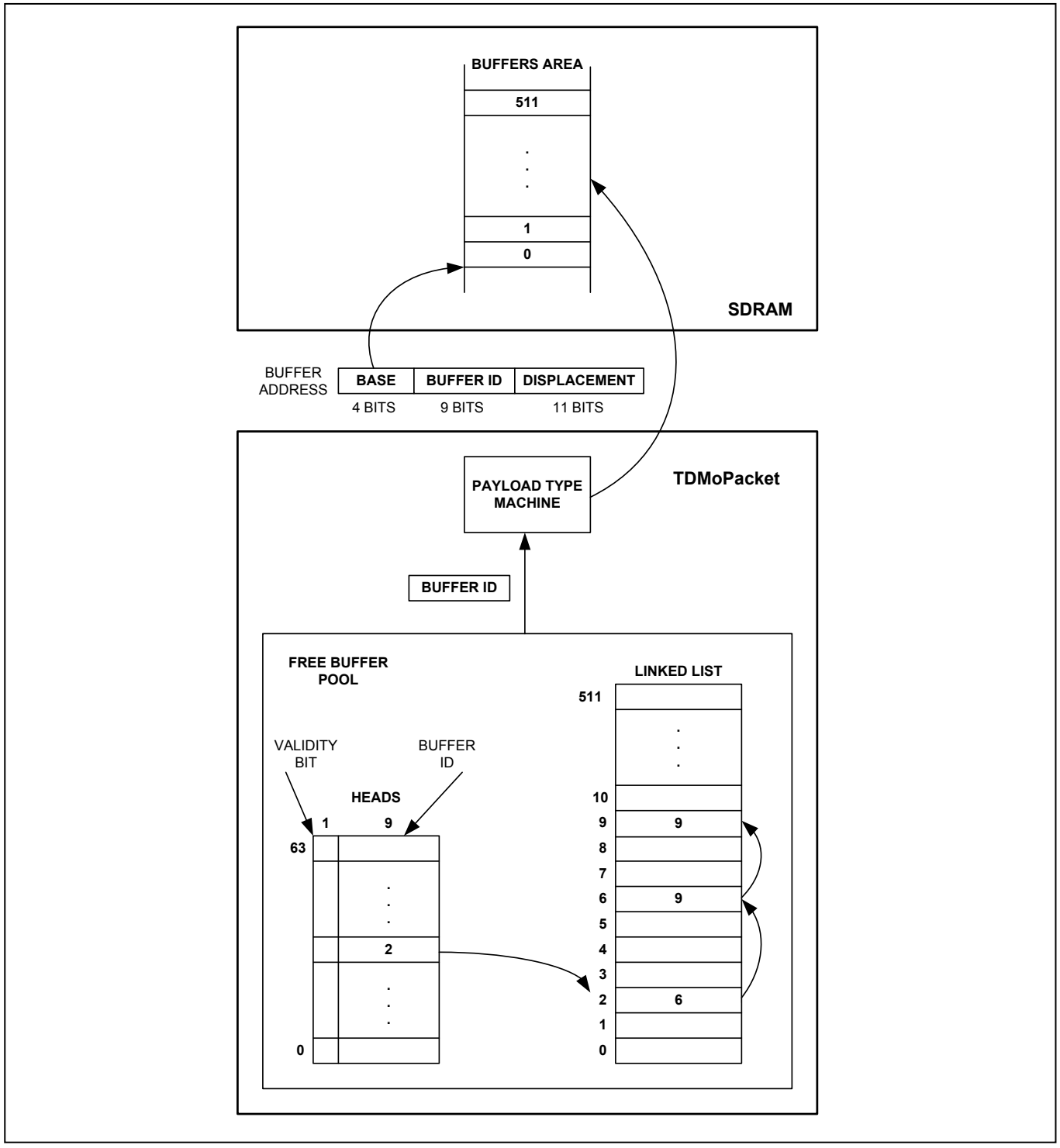
Before the payload-type machines can process any data, the CPU must initialize the free buffer pool. The free buffer pool contains pointers to SDRAM buffers that are used by the payload-type machines to store packets. There are a total of 512 SDRAM buffers. The CPU needs to pre-assign (statically) these SDRAM buffers to each bundle. The number of buffers allocated per specific bundle depends on the number of timeslots in the bundle. It is recommended to assign 4 buffers per timeslot.

The buffers are located in a continuous area in the SDRAM. The buffer address consists of the base address, the buffer number and the displacement within the buffer. The base address is specified by the Tx_buf_base_add field in General_cfg_reg1. Free buffer numbers are contained in linked lists, with a head pointing to the first buffer, each buffer pointing to the next buffer and the last buffer pointing to itself. There are 64 heads (one per bundle), each one containing a validity indication bit (MSB) and another 9 bits pointing to the first free buffer in the linked list. The register descriptions for the Per-Bundle Head Pointers and Per-Buffer Next-Buffer Pointers are in section 11.4.7.

The CPU must define the number of buffers for each bundle by initializing the linked list for the bundle. Software prepares these buffers by writing the Ethernet, IP/MPLS/L2TPv3/MEF headers in advance, so that the payload-type machines need only to write the packet payload. Since the headers contain bundle-specific data (e.g., destination address), the same buffers are used for the same bundle until the bundle is closed by CPU software.

When closing a bundle, the CPU should check that all buffers have been returned, by following the linked list from the head to the last buffer. The buffers of a closed bundle may be used for a different new bundle. The linked list operation is depicted below.

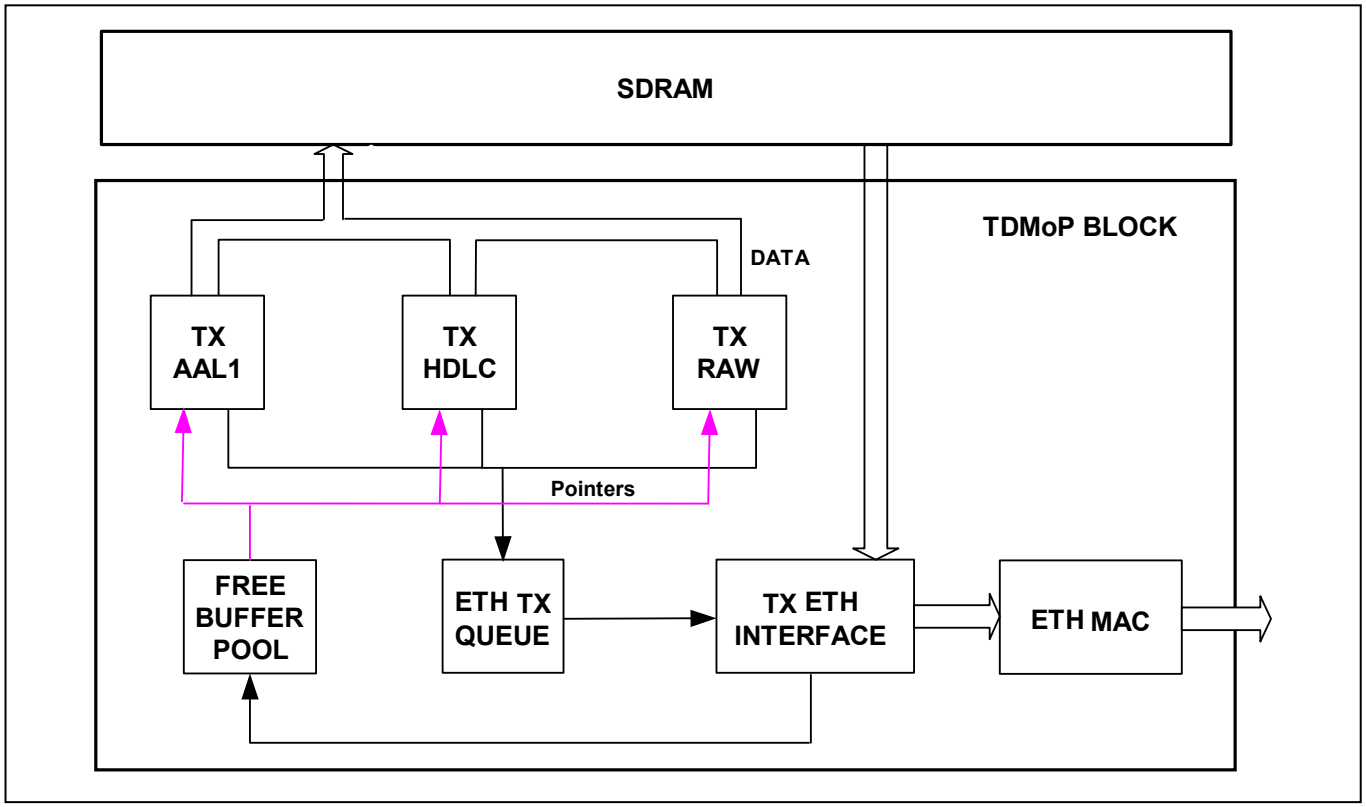
*Figure 10-50. Free Buffer Pool Operation*



#### 10.6.11.8 TDM to Ethernet Flow

Each payload-type machine receives the data of specific bundle timeslots and maps it into packets. To store a new packet in preparation, the machine extracts a pointer from the free buffer pool (section 10.6.11.7) and fills the associated buffer with TDM timeslot data, one by one. When a packet is completed in a buffer, the payload-type machine places the buffer pointer in the Ethernet Tx queue. The Tx Ethernet interface polls the queue, extracts the pointer, and transfers the packets from the buffer to the Ethernet MAC block, to be sent over the Ethernet network. Then, it returns the pointer to the free buffer pool. The buffer can then be used again by the payload-type machine to store subsequent TDM data for the bundle.

*Figure 10-51. TDM-to-Ethernet Flow*

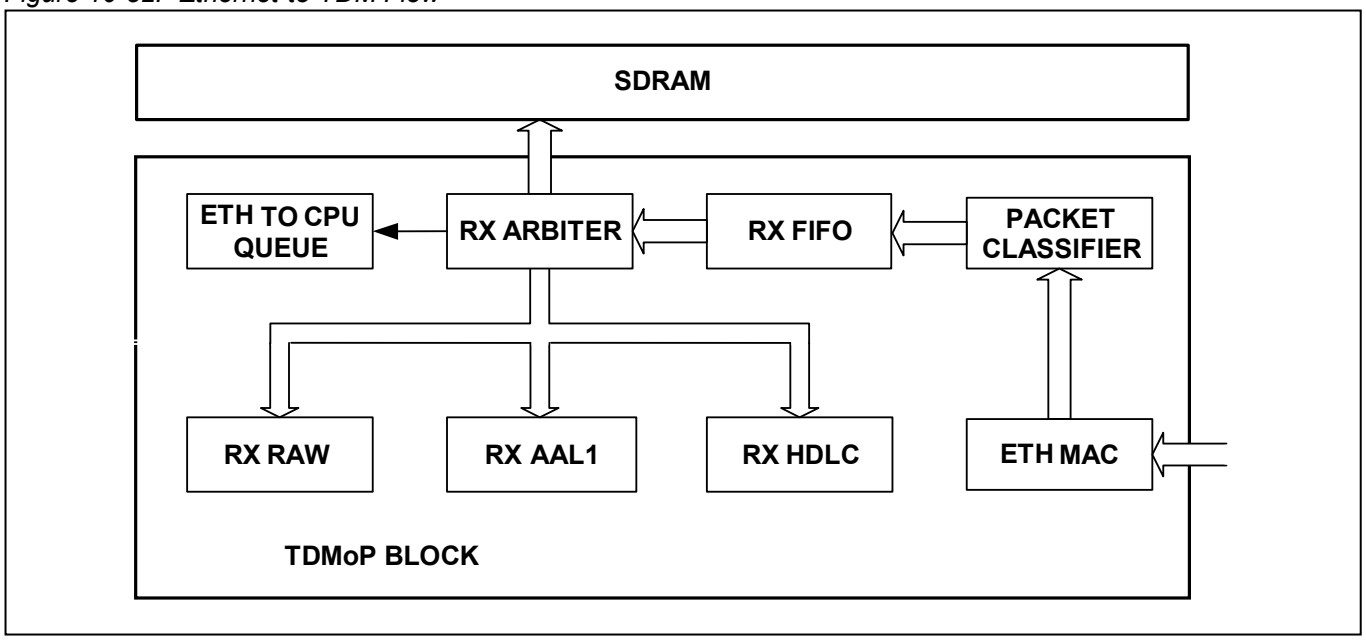


#### 10.6.11.9 Ethernet to TDM Flow

A packet arriving from the Ethernet port passes through the Ethernet MAC block. The MAC block does not store the packet, but it does calculate the CRC to verify packet data integrity. If the packet is bad, the MAC signals this to the packet classifier on the last word of the packet, and the packet classifier discards it.

The packet classifier examines the packet header and decides to either discard the packet or transfer it into the chip based on the settings of the packet classifier configuration registers (see Table 11-4). The packet classifier tags the buffer descriptor for one of the following destinations: ETH-to-CPU queue or payload-type machines. The packet classifier stores the packet payload preceded by the buffer descriptor in the Rx FIFO and notifies the Rx arbiter. The Rx arbiter then passes it to one of the payload-type machines. The payload-type machine extracts the TDM data and inserts it into the jitter buffer in the SDRAM. From there, the data is transmitted serially out the TDM port.

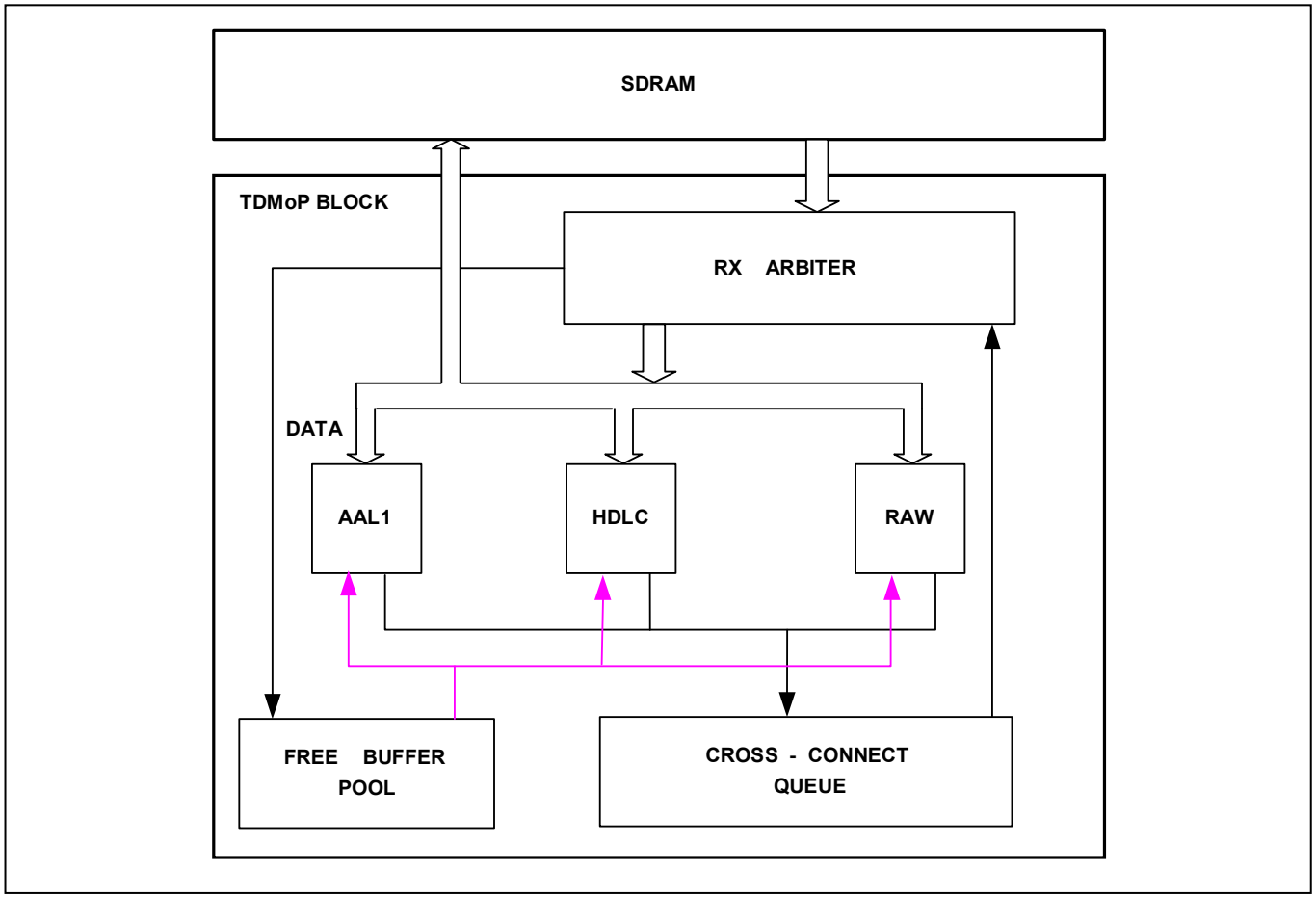
*Figure 10-52. Ethernet-to-TDM Flow*



#### 10.6.11.10 TDM to TDM (Cross-Connect) Flow

Each payload-type machine receives the data of bundle-specific TDM timeslots and maps the data into Ethernet packets. To store a packet, the payload-type machine needs an SDRAM buffer which it gets by extracting a buffer pointer from the free buffer pool. It then fills the buffer as it processes the TDM timeslots. When a packet is completed in a buffer, the machine places the buffer pointer in the cross-connect queue. The RX arbiter polls the cross-connect queue, extracts the pointer, transfers the buffer data to the appropriate payload-type machine, and then returns the pointer to the free buffer pool. The payload-type machine then extracts the TDM data and inserts it into the jitter buffer in the SDRAM. From there, the data is transmitted serially out the TDM port.

*Figure 10-53. TDM-to-TDM Flow*

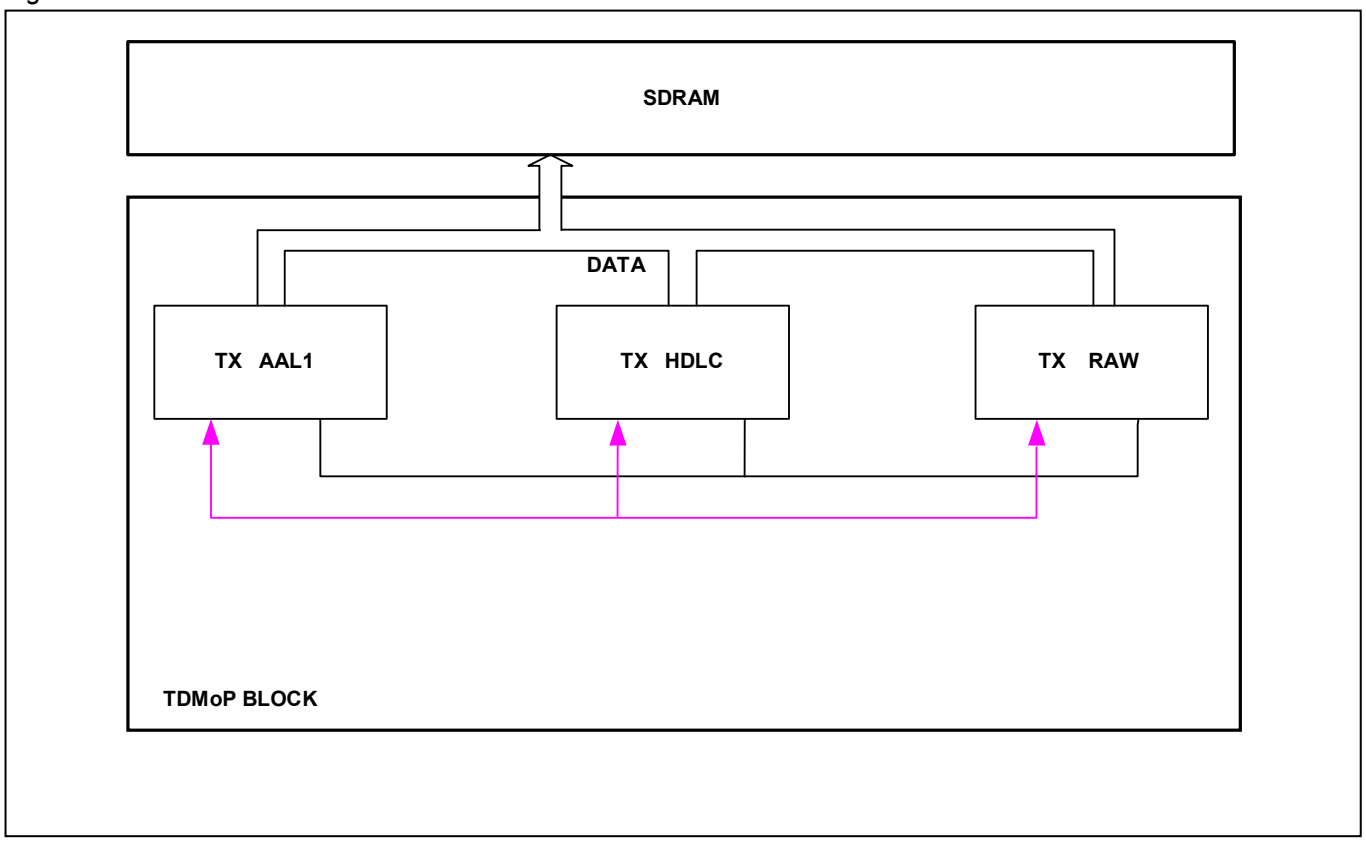


#### 10.6.11.11 TDM to CPU Flow

The payload-type machines identify the destination of their packets according to the per-bundle configuration. Upon getting the first byte of a packet in a bundle destined to the CPU, the machine needs a buffer to store the packet. It therefore checks whether a buffer is available in the TDM-to-CPU pool. If the pool is empty, the machine discards the current data. If a buffer is available, the machine stores the packet payload in the buffer and then adds the buffer pointer to the TDM-to-CPU queue. The CPU polls this queue to look for packets that need to be processed, gets the buffer pointer, and reads the packet from the SDRAM. After processing the packet, the CPU closes the loop by returning the pointer to the TDM-to-CPU pool.

The TDM-to-CPU pool and queue can contain up to 128 pointers each. Section 11.4.6 describes the pool and queue registers.

*Figure 10-54. TDM-to-CPU Flow*

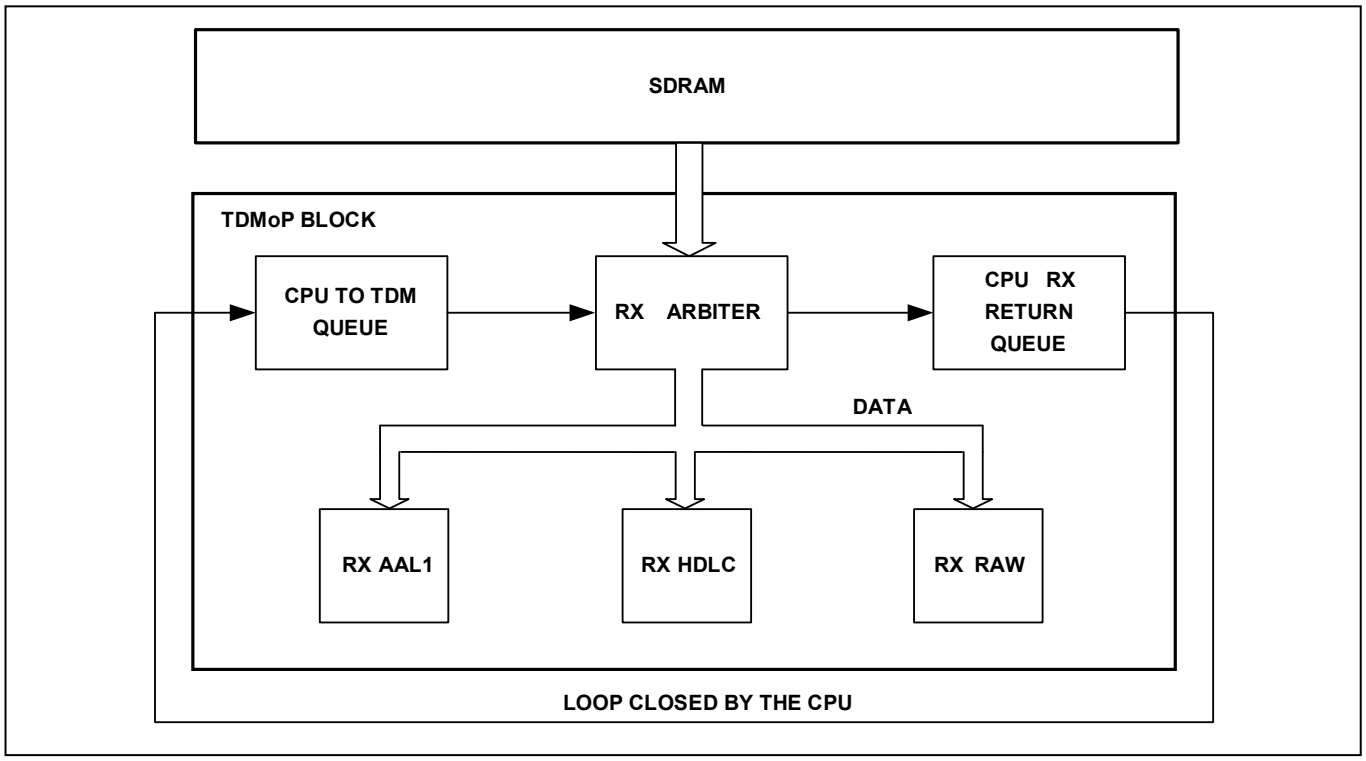


#### 10.6.11.12 CPU to TDM Flow

The Rx arbiter polls the CPU-to-TDM queue for new packets waiting in the SDRAM to be processed. If the queue level is greater than zero and there are no buffers pending in the Rx FIFO or the cross-connect queue, the Rx arbiter extracts the pointer and copies the relevant data from the SDRAM buffer to the appropriate payload-type machine. The arbiter then checks whether the CPU Rx return queue is not full to return the pointer. If the return queue is full, the arbiter keeps the pointer and does not poll the CPU-to-TDM queue until it succeeds in returning the pointer. After returning the pointer to the CPU Rx return queue for reuse, the arbiter is ready to take another pointer from the CPU-to-TDM queue.

The CPU-to-TDM queue and the CPU Rx return queue can contain up to 32 pointers each. Section 11.4.6 describes the pool and queue registers.

*Figure 10-55. CPU-to-TDM Flow*

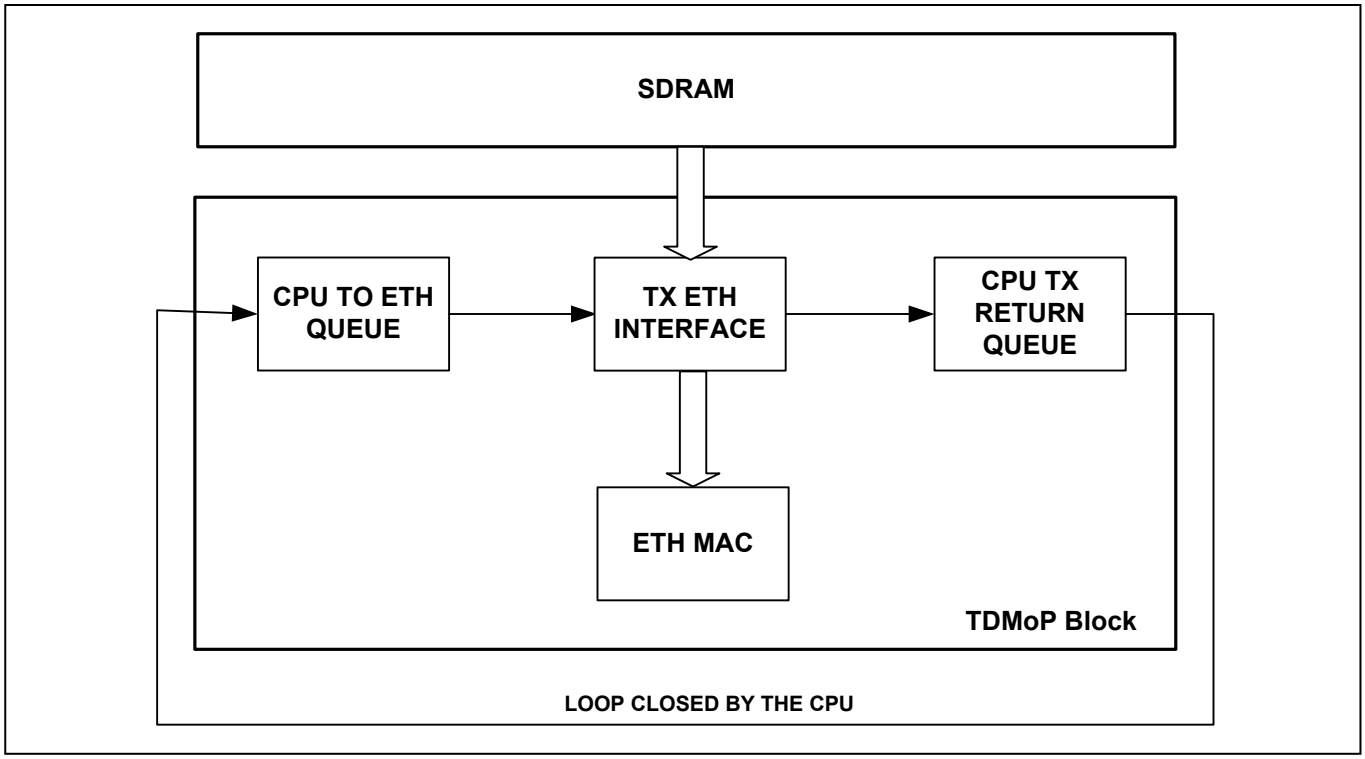


#### 10.6.11.13 CPU to Ethernet Flow

The Tx Ethernet interface polls the CPU-to-Ethernet queue for new packets waiting in the SDRAM to be processed. If the queue level is greater than zero and no buffers from the payload-type machines are waiting in the Ethernet Tx queue, the Tx Ethernet interface extracts the pointer and copies the relevant data from the SDRAM buffer to the Ethernet MAC block. It then checks whether the CPU TX return queue is not full to return the pointer. If the return queue is full, it keeps the pointer and does not poll the CPU-to-ETH queue until it succeeds in returning the pointer. After returning the pointer to the CPU TX return queue for reuse, the Tx Ethernet interface is ready to take another pointer from the CPU-to-ETH queue.

The CPU-to-Ethernet queue and the CPU Tx return queue can contain up to 32 pointers each. Section 11.4.6 describes the pool and queue registers.

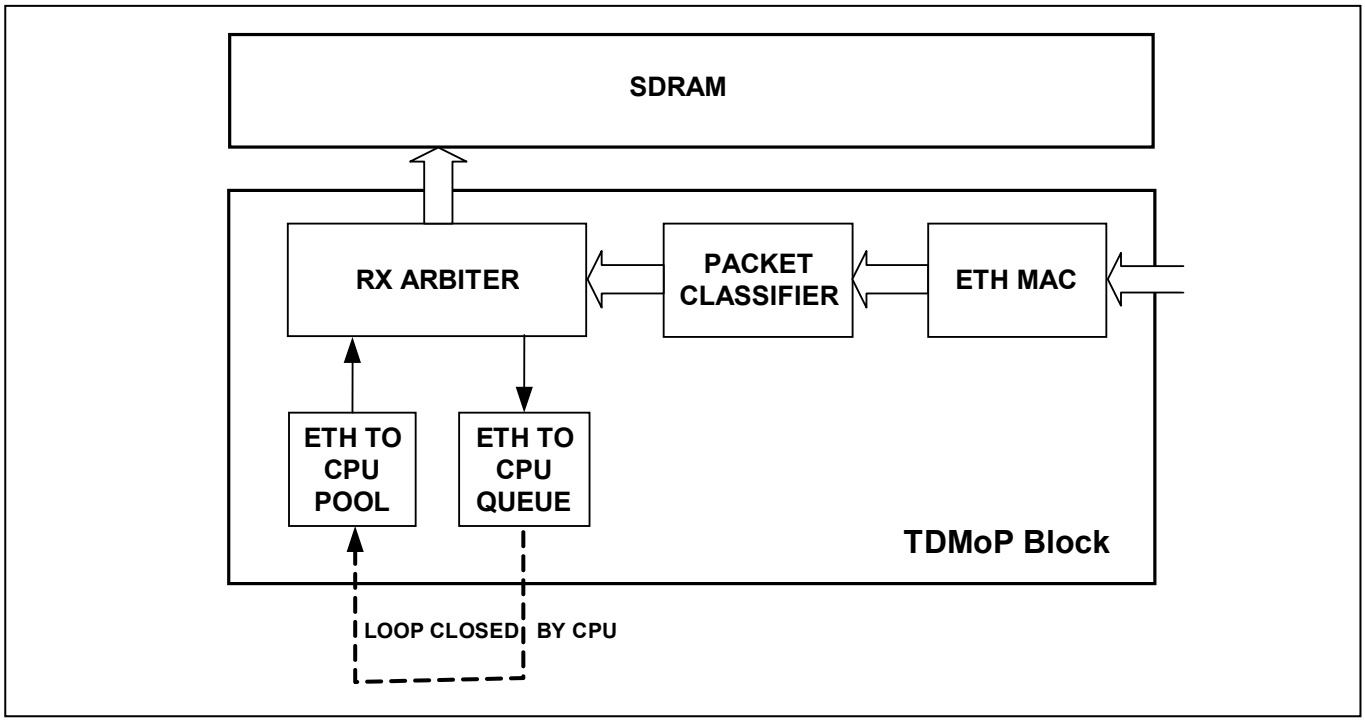
*Figure 10-56. CPU-to-Ethernet Flow*



#### 10.6.11.14 Ethernet to CPU Flow

Ethernet packets enter the chip via the Ethernet MAC block and the packet classifier into the Rx arbiter. When the Rx arbiter identifies that a packet is destined to the CPU, it extracts a pointer from the Ethernet-to-CPU pool (if the pool is empty, the Rx arbiter discards the packet) and stores the packet data into the SDRAM in the buffer indicated by the pointer. Then, it sends the pointer to the Ethernet-to-CPU queue (processed by the CPU). If the queue is full, the Rx arbiter keeps the pointer for itself for future use. The Ethernet-to-CPU queue and pool contain up to 128 pointers each. Section 11.4.6 describes the pool and queue registers.

*Figure 10-57. Ethernet-to-CPU Flow*




### 10.6.12 Ethernet MAC

#### 10.6.12.1 Introduction

The Ethernet MAC can operate at 10 or 100 Mbps. It supports MII, RMII (Reduced pin-count MII), and SSMII (source-synchronous serial MII). The MAC interface to the physical layer must be configured by the CPU.

The UNH-tested Ethernet MAC complies with IEEE 802.3. Its counters enable the software to generate network management statistics compatible with IEEE 802.3 Clause 5.

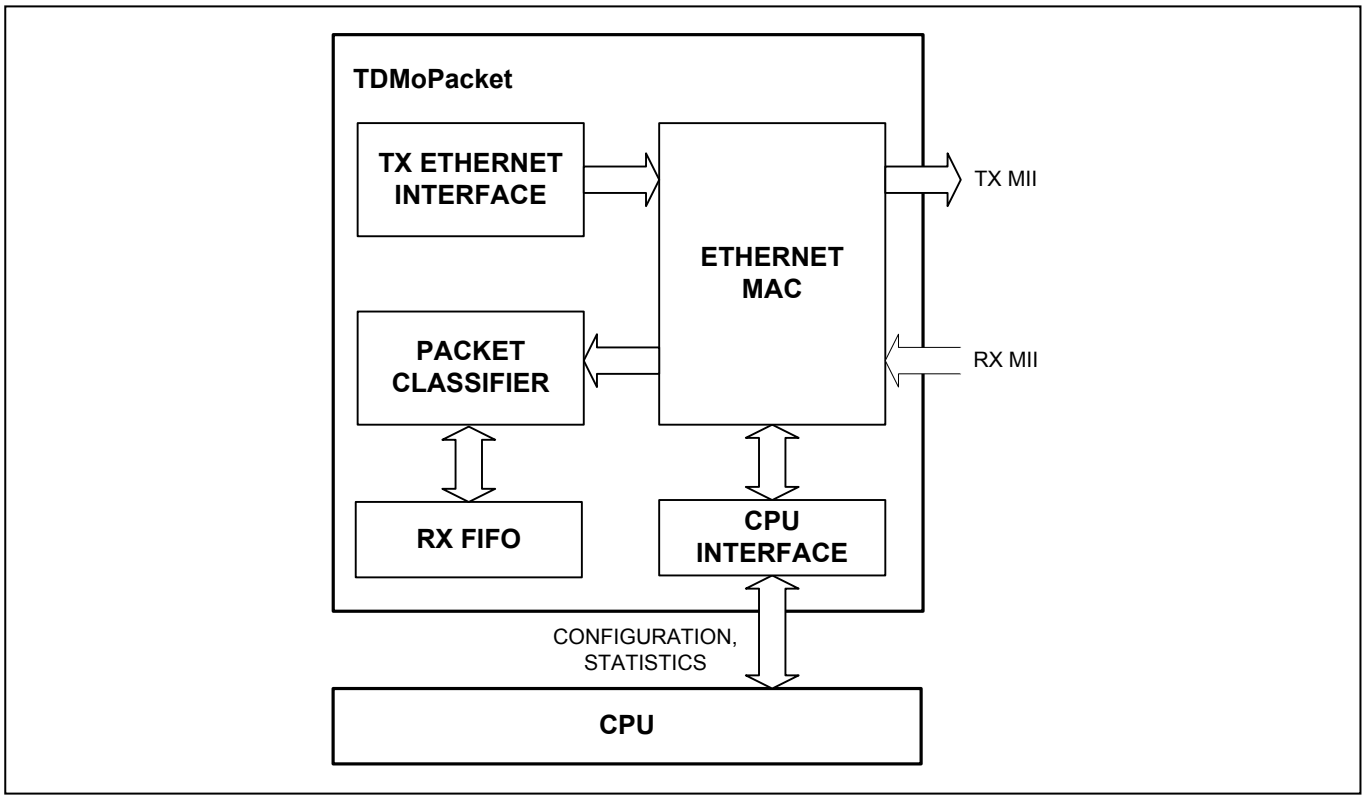
The Ethernet MAC supports physical layer management through an MDIO interface. The control registers drive the MDIO interface and select modes of operation, such as full or half duplex. Half-duplex flow control is achieved by forcing collisions on incoming packets. Full-duplex flow control supports recognition of incoming pause packets.

In the receive path, the MAC checks the incoming packets for valid preamble, FCS, alignment and length, and presents received packets to the packet classifier. Although packets with physical errors are discarded by default, the MAC can be configured to ignore errors and keep such packets.

In the transmit path, the MAC takes data from the Tx Ethernet interface, adds preamble and, if necessary, pad and FCS, then transmits data according to the CSMA/CD (carrier sense multiple access with collision detect) protocol.

In half-duplex mode the start of transmission is deferred if MII_CRS (carrier sense) is active. If MII_COL (collision) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. MII_CRS and MII_COL have no effect in full-duplex mode.

*Figure 10-58. Ethernet MAC*



#### 10.6.12.2 Pause Packet Support

Ethernet transmission pause in response to a received pause packet is enabled when Pause_enable=1 in the MAC_network_configuration register.

When a valid pause packet is received, the MAC_pause_time register is updated with the packet's pause time regardless of its current contents and regardless of the state of Pause_enable bit. In addition, the Pause_packet_ Rxd interrupt in the MAC_interrupt_status is triggered if it is enabled in the MAC_interrupt_mask register.

If Pause_enable=1 and the value of the MAC_pause_time register is non-zero, no new packet is transmitted.

A valid pause packet is defined as having a destination address that matches 0x0180C2000001, an Ethertype of 0x8808, and the pause opcode of 0x0001 as shown in Table 10-28.

**Table 10-28. Start of an 802.3 Pause Packet**

| Destination Address | Source Address | Ethertype (MAC Control Frame) | Pause opcode | Pause Time |
|---|---|---|---|---|
| 0x0180C2000001 | 6 bytes | 0x8808 | 0x0001 | 2 bytes |

Pause packets that have FCS or other errors are treated as invalid and discarded. Valid received pause packets increment the Pause_packets_Rxd_OK counter.

The MAC_pause_time register decrements every 512 bit times after transmission has stopped. For test purposes, the register decrements every MII receive clock cycle instead if Retry_test=1 in the MAC_network_configuration register. If the Pause_enable bit is not set, the decrementing happens regardless of whether transmission has stopped or not.

The Pause_time_zero interrupt in the MAC_interrupt_status register is asserted whenever the MAC_pause_time register decrements to zero (assuming it is enabled in the MAC_interrupt_mask).

Automatic transmission of pause packets is supported through the transmit pause packet bits of the MAC_network_control register. If either Transmit_pause_packet or Transmit_zero_quantum_pause_ packet is set, a pause packet is transmitted only if Full_duplex=1 in the MAC_network_configuration register and Transmit_enable=1 in the MAC_network_control register. Pause packet transmission takes place immediately if transmit is inactive or if transmit is active between the current packet and the next packet due to be transmitted. The transmitted pause packet comprises the items in the following list:

- Destination address of 01-80-C2-00-00-01
- Source address taken from the MAC_specific_address registers
- Ethertype of 0x8808 (MAC control frame)
- Pause opcode of 0x0001
- Pause quantum
- Fill of 0x00 to take the frame to minimum frame length
- Valid FCS.

The pause quantum used in the generated packet depends on the trigger source for the packet as follows:

- If Transmit_pause_packet=1, the pause quantum comes from the MAC_transmit_paulse_quantum register. The Transmit Pause Quantum register resets to a value of 0xFFFF giving a maximum pause quantum as a default.
- If Transmit_zero_quantum_pause_ packet=1, the pause quantum is zero.

After transmission, no interrupts are generated and the only counter incremented is the Transmitted_pause_packets.

Pause packets can also be transmitted by the MAC using normal packet transmission methods. It is possible to transmit a pause packet while the transmitter is paused by resetting the Pause_enable bit.

### 10.6.13 Packet Classifier

The Packet Classifier is part of the receive path, immediately following the Ethernet MAC block. It analyzes the header of each incoming packet, by comparing the header fields to the chip's configured parameters, and then decides whether to discard the packet or add a buffer descriptor and forward the packet to the CPU or one of the payload-type machines. Section 11.4.1 has register descriptions for the packet classifier configuration registers.

**IP version:**

- Packets with IP version different than 4 or 6 are always discarded.
- The chip has three IPv4 addresses and two IPv6 addresses (all software configurable)
- The chip works in one of four modes defined by two bits in General_cfg_reg1, as described in Table 10-29.

**Table 10-29. Handling IPv4 and IPv6 Packets**

| IP_version | Dual_stack | Transmitted Packets IP Version | Received Packets IP Version |
|---|---|---|---|
| 0 | 0 | IPv4 | Receive only IPv4 packets (other IP versions are discarded) |
| 1 | 0 | IPv6 | Receive only IPv6 packets (other IP versions are discarded) |
| 0 | 1 | IPv4 | Receive both IPv4 and IPv6 packets (dual stack mode) |
| 1 | 1 | IPv6 | Receive both IPv4 and IPv6 packets (dual stack mode) |

Although the chip has more than one IP address, in most cases all three IPv4 addresses should have the same value and both IPv6 addresses should have the same value. The chip also has two configurable MAC addresses.

Packets with CRC errors are discarded regardless to their contents, unless the Ethernet MAC has been configured to ignore them (in which case they are treated as correct packets).

IP Packets with IP checksum error are discarded, unless the Discard_ip_checksum_err configuration bit is cleared in General_cfg_reg0.

Packets other than TDM-over-IP or TDM-over-MPLS or TDM-over-MEF packets destined to the chip are not transferred to the payload-type machines. Instead, they are either discarded or transferred to the CPU according to the nine Discard_switch configuration bits in Packet_classifier_cfg_reg3:

Discard_Switch_0: An ARP packet whose Ipv4 destination address is not identical to any of the chip's Ipv4 addresses is discarded if Discard_Switch_0 is set. Otherwise it is transferred to the CPU.

Discard_Switch_1: An IP (both Ipv4 or Ipv6) packet whose IP destination address is not identical to any of the chip's IP addresses is discarded if Discard_Switch_1 is set. Otherwise it is transferred to the CPU.

Discard_Switch_2: A packet whose Ethertype is not known by the block is discarded if Discard_Switch_2 is set. Otherwise it is transferred to the CPU.

Discard_Switch_3: An ARP packet whose Ipv4 destination address is identical to one of the chip's Ipv4 addresses is discarded if Discard_Switch_3 is set. Otherwise it is transferred to the CPU.

Discard_Switch_4: An IP packet destined to the chip whose protocol is different than UDP and L2TPv3 is discarded if Discard_Switch_4 is set. Otherwise it is transferred to the CPU.

Discard_Switch_5: An IP/UDP packet destined to the chip whose UDP destination/source port number is not identical to one of the chip's TDM-over-Packet port numbers (according to TDMoIP_port_num_loc in Packet_classifier_cfg_reg3) is discarded if Discard_Switch_5 is set. Otherwise it is transferred to the CPU.

Discard_Switch_6: A TDMoP/MPLS/MEF packet destined to the chip whose bundle identifier is not identical to one of the chip's OAM Bundle Numbers or one of the bundle identifiers

assigned to the chip's internal bundles, is discarded if Discard_Switch_6 is set. Otherwise it is transferred to the CPU.

Discard_Switch_7: A packet recognized as OAM packet (see section 10.6.13.3) is discarded if Discard_Switch_7 is set. Otherwise it is transferred to the CPU.

Discard_Switch_8: A packet with Ethertype equal to CPU_dest_ether_type configuration is discarded when Discard_Switch_8 is set. Otherwise it is transferred to the CPU.

A packet is identified as a TDM-over-Packet packet destined to the chip if it meets the following conditions:

- It is unicast with its destination address identical to the chip's MAC addresses, multicast or broadcast
- It has either no VLAN tags, one VLAN tag or two VLAN tags (supports VLAN stacking). See section 10.6.13.4.
- Its protocol is UDP/IP or L2TPv3
- Its IP address is identical to one of the IP addresses of the chip
- Its UDP destination port number is identical to one of the chip's TDM-over-Packet port numbers (optional). See section 10.6.13.1.
- Its bundle identifier is identical to one of the bundle identifiers assigned to the chip's internal bundles or the packet is identified as an OAM packet. See section 10.6.13.2.

A packet is identified as a TDMoMPLS or TDMoMEF packet destined to the chip if it meets the following conditions:

- It is unicast with its destination address identical to the chip's MAC addresses, multicast or broadcast
- It has either no VLAN tags, one VLAN tag or two VLAN tags (VLAN stacking)
- Its Ethertype is MPLS unicast, MPLS multicast, or MEF (see section 10.6.13.5)
- The bundle identifier located at the inner label is identical to one of the bundle identifiers assigned to the chip's internal bundles or the packet is identified as an OAM packet.

The structure of packets identified as TDM-over-Packet packets destined to a specific bundle of the chip or as OAM packets destined to the chip is shown below.

*Figure 10-59. Format of TDMoIP Packet with VLAN Tag*

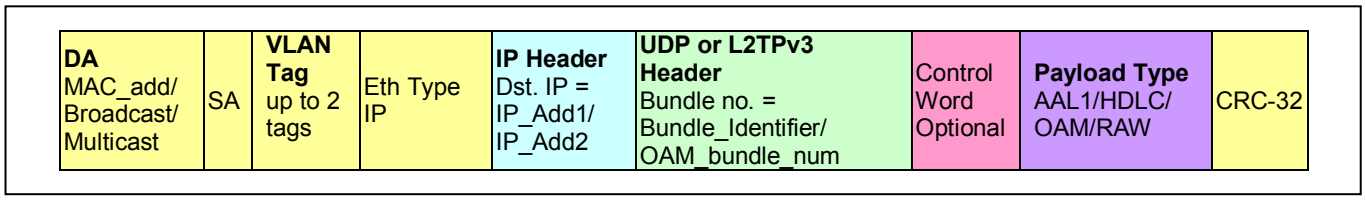



*Figure 10-60. Format of TDMoMPLS Packet with VLAN Tag*

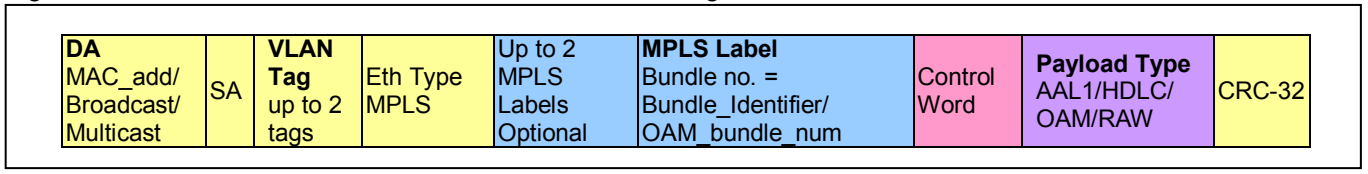
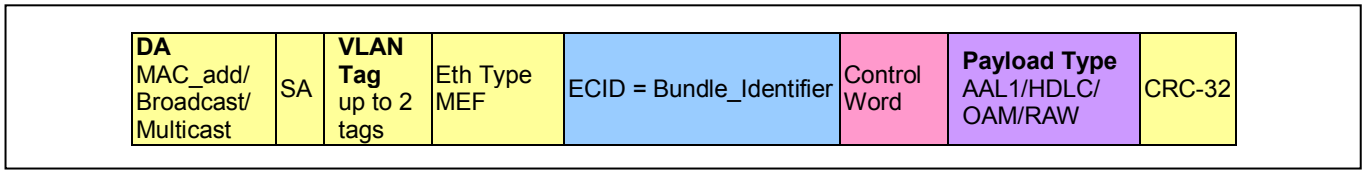



*Figure 10-61. Format of TDMoMEF Packet with VLAN Tag*




Packets that pass the classification process are temporarily stored in the Rx FIFO. This FIFO is used to buffer momentary bursts from the network if the internal hardware is busy. The Rx arbiter transfers the packets from the Rx FIFO to the payload-types machines or to external SDRAM.

#### 10.6.13.1 TDMoIP Port Number

The TDMoIP_port_num1 and TDMoIP_port_num2 configuration fields are used by the block to identify UDP/IP TDMoIP packets. Although the chip has two of these fields, in most cases both fields should have the default value (0x085E) as assigned by IANA for TDM-over-Packet. The UDP source

Both values are compared against the UDP_SRC_PORT_NUM or the UDP_DST_PORT_NUM of incoming packets as specified by the TDMoIP_port_num_loc field in Packet_classifier_cfg_reg3 (see Table 10-30).

**Table 10-30. TDMoIP Port Number Comparison for TDMoIP Packet Classification**

| TDMoIP_port_num_loc Value | Comparison |
|---|---|
| 00 | TDMoIP_port_num1/2 are ignored (no checking is performed) |
| 01 | TDMoIP_port_num1/2 are compared to source UDP port # of incoming packets |
| 10 | TDMoIP_port_num1/2 are compared to destination UDP port # of incoming packets |
| 11 | Reserved |

#### 10.6.13.2 Bundle Identifier Location and Width

The block determines the packet bundle identifier and its width after determining the packet type.

**Table 10-31. Bundle Identifier Location and Width**

| Packet Type | Bundle Identifier Location | Bundle Identifier Width |
|---|---|---|
| MPLS | Inner label | 20 bits |
| MEF | Inner label | 20 bits |
| L2TPv3/IP | Session ID | 32 bits |
| UDP/IP | Source UDP port number or destination UDP port number, as specified by Ip_udp_bn_loc in Packet_classifier_cfg_reg3 | 1-16 bits as specified by Ip_udp_bn_mask_n in Packet_classifier_cfg_reg6. |

#### 10.6.13.3 OAM Packet Identification

The block identifies OAM packets according to *one* of the following criteria:

- UDP/IP-specific OAM packets: Match between the packet's bundle identifier and one of the values (up to 8 different) configured in the OAM_Identification registers.
- VCCV OAM packets: Match between the packet's control word bits 31:16 and a 1 to 16 bit value specified by the combination of VCCV_oam_mask_n and VCCV_oam_value fields in Packet_classifier_cfg_reg18. Such a match is taken into account only when OAM_ID_in_CW=1 in the Bundle Configuration Tables.
- MEF OAM packets: Match between packet Ethertype and Mef_oam_ether_type in register Packet_classifier_cfg_reg9.

#### 10.6.13.4 VLAN Tag Identification

A VLAN tag is identified according to one of the following criteria:

- Tag protocol identifier = 0x8100
- Tag protocol identifier = vlan_2nd_tag_identifier in Packet_classifier_cfg_reg7 (Created to support 0x9100 as a tag identifier)

#### 10.6.13.5 Known Ethertypes

The block considers the following Ethertypes as known Ethertypes:

- IPv4 (0x800)
- IPv6 (0x86DD)
- MPLS unicast (0x8847)
- MPLS multicast (0x8848)
- ARP (0x806)
- MEF Ethertype as configured in Mef_ether_type in Packet_classifier_cfg_reg9
- MEF OAM Ethertype as configured in Mef_oam_ether_type in Packet_classifier_cfg_reg9
- Specific Ethertype as configured in CPU_dest_ether_type in Packet_classifier_cfg_reg7

#### 10.6.13.6 Received OAM Time-Stamping

For any received packet forwarded to the CPU (ETH → CPU path) the third dword of the buffer descriptor holds the timestamp as latched by the block as the packet was received. This timestamp can be used by the CPU for network delays measurements. The timestamp is 1 μs or 100 μs as specified by the OAM_timestamp_resolution field in General_cfg_reg0.

#### 10.6.13.7 Neighbor Discovery (RFC 2461)

Where IPv4 has ARP, IPv6 has NDP, the neighbor discovery protocol. For the purposes of this discussion, NDP and ARP are very similar: one node sends out a request packet (called a *neighbor solicitation* in NDP), and the node it was looking for sends back a reply (*neighbor advertisement*) giving its link-layer address. NDP is part of ICMPv6, unlike ARP, which doesn't run over IP. NDP also uses multicast rather than broadcast packets.

For NDP (ICMPv6) packets to be forwarded to the CPU, Discard_switch_4 must be cleared.

#### 10.6.13.8 Packet Payload Length Sanity Check

The packet classifier performs a sanity check between the payload length of the received packet and the AAL1/SAToP/CESoPSN bundle's configuration. Discarding packets that fail the sanity check can be disabled per bundle by setting Rx_ discard_sanity_fail=1 in the Bundle Configuration Tables.

### 10.6.14 Packet Trailer Support

There are Ethernet switch chips that in some of their modes transmit packets with a trailer and expect the incoming packets to have a trailer. A trailer is an addition of several bytes at the end of the packet that helps the switch to decide about the incoming packet destination and to tag out-going packets.

When the device operates opposite such a switch, the trailer is supported in the following manner:

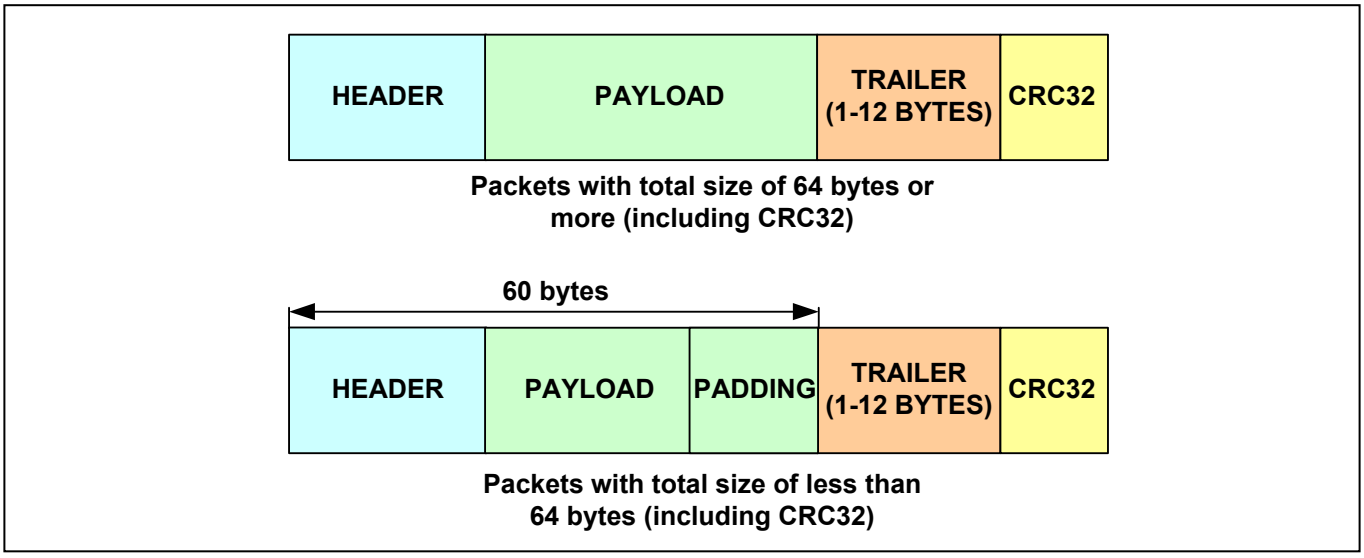
- Transmitted packets: A 1 to 12 byte trailer is added to all transmitted packets. The trailer contents that are stored in the packet buffer (immediately after the buffer descriptor starting from offset 0x8) may be varied per packet.

- Received packets: The trailer content is ignored. It is removed from packets destined to the payload-type machines and not transferred with packets destined to CPU.

- Trailer size is set for all transmitted/received packets in the Packet_trailer_length field in General_cfg_reg0.

The structure of packets with trailer is illustrated in Figure 10-62.

*Figure 10-62. Structure of Packets with Trailer*




The CRC is calculated over all packet bytes including over the trailer bytes. The transmitted bytes counter and the received bytes counter (section 11.4.3.3) do not count the trailer bytes.

### 10.6.15 Counters and Status Registers

For information about counters and registers in the TDMoP block, see section 11.4.

### 10.6.16 Connection Level Redundancy

The TDMoP block provides optional connection level redundancy for AAL1, SAToP and CESoPSN bundles. In the TDM-to-Ethernet direction, on a bundle basis, each packet may be transmitted once with certain headers, or twice, each time with different headers. When transmitted twice, the packets have the same payload, same control word and same RTP header (if used) but may have different packet headers (including layer 2, 3 and 4 headers).

For example, the chip can duplicate a bundle's packets on transmission where the only difference between the duplicated packets is their bundle number or their VLAN ID.

On the receive side, when two redundant streams use different bundle numbers, the chip can be configured to receive only the packets with the first bundle number or the packets with the second bundle number.

To enable this feature, CPU software must initialize the transmit buffers of a bundle with both headers. The second header must be located at offset 0x782 from start of the buffer and its length (in bytes) is indicated by the buffer descriptor Hdr2_length field (not including the RTP header length neither the control word length). By changing the Protection_mode configuration field of the bundle, the user can choose (per bundle) whether to transmit each of the packets once with the first or the second header, or twice, each time with a different header.

On the receive side, only the packets with their bundle number configured in the Rx_bundle_identifier field of a specific bundle, are forwarded. The CPU may change this value dynamically, in order to switch to the redundant connection at any time.

On the receive side, when both streams use the same bundle number, switching from one stream to another is almost seamless. No software intervention is needed as the payload-type machine discards the duplicated packets. During this process the end-to-end delay may change because of different route delays and 1–2 packet of packet loss may occur.

The destination MAC/IP (and/or VLAN) of the duplicated packets can be different as the chip supports more than one MAC/IP address in the packet classifier.

### 10.6.17 OAM Signaling

TDMoP bundles require a signaling mechanism to provide feedback regarding problems in the communications environment. In addition, such signaling can be used to collect statistics related to the performance of the underlying PSN. The OAM procedures detailed below are ICMP-like.

#### 10.6.17.1 Connectivity Check Messages

In most conventional IP applications, a server sends some finite amount of information over the network after an explicit request from a client. With TDM-over-Packet, the source sends a continuous stream of packets towards the destination, without knowing whether the destination device is ready to accept them, leading to flooding of the PSN. The problem may occur when a TDM-over-Packet gateway fails or is disconnected from the PSN, or the bundle is broken. After an aging time, the destination gateway disappears from the routing tables, and intermediate routers may flood the network with the TDM-over-Packet traffic in an attempt to find a new path.

The solution to this problem is to significantly reduce the number of TDM-over-Packet packets transmitted per second when bundle failure is detected, and to return to full rate only when the bundle is restored. The detection of failure and restoration is made possible by the periodic exchange of one-way connectivity check messages. Connectivity is tested by periodically sending OAM messages from the source gateway to the destination gateway, and having the destination reply to each message.

The connectivity check mechanism can also be useful during setup and configuration. Without OAM signaling, one must ensure that the destination gateway is ready to receive packets before starting to send them. Since TDM-over-Packet gateways operate full duplex, both must be set up and properly configured simultaneously to avoid flooding. By using the connectivity mechanism, a configured gateway waits until it can detect its destination before transmitting at full rate. In addition, errors in configuration can be readily discovered by using the service-specific field.

#### 10.6.17.2 Performance Measurements

In addition to one-way connectivity, the OAM signaling mechanism can be used to request and report on various PSN metrics, such as one-way delay, round trip delay, packet delay variation, etc. It can also be used for remote diagnostics, and for unsolicited reporting of potential problems (e.g. dying gasp messages).

#### 10.6.17.3 Processing OAM Packets

In the Ethernet-to-CPU direction, the device identifies OAM packets as described in section 10.6.13.3.

In the CPU-to-Ethernet direction the chip timestamps packets when the Stamp field of the buffer descriptor field is set. The timestamp location in the packet is specified by the Ts_offset buffer descriptor field. When the CPU transmits an OAM packet, the buffer descriptor must identify the packet as a non-TDMoP/MPLS packet (i.e. is not assigned to any bundle), as other packet types are not time-stamped in any case.

## 10.7 Global Resources

See the top-level block diagram in Figure 6-1. Global resources in the device include CLAD1, CLAD2 and the CPU Interface block. These resources are configured in the global registers described in section 11.3. These registers also handle device identification, top-level mode configuration, I/O pin configuration, global resets, and top-level interrupts.

## 10.8 Per-Port Resources

See the top-level block diagram in Figure 6-1. Each port is independently configured in the Port[n]_cfg_reg register. In addition to E1 and T1 modes, a port can also be configured as a serial data port that can connect to a serial interface transceiver for V.35 or RS-530 support. This would usually be in a DCE application of some kind. The port can be configured for this mode by setting Port[n]_cfg_reg:Int_type=00.

The device also features one 10/100 Ethernet port that can be configured to have an MII, RMII or SSMII interface. The Ethernet port can work in half or full duplex mode and supports VLAN tagging and priority labeling according to 802.1p 802.1Q, including VLAN stacking. Section 11.4.16 describes the Ethernet port.

## 10.9 Device Interrupts

The H_INT pin indicates interrupt requests. The only source for interrupts in the DS34S10x devices is the TDMoP block (which includes the MAC). The TDMoPIM bit in GTIMR must be set to 1 enable interrupts from the TDMoP block. The Intpend register indicates the source(s) of interrupt(s) from the TDMoP block. If one of the Intpend bits is set, it can be cleared only by writing 1 to it. At reset, all Intpend interrupts are disabled due to the Intmask register default values. Writing 0 to an Intmask bit enables the corresponding Intpend interrupt.

The TDMoP interrupts indicated in the Intpend register are of two types. The first type consists of interrupts generated by a single source. The second type consists of interrupts that can originate from any of several possible interrupt sources including the ETH_MAC, CW_bits_change, Rx_CAS_change, Tx_CAS_Change, and JB_underrun interrupts.

The JBC_underrun interrupts can be masked per timeslot by setting the appropriate bits in the JBC_underrun_mask registers.

The Tx_CAS_change interrupts can be masked per timeslot by setting the appropriate bits in the Tx_CAS_change_mask registers.

The CW_bits_change interrupts can be masked per bundle by setting the appropriate bits in the CW_bits_mask registers. In addition, the fields of the control word that cause an interrupt when changed (L, R, M, FRG) can be configured in the CW_bits_change_mask register.

When an interrupt is indicated on H_INT, the CPU should read the Intpend register to identify the interrupt source and then proceed as follows:

| Interrupt Type | Interrupt Procedure |
|---|---|
| Single-source Interrupts | 1. Clear the pending interrupt(s) by writing 1 to the corresponding Intpend bit(s).<br>2. Service the source of the interrupt. |
| Rx_CAS_change | 1. Read the Rx_CAS_change bits in the Intpend register to determine which port(s) are indicating Rx CAS change.<br>2. Clear the set Rx_CAS_change bits in the Intpend register by writing 1 to them.<br>3. Read the corresponding Rx_CAS_change register(s) to determine which timeslot(s) have been changed.<br>4. Clear the set bits in the Rx_CAS_change register(s) by writing 1 to |

| Interrupt Type | Interrupt Procedure |
|---|---|
| | them.<br>5. Read the corresponding Rx CAS information from the Rx Line CAS registers (section 11.4.10). |
| Tx_CAS_change | 1. Read the Tx_CAS_change bits in the Intpend register to determine which port(s) are indicating Tx CAS change.<br>2. Clear the set Tx_CAS_change bits in the Intpend register by writing 1 to them.<br>3. Read the corresponding Tx_CAS_change register(s) to determine which timeslot(s) have been changed.<br>4. Clear the set bits in the Tx_CAS_change register(s) by writing 1 to them.<br>5. Read the appropriate Tx CAS information from neighboring framer IC(s). |
| CW_bits_change | 1. Clear the CW_bits_change bit in the Intpend register by writing 1 to it.<br>2. Read the CW_bits_change_low_bundles and CW_bits_change_high_bundles registers to determine which bundles(s) have control bits that have changed.<br>3. Clear the set bits in the CW_bits_change_low_bundles and CW_bits_change_high_bundles registers by writing 1 to them.<br>4. Read the state of the control word fields from the Packet Classifier Status register in the per-bundle status tables (section 11.4.4.1). |
| JB_underrun_Pn | 1. Read the JBC_underrun bits in the Intpend register to determine which port(s) are indicating jitter buffer underrun.<br>2. Clear the set JBC_underrun bits in the Intpend register by writing 1 to them.<br>3. Read the corresponding JBC_underrun register(s) to determine which buffers had underruns.<br>4. Clear the set bits in the JBC_underrun register(s) by writing 1 to them.<br>5. Service the underrun(s) as needed. |
| ETH_MAC | 1. Clear the ETH_MAC bit in the Intpend register by writing 1 to it.<br>2. Read the MAC_interrupt_status register to determine the source(s) of interrupts in the MAC (all bits are reset to 0 upon read).<br>3. Service the source(s) of the interrupt(s). |

If a bit in the Intpend register is set and that interrupt is then masked, the device generates an interrupt immediately after the CPU clears the corresponding mask bit. To avoid this behavior, the CPU should clear the interrupt from the Intpend register before clearing the mask bit.

# 11. Device Registers

## 11.1 Addressing

Device registers and memory can be accessed either 2 or 4 bytes at a time, as specified by configuration pin DAT_32_16_N. In the 16-bit addressing mode, addresses are multiples of 2, while in 32-bit addressing, addresses are multiples of 4.

The prefix “0x” indicates hexadecimal (base 16) numbering, as does the suffix “h” (Example: 2FFh). Addresses are always indicated in hexadecimal format.

The byte order for both addressing modes is “big-endian” meaning the most significant byte has the lowest address. See byte order numbers in grey in Figure 11-1 and Figure 11-2.

*Figure 11-1. 16-Bit Addressing*

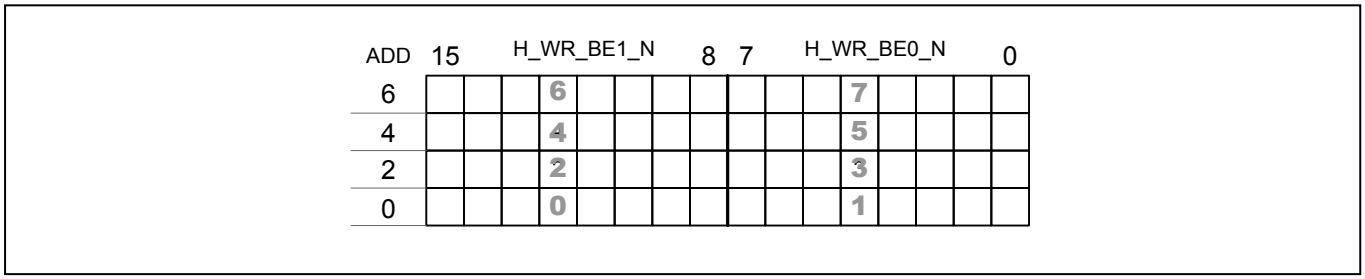
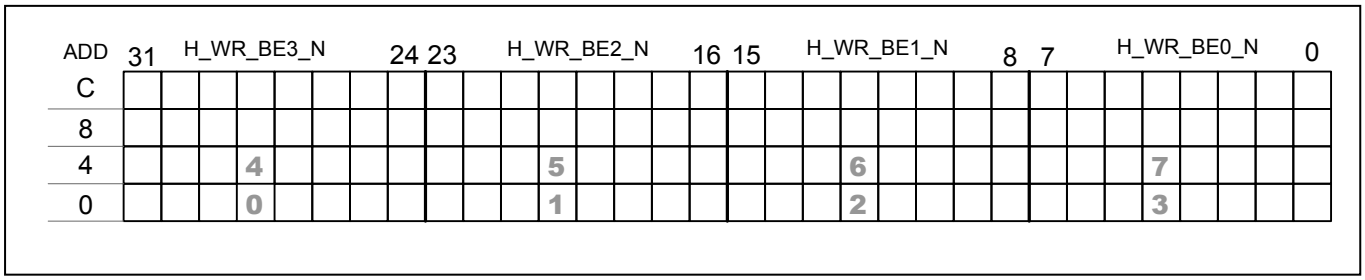
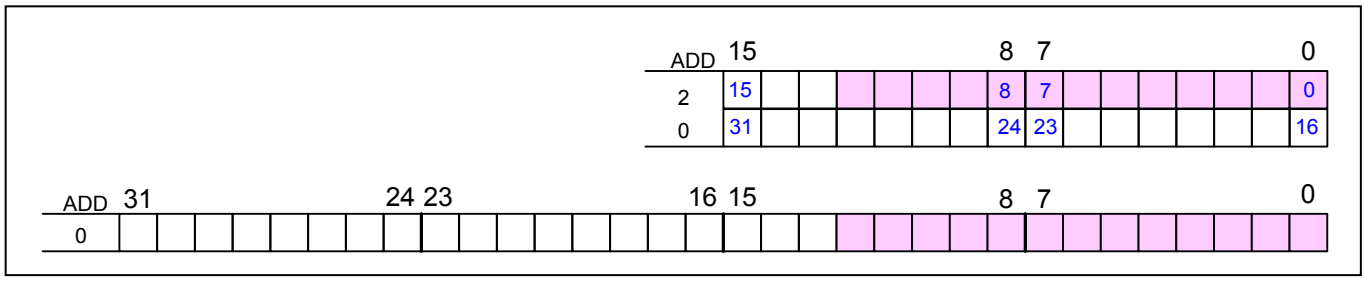



*Figure 11-2. 32-Bit Addressing*



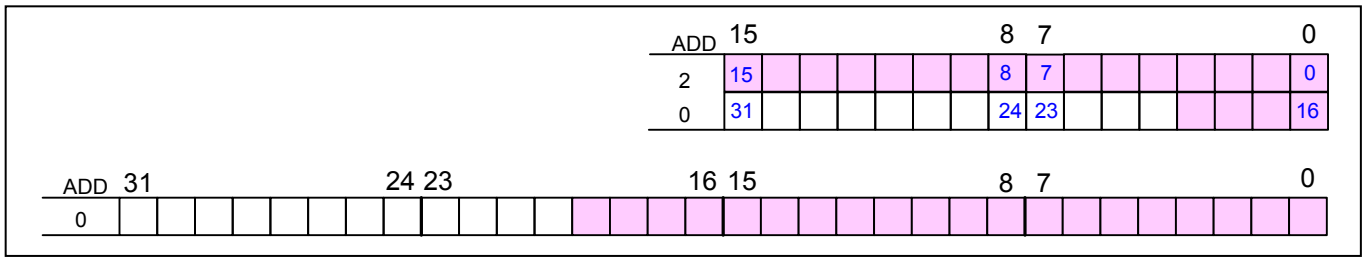

Partial data elements (shorter than 16 or 32 bits) are always positioned from LSb to MSb with the rest of the bits left unused. Thus, the bit numbers of data elements shorter than 16 bits are identical for both addressing modes (see bits [12:0] in Figure 11-3) and the CPU can access all bits by a single read/write.

*Figure 11-3. Partial Data Elements (shorter than 16 bits)*




Data elements 17 to 32 bits long need one read/write access in 32-bit addressing and two in 16-bit addressing. In Figure 11-4, the 20-bit data element needs one 32-bit CPU access (bits [19:0]) and two 16-bit accesses (bits [15:0] and then [3:0]).

*Figure 11-4. Partial Data Elements (16 to 32 bits long)*




SPI interface mode (H_CPU_SPI_N=0) always uses 32-bit addressing. See section 10.3.

## 11.2 Top-Level Memory Map

**Table 11-1. Top-Level Memory Map**

| Address Range | Contents | Page |
|---|---|---|
| 0 – 7F,FFF | TDM-over-Packet Registers | 93 |
| 80,000 – 107,FFF | Reserved | --- |
| 108,000 – 108,FFF | Global Registers | 91 |
| 109,000 – FFF,FFF | Reserved | --- |
| 1,000,000 – 1,FFF,FFF | External SDRAM | --- |

## 11.3 Global Registers

Functions contained in the global registers include device ID, CLAD configuration and top-level interrupt masking. The global register base address is 0x108,000.

**Table 11-2. Global Registers**

| Addr Offset | Register Name | R/W | Description | Page |
|---|---|---|---|---|
| 0x00 | GCR1 | R/W | Global Control Register 1 | 91 |
| 08 | GTRR | R/W | Global Transceiver Reset Register | 91 |
| 0C | IDR | RO | Identification Device Register | 92 |
| 10 | GTISR | RO | Global Transceiver Interrupt Status Register | 92 |
| 14 | GTIMR | R/W | Global Transceiver Interrupt Mask Register | 92 |

**GCR1 (Global Control Register) 0x00**

| Bits | Data Element Name | R/W | Default | Description |
|---|---|---|---|---|
| [31:15] | Not Used | - | 0 | Must be set to zero. |
| [14] | SYSCLKS | R/W | 0 | TDMoP System Clock Frequency Select<br>When a 25MHz clock is applied to the CLK_SYS pin (i.e. when the CLK_SYS_S pin is high), this bit configures the CLAD2 block to provide either a 50MHz clock or a 75MHz clock to the TDMoP block. When CLK_SYS_S=0 this bit is a don't care. See section 10.4.<br>0 = 50MHz<br>1 = 75MHz |
| [13:12] | FREQSEL | R/W | 00 | Frequency Select<br>Specifies the frequency of the signal applied to the CLK_HIGH pin.<br>00 = 38.88MHz (CLAD bypass; 38.88MHz in and out).<br>01 = 19.44MHz<br>10 = 10.000MHz<br>11 = 77.76MHz |
| [11:9] | Not Used | - | 0 | Must be set to zero. |
| [8] | CLK_HIGHD | R/W | 0 | CLK_HIGH Disable<br>Disables the 38.88MHz master clock to the clock recovery machines of the TDMoP block to save power. This bit should be set only when not using any of the TDMn_ACLK signals. See section 10.4.<br>0 = Enabled<br>1 = Disabled |
| [7:0] | Not Used | - | 0 | Must be set to zero. |

**GTRR (Global Transceiver Reset Register) 0x08**

| Bits | Data Element Name | R/W | Default | Description |
|---|---|---|---|---|
| [31:19] | Not Used | - | 0 | Must be set to zero. |
| [18] | TOPRST | R/W | 0 | TDMoP Core Software Reset<br>When set, this bit resets all of the TDMoP configuration registers to their default value.<br>0 = Normal operation<br>1 = Reset the TDMoP core |
| [17:0] | Not Used | - | 0 | Must be set to zero. |

### IDR (Identification Device Register) 0x0C

| Bits | Data Element Name | R/W | Default | Description |
|---|---|---|---|---|
| [31:16] | ID[31:16] | RO | 0 | These bits are always zero. |
| [15:4] | ID[15:4] | RO | See JTAG ID. | Device ID<br>These bits have the same information as the lower 12 bits of the Device ID portion of the JTAG ID register. See Table 12-2. |
| [3:0] | ID[3:0] | RO | See JTAG ID. | Device Revision<br>These bits have the same information as the four REV bits of the JTAG ID register. See Table 12-2. |

### GTISR (Global Transceiver Interrupt Status Register) 0x10

| Bits | Data Element Name | R/W | Default | Description |
|---|---|---|---|---|
| [31:25] | Not used. | - | 0 | Must be set to zero. |
| [24] | TDMoPIS | RO | 0 | TDM-over-Packet Interrupt Status<br>This status bit indicates when the TDM-over-Packet block is signaling an interrupt request. Interrupt mask is GTIMR.TDMoPIM.<br>0 = TDM-over-Packet has not issued an interrupt.<br>1 = TDM-over-Packet has issued an interrupt. |
| [23:0] | Not Used | - | 0 | Must be set to zero. |

### GTIMR (Global Transceiver Interrupt Mask Register) 0x14

| Bits | Data Element Name | R/W | Default | Description |
|---|---|---|---|---|
| [31:25] | Not used. | - | 0 | Must be set to zero. |
| [24] | TDMoPIM | R/W | 0 | TDM-over-Packet Interrupt Mask<br>This bit is the interrupt mask for GTISR.TDMoPIS.<br>0 = Interrupt masked.<br>1 = Interrupt enabled. |
| [23:0] | Not Used | - | 0 | Must be set to zero. |

## 11.4 TDM-over-Packet Registers

The base address for the TDMoP registers is **0x0**.

**Table 11-3. TDMoP Memory Map**

| Address Offset | Contents | Page |
|---|---|---|
| 0x0,000 | Configuration and Status Registers | 94 |
| 8,000 | Bundle Configuration Tables | 108 |
| 10,000 | Counters | 117 |
| 12,000 | Status Tables | 120 |
| 18,000 | Timeslot Assignment Tables | 120 |
| 20,000 | CPU Queues | 122 |
| 28,000 | Transmit Buffers Pool | 124 |
| 30,000 | Jitter Buffer Control | 130 |
| 38,000 | Transmit Software CAS | 134 |
| 40,000 | Receive Line CAS | 136 |
| 48,000 | Clock Recovery | 137 |
| 50,000 | Receive SW Conditioning Octet Select | 138 |
| 58,000 | Receive SW CAS | 139 |
| 68,000 | Interrupt Controller | 140 |
| 70,000 | Packet Classifier | 147 |
| 72,000 | Ethernet MAC | 148 |

### 11.4.1 Configuration and Status Registers

The base address for the TDMoP configuration and status registers is **0x0,000**.

**Table 11-4. TDMoP Configuration Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x00 | General_cfg_reg0 | General configuration register0 | 95 |
| 04 | General_cfg_reg1 | General configuration register1 | 96 |
| 08 | General_cfg_reg2 | General configuration register2 | 97 |
| 0C | Port1_cfg_reg | Port 1 configuration register | 97 |
| 10 | Port2_cfg_reg | Port 2 configuration register | 97 |
| 14 | Port3_cfg_reg | Port 3 configuration register | 97 |
| 18 | Port4_cfg_reg | Port 4 configuration register | 97 |
| 1C | Port5_cfg_reg | Port 5 configuration register | 97 |
| 20 | Port6_cfg_reg | Port 6 configuration register | 97 |
| 24 | Port7_cfg_reg | Port 7 configuration register | 97 |
| 28 | Port8_cfg_reg | Port 8 configuration register | 97 |
| 2C | Rst_reg | Reset register | 100 |
| 30 | TDM_cond_data_reg | TDM AAL1/SAToP conditioning data register | 101 |
| 34 | ETH_cond_data_reg | Ethernet AAL1/SAToP conditioning data register | 101 |
| 38 | Packet_classifier_cfg_reg0 | Packet classifier configuration register0 | 101 |
| 3C | Packet_classifier_cfg_reg1 | Packet classifier configuration register1 | 101 |
| 40 | Packet_classifier_cfg_reg2 | Packet classifier configuration register2 | 101 |
| 44 | Packet_classifier_cfg_reg3 | Packet classifier configuration register3 | 102 |
| 48 | Packet_classifier_cfg_reg4 | Packet classifier configuration register4 | 103 |
| 4C | Packet_classifier_cfg_reg5 | Packet classifier configuration register5 | 103 |
| 50 | Packet_classifier_cfg_reg6 | Packet classifier configuration register6 | 103 |
| 54 | Packet_classifier_cfg_reg7 | Packet classifier configuration register7 | 103 |
| 58 | Packet_classifier_cfg_reg8 | Packet classifier configuration register8 | 104 |
| 5C | Packet_classifier_cfg_reg9 | Packet classifier configuration register9 | 104 |
| 60 | Packet_classifier_cfg_reg10 | Packet classifier configuration register10 | 104 |
| 64 | Packet_classifier_cfg_reg11 | Packet classifier configuration register11 | 104 |
| 68 | Packet_classifier_cfg_reg12 | Packet classifier configuration register12 | 104 |
| 6C | Packet_classifier_cfg_reg13 | Packet classifier configuration register13 | 105 |
| 70 | Packet_classifier_cfg_reg14 | Packet classifier configuration register14 | 105 |
| 74 | Packet_classifier_cfg_reg15 | Packet classifier configuration register15 | 105 |
| 78 | Packet_classifier_cfg_reg16 | Packet classifier configuration register16 | 105 |
| 7C | Packet_classifier_cfg_reg17 | Packet classifier configuration register17 | 105 |
| 80 | Packet_classifier_cfg_reg18 | Packet classifier configuration register18 | 105 |
| D4 | CPU_rx_arb_max_fifo_level_reg | Rx arbiter maximum FIFO level register | 106 |

**Table 11-5. TDMoP Status Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0xE0 | General_stat_reg | General latched status register | 107 |
| E4 | Version_reg | TDMoP version register | 107 |
| E8 | Port1_sticky_reg1 | Port 1 latched status register | 107 |
| EC | Port1_sticky_reg2 | Port 2 latched status register | 107 |
| F0 | Port1_sticky_reg3 | Port 3 latched status register | 107 |
| F4 | Port1_sticky_reg4 | Port 4 latched status register | 107 |
| F8 | Port1_sticky_reg5 | Port 5 latched status register | 107 |
| FC | Port1_sticky_reg6 | Port 6 latched status register | 107 |
| 100 | Port1_sticky_reg7 | Port 7 latched status register | 107 |
| 104 | Port1_sticky_reg8 | Port 8 latched status register | 107 |
| 108 | Port1_status_reg1 | Port 1 status bit register 1 | 108 |
| 10C | Port1_status_reg2 | Port 1 status bit register 2 | 108 |

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 110 | Port2_status_reg1 | Port 2 status bit register 1 | 108 |
| 114 | Port2_status_reg2 | Port 2 status bit register 2 | 108 |
| 118 | Port3_status_reg1 | Port 3 status bit register 1 | 108 |
| 11C | Port3_status_reg2 | Port 3 status bit register 2 | 108 |
| 120 | Port4_status_reg1 | Port 4 status bit register 1 | 108 |
| 124 | Port4_status_reg2 | Port 4 status bit register 2 | 108 |
| 128 | Port5_status_reg1 | Port 5 status bit register 1 | 108 |
| 12C | Port6_status_reg2 | Port 5 status bit register 2 | 108 |
| 130 | Port6_status_reg1 | Port 6 status bit register 1 | 108 |
| 134 | Port6_status_reg2 | Port 7 status bit register 2 | 108 |
| 138 | Port7_status_reg1 | Port 7 status bit register 1 | 108 |
| 13C | Port7_status_reg2 | Port 7 status bit register 2 | 108 |
| 140 | Port8_status_reg1 | Port 8 status bit register 1 | 108 |
| 144 | Port8_status_reg2 | Port 8 status bit register 2 | 108 |

#### *11.4.1.1 TDMoP Configuration Registers*

**General_cfg_reg0 0x00**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | Discard_ip_checksum_err | R/W | 0x0 | Indicates to discard packets received with a wrong IP checksum. See section 10.6.13. |
| [30:27] | Packet_trailer_length | R/W | 0x0 | The length of the trailer attached to all received and transmitted packets. Allowed values: 0–12 (decimal). When set to zero no trailer is attached. See section 10.6.14. |
| [26] | Clock_recovery_en | R/W | 0x0 | 0 = Clock recovery block is disabled (power saving mode)<br>1 = Normal operation<br>Should be cleared to reduce the chip power consumption when adaptive clock recovery is not used. When cleared, the clock recovery registers (offset 0x48,000) must not be accessed by the CPU because the clock recovery block does not assert H_READY_N. See section 10.4. |
| [25:16] | Rx_fifo_priority_lvl | R/W | 0x100 | Rx FIFO threshold level in dwords. If the Rx FIFO level is higher than this threshold, then the Rx_fifo receives the higher priority instead of the cross-connect queue.<br>This parameter is relevant only when there are bundles configured as cross-connect. The recommended value is 0x3FF (maximal value). See section 10.6.11.5. |
| [15:14] | MII_mode_select | R/W | 0x0 | 00 = MII<br>01 = RMII<br>10 = Reserved<br>11 = Source sync SMII (SSMII) |
| [13:12] | Reserved | R/W | 0x0 | Must be set to zero |
| [11] | High_speed | R/W | 0x0 | 0 = All ports active in E1/T1/J1 mode<br>1 = Port1 enabled in high-speed E3/T3/STS-1 mode, all other ports disabled |
| [10] | OAM_timestamp_resolution | R/W | 0x1 | 0 = OAM timestamp is incremented every 1µs<br>1 = OAM timestamp is incremented every 100µs<br>See section 10.6.13.6. |
| [9:8] | Reserved | R/W | 0x0 | Must be set to zero |
| [7] | Mem_size | R/W | 0x0 | SDRAM size:<br>0 = 64 Mb<br>1 = 128 Mb |

**General_cfg_reg0 0x00**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [6:5] | Fq | R/W | 0x0 | SDRAM clock:<br>00 = 50 MHz<br>01 = 75 MHz<br>10 = Reserved<br>11 = Reserved for 100 MHz |
| [4:3] | Col_width | R/W | 0x0 | SDRAM columns and rows<br>00 = 8 bit (256 columns)<br>01 = 9 bit (512 columns)<br>10 = 10 bit (1K columns)<br>11 = 11 bit (2K columns) |
| [2:1] | CAS_latency | R/W | 0x2 | SDRAM CAS latency:<br>00 = {reserve value}<br>01 = 1<br>10 = 2<br>11 = 3 |
| [0] | Rst_SDRAM_n | R/W | 0x0 | Resets SDRAM controller. Active low.<br>After all configuration bits of the SDRAM controller have been written, the SDRAM controller must be reset by taking this bit low then high. |

**General_cfg_reg1 0x04**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | RTP_timestamp_generation_ mode | R/W | 0x0 | Indicates the RTP timestamp generation mode:<br>0 = Absolute mode<br>1 = Differential (common clock) mode<br>See the description of the TS field in Table 10-16 for more details. |
| [30:24] | Sw_packet_offset | R/W | 0x04 | The offset from the first byte of the packet to the start of the CPU buffer.<br>For the Ethernet-to-CPU packets, 8 bytes are added automatically to each configured value. For example, if you intend to set the offset to 20 bytes, configure this value to 12 bytes.<br>Allowed values are in the range of 4–127 (decimal) bytes. |
| [23:19] | Tx_payload_offset | R/W | 0x00 | Number of 32-bit words between the start of transmit buffer to the control word or to start of the TDM payload if the control word does not exist |
| [18] | Reserved | R/W | 0x0 | Must be set to zero |
| [17:10] | JBC_sig_base_add | R/W | 0x060 | Base address (8 MSbits) of Rx jitter buffer signaling section in SDRAM |
| [9:6] | Tx_buf_base_add | R/W | 0x2 | Base address (4 MSbits) of transmit buffers in SDRAM |
| [5] | IP_version | R/W | 0x0 | The IP version of transmitted TDMoP packets. See section 10.6.13.<br>0 = Ipv4<br>1 = Ipv6 |
| [4] | Dual_stack | R/W | 0x0 | The IP version of received TDMoP packets . See section 10.6.13.<br>0 = Ipv4/Ipv6, according to IP_version field above<br>1 = Both Ipv4 and Ipv6 packets |
| [3] | Frames_count_check_en | R/W | 0x1 | Specifies whether to check received packets that are CESoPSN structured with CAS bundles and discard those that contain the wrong number of TDM frames<br>0 = Do not check<br>1 = Check |
| [2] | Reserved | R/W | 0x0 | Must be set to zero |
| [1:0] | JBC_data_base_add | R/W | 0x0 | Base address (2 MSbits) of Rx jitter buffer data section in SDRAM |

**General_cfg_reg2 0x08**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:29] | Rx_HDLC_min_flags | R/W | 0x0 | Minimum number flags between 2 adjacent HDLC frames transmitted on the TDM pins. The number of flags is equal to Rx_hdlc_min_flags + 1. Range: 1 – 8. |
| [28:24] | Reserved | R/W | 0x0 | Must be set to zero |
| [23:20] | Rx_SAToP/CESoPSN_discard_ mask | R/W | 0x0 | Each bit of this field determines whether a specific type of discarded packet is to be counted by the 'SAToP/CESoPSN _discarded_packets' counter.<br>0 = don't count<br>1 = count<br>bit 23: count packets that were discarded because of jump operation that caused overflow in jitter buffer.<br>bit 22: count packets that were discarded due to incorrect sequence number.<br>bit 21: count packets that were discarded due to over-run state in jitter buffer.<br>bit 20: count packets that were discarded because they were considered duplicated, or because they were received too late to be inserted into the jitter buffer. |
| [19:0] | Reserved | R/W | 0x0 | Must be set to zero |

In the Port[n]_cfg_reg description below, the index **n** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101.

**Port[n]_cfg_reg 0x08+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:30] | Reserved | R/W | 0x0 | Must be set to zero. |
| [29:24] | Unframed_int_rate | R/W | 0x0 | The bit rate of an unframed interface type (Used only for absolute mode RTP timestamping).<br>1 = 64 kbps<br>2 = 128 kbps<br>.<br>.<br>.<br>32 = 2.048 Mbps<br>33 =1.544 Mbps<br>34 = 34 Mbps (E3 rate)<br>45 = 45 Mbps (T3 rate)<br>52 = 51.84 Mbps (STS-1 rate)<br>**Note**: E3, T3 and STS-1 configurations are available for Port 1 only in high-speed mode, i.e. when General_cfg_reg0.High_speed=1. |
| [23] | PCM_rate | R/W | 0x0 | Indicates the PCM frequency, i.e. the TDM rate in and out of the TDMoP port. Only applies when int_frame_type (bits 3:2 below) is set for frame, multiframe or ESF and int_type (bits 1:0 below) is set for E1 or T1.<br>0 = 1.544 MHz<br>1 = 2.048 MHz<br>This bit is for enabling T1 data over an E1-rate port. The combination of Int_type=E1 and PCM_rate=1.544 MHz is not allowed. |

**Port[n]_cfg_reg 0x08+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [22:21] | Tx_defect_modifier | R/W | 0x0 | Used in the control word M field for packets in all bundles associated with TDMoP port n. |
| [20] | Port_Rx_enable<br>(Rx means from Ethernet MII) | R/W | 0x0 | 0 = Outgoing TDM traffic from Port n of the TDMoP block is discarded (TDMn_TX and TDMn_TSIG are held high)<br>1 = Outgoing TDM traffic from Port n of the TDMoP block is enabled.<br>**Note**: (Port 1 only) This bit also applies in high-speed mode, i.e. when General_cfg_reg0.High_speed=1. |
| [19] | CTS | R/W | 0x1 | When the Int_type field (below) specifies a serial interface, the value of the TDMn_TSIG_CTS pin--which behaves as CTS (Clear To Send)—comes from this field. |
| [18] | CD_en | R/W | 0x0 | When the Int_type field (below) specifies a serial interface, this field is the output enable control for the CD (Carrier Detect) function of the TDMn_TX_MF_CD pin. When this pin is active, the output state of the TDMn_TX_MF_CD pin comes from the CD field (below). |
| [17] | CD | R/W | 0x1 | When the Int_type field (below) specifies a serial interface, the value of the TDMn_TX_MF_CD pin—which behaves as CD (Carrier Detect)—comes from this field when the CD_en bit (above) is high. |
| [16] | Loss | R/W | 0x0 | Loss of sync on TDM port n. Causes the L bit in the control word to be set for packets in all bundles associated with TDMoP port n. |
| [15:11] | Adapt_JBC_indx | R/W | 0x00 | Index of the jitter buffer used by the clock recovery block to generate the clock for TDMoP port n. |
| [10:9] | SF_to_ESF_low_CAS_bits | R/W | 0x0 | In the case where a SF (superframe) formatted T1 is connected by a structured-with-CAS bundle to an ESF interface, this field is the source of the C and D CAS bits for the ESF interface (in the Ethernet-to-TDM direction). See section 10.6.5. |
| [8] | TSA_act_blk | R/W | 0x0 | 0 = TSA bank1 is the active bank for Port n.<br>1 = TSA bank2 is the active bank for Port n.<br>Swapping banks takes effect at the next sync input assertion |
| [7] | Port_Tx_enable<br>(Tx mean toward Ethernet MII) | R/W | 0x0 | 0 = Incoming TDM traffic to Port n of the TDMoP block is discarded<br>1 = Incoming TDM traffic to Port n of the TDMoP block is enabled<br>**Note**: (Port 1 only) This bit also applies in high-speed mode, i.e. when General_cfg_reg0.High_speed=1. |
| [6] | Rx_sample | R/W | 0x1 | In one-clock mode (Two_clocks field below is 0) this field is ignored. In two-clock mode (Two_clocks=1) this field specifies the TDMn_RCLK edge on which TDMn_RX, TDMn_RX_SYNC and TDMn_RSIG_RTS are sampled.<br>0 = falling edge<br>1 = rising edge<br>See the timing diagrams in Figure 14-10 through Figure 14-13. |
| [5] | Tx_sample | R/W | 0x0 | In one-clock mode (Two-clocks field below is 0) this field specifies the TDMn_TCLK edge on which TDMn_TX_SYNC, TDMn_TX_MF_CD, TDMn_RX, TDMn_RX_SYNC and TDMn_RSIG_RTS are sampled and the edge on which TDMn_TX and TDMn_TSIG_CTS are updated.<br>0 = Inputs sampled on the falling edge, outputs updated on the rising edge<br>1 = Inputs sampled on the rising edge, outputs updated on the falling edge |

**Port[n]_cfg_reg 0x08+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | In two-clock mode (Two-clocks=1) this field specifies the TDMn_TCLK edge on which TDMn_TX_SYNC, TDMn_TX_MF_CD are sampled and the edge on which TDMn_TX and TDMn_TSIG_CTS are updated. The Rx_sample field (above) specifies the TDMn_RCLK edge for the Rx-side signals.<br>0 = Inputs sampled on the falling edge, outputs updated on the rising edge<br>1 = Inputs sampled on the rising edge, outputs updated on the falling edge<br>See the timing diagrams in Figure 14-8 through Figure 14-13. |
| [4] | Two_clocks | R/W | 0x1 | One-clock or two-clock mode.<br>0 = one-clock mode: TDMn_TCLK is used for both Rx and transmit interfaces<br>1 = two-clock mode: TDMn_RCLK is used for the Rx interface and TDMn_TCLK is used for the transmit interface.<br>***Note**: (Port 1 only) This bit must be set in high-speed mode (i.e. when General_cfg_reg0.High_speed=1).* |
| [3:2] | Int_framed_type | R/W | 0x0 | Interface Framing Type<br>00 = Unframed (no frame sync, no multiframe sync)<br>01 = Frame (frame sync only, no multiframe sync)<br>10 = Multiframe (E1), SF (T1) (sync and mf sync)<br>11 = ESF(T1) (frame sync and multiframe sync)<br>Changing value from 10 or 11 to 00 or 01 must be performed only after asserting the RST_SYS_N pin. |
| [1:0] | Int_type | R/W | 0x1 | Interface Type<br>00 = Serial<br>01= E1<br>10 = T1<br>11 = Reserved |

**Rst_reg 0x2C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | Reserved | - | 0x0 | Must be set to zero |
| [27:24] | Rst_tx_port_num | R/W | 0x0 | Port number associated with Rst_tx field (below).<br>0000 = Port 1<br>0001 = Port 2<br>0010 = Port 3<br>0011 = Port 4<br>0100 = Port 5<br>0101 = Port 6<br>0110 = Port 7<br>0111 = Port 8 |
| [23:18] | Rst_tx_internal_bundle_num | R/W | 0x00 | Bundle number associated with Rst_tx field (below) |
| [17] | Rst_tx_open/close | R/W | 0x0 | Valid when Rst_tx is set<br>0 = When Rst_tx is done during bundle close procedure<br>1 = When Rst_tx is done during bundle open procedure<br>This bit is also used in high-speed mode. |
| [16] | Rst_tx | R/W | 0x0 | If set, the relevant transmit payload type machine resets its variables (should be given with bundle number and a proper value of the RST_tx_open/close bit). The CPU should poll this bit until it is 0 meaning, “reset acknowledged”. This bit is also used in high-speed mode. |
| [15:7] | Reserved | R/W | 0x0 | Must be set to zero |
| [6:1] | Rst_rx_internal_bundle_num | R/W | 0x00 | Bundle number associated with Rst_rx |
| [0] | Rst_rx | R/ set | 0x0 | 1 = Packet classifier generates a reset frame (Rst_rx_internal_bundle_num is valid). The CPU should poll this bit until it finds 0; this means “reset acknowledged”. |

The TDM_cond_data_reg register below holds four octets to be transmitted as conditioning data in the TDM direction during jitter buffer underrun. This data applies to all bundle types.

**TDM_cond_data_reg 0x30**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:24] | TDM_cond_octet_a | R/W | 0x00 | TDM Conditioning Octet A<br>Must be set to 0x7E for HDLC bundles<br>Also used in high-speed mode |
| [23:16] | TDM_cond_octet_b | R/W | 0x00 | TDM Conditioning Octet B<br>Must be set to 0x7E for HDLC bundles |
| [15:8] | TDM_cond_octet_c | R/W | 0x00 | TDM Conditioning Octet C<br>Must be set to 0x7E for HDLC bundles |
| [7:0] | TDM_cond_octet_d | R/W | 0x00 | TDM Conditioning Octet D<br>Must be set to 0x7E for HDLC bundles |

The ETH_cond_data_reg register below holds four octets to be transmitted as conditioning data towards the packet network (i.e. toward the Ethernet MAC) when no valid data is available from the TDM port. This applies only to AAL1 or SAToP/CESoPSN bundles. Tx_cond_octet_type in the Bundle Configuration Tables specifies which of these octets is used on a per-bundle basis.

**ETH_cond_data_reg 0x34**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:24] | ETH_cond_octet_d | R/W | 0x00 | Ethernet Conditioning octet D |
| [23:16] | ETH_cond_octet_c | R/W | 0x00 | Ethernet Conditioning octet C |
| [15:8] | ETH_cond_octet_b | R/W | 0x00 | Ethernet Conditioning octet B |
| [7:0] | ETH_cond_octet_a | R/W | 0x00 | Ethernet Conditioning octet A |

**Packet_classifier_cfg_reg0 0x38**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv4_add1 | R/W | 0x0 | This field holds the first of three IPv4 addresses for the device. The other addresses are held in register Packet_classifier_cfg_reg1 and Packet_classifier_cfg_reg8. Relevant only for packets received from the Ethernet port. |

**Packet_classifier_cfg_reg1 0x3C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv4_add2 | R/W | 0x0 | This field holds the second of three IPv4 addresses for the device. The other addresses are held in register Packet_classifier_cfg_reg0 and Packet_classifier_cfg_reg8. Relevant only for packets received from the Ethernet port. |

**Packet_classifier_cfg_reg2 0x40**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | MAC_add1 | R/W | 0x0 | This field holds bits 31:0 of the first of two MAC addresses for the device. The upper bits of this MAC address are in Packet_classifier_cfg_reg3. The other MAC address is in Packet_classifier_cfg_reg5 and |

**Packet_classifier_cfg_reg2 0x40**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | Packet_classifier_cfg_reg6.<br>Relevant only for packets received from Ethernet port. |

**Packet_classifier_cfg_reg3 0x44**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:29] | Reserved | - | 0x0 | Must be set to zero |
| [28] | Discard_packet_length_ mismatch | R/W | 0x0 | Must be set to zero |
| [27] | Ip_udp_bn_loc | R/W | 0x0 | 0 = Bundle identifier is located in the source UDP port number field in IP/UDP packets<br>1 = Bundle identifier located in the destination UDP port number field in IP/UDP packets<br>See section 10.6.13.2. |
| [26:25] | TDMoIP_port_num_loc | R/W | 0x0 | Used for UDP only:<br>00 = Packet_classifier_cfg_reg4.TDMoIP_port_num1/2 is ignored (no checking is performed)<br>01 = TDMoIP_port_num1/2 should be compared to the source UDP port number field in IP/UDP packets<br>10 = TDMoIP_port_num1/2 should be compared to the destination UDP port number field in IP/UDP packets<br>11 = Reserved<br>See section 10.6.13.1. |
| [24] | Discard_switch_8 | R/W | 0x0 | Packets with Ethertype = CPU_dest_ether_type. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |
| [23] | Discard_switch_7 | R/W | 0x0 | TDMoP OAM packets. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |
| [22] | Discard_switch_6 | R/W | 0x0 | TDMoP packets whose Rx_Bundle_Identifier doesn't match any of the chip's assigned bundle numbers or OAM bundle numbers. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |
| [21] | Discard_switch_5 | R/W | 0x0 | IP/UDP packets whose UDP destination/source port number is different from Packet_classifier_cfg_reg4. TDMoIP_Port_Num1 or 2. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard<br>See TDMoIP_port_num_loc above. |
| [20] | Discard_switch_4 | R/W | 0x0 | IP packets whose IP protocol field is different from UDP or L2TPv3. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |
| [19] | Discard_switch_3 | R/W | 0x0 | ARP packets whose IP destination address matches one of the chip's IPv4 addresses. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |
| [18] | Discard_switch_2 | R/W | 0x0 | Packets with Ethertype different from IP, MPLS or ARP. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |
| [17] | Discard_switch_1 | R/W | 0x0 | IP packets whose IP destination address does not match chip's IP addresses. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |

**Packet_classifier_cfg_reg3 0x44**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [16] | Discard_switch_0 | R/W | 0x0 | ARP packets whose IP destination address does not match chip's addresses. See section 10.6.13.<br>0 = Forward to CPU<br>1 = Discard |
| [15:0] | MAC_add1 | R/W | 0x0000 | This field holds bits 47:32 of the first of two MAC addresses for the device. The lower bits of this MAC address are in Packet_classifier_cfg_reg2. The other MAC address is in Packet_classifier_cfg_reg5 and Packet_classifier_cfg_reg6. Relevant only for packets received from Ethernet port. |

**Packet_classifier_cfg_reg4 0x48**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | TDMoIP_port_num2 | R/W | 0x085E | Packets with UDP destination port number equal to this field are recognized as TDMoIP packets. See section 10.6.13.1. |
| [15:0] | TDMoIP_port_num1 | R/W | 0x085E | Packets with UDP destination port number equal to this field are recognized as TDMoIP packets. See section 10.6.13.1. |

**Packet_classifier_cfg_reg5 0x4C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | MAC_add2 | R/W | 0x0 | This field holds bits 31:0 of the second of two MAC addresses for the device. The upper bits of this MAC address are in Packet_classifier_cfg_reg6. The other MAC address is in Packet_classifier_cfg_reg2 and Packet_classifier_cfg_reg3. Relevant only for packets received from Ethernet port. |

**Packet_classifier_cfg_reg6 0x50**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Ip_udp_bn_mask_n | R/W | 0x0000 | This mask Indicates the width of the bundle identifier. For example, if the desired width is 8 bits, the following should be written to this field: 0000000011111111b. See section 10.6.13.2. |
| [15:0] | MAC_add2 | R/W | 0x0000 | This field holds bits 47:32 of the second of two MAC addresses for the device. The lower bits of this MAC address are in Packet_classifier_cfg_reg5. The other MAC address is in Packet_classifier_cfg_reg2 and Packet_classifier_cfg_reg3. Relevant only for packets received from Ethernet port. |

**Packet_classifier_cfg_reg7 0x54**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | CPU_dest_ether_type | R/W | 0x0800 | Ethertype which identifies packets destined for the CPU. Such packets are sent to CPU or discarded as specified by Packet_classifier_cfg_reg3.Discard_switch_[8:0]. This field must be set to a value greater than 0x5DC. See |

**Packet_classifier_cfg_reg7 0x54**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | section 10.6.13.5. |
| [15:0] | vlan_2nd_tag_identifier | R/W | 0x8100 | Second VLAN tag protocol identifier (the first is 0x8100). See section 10.6.13.4. |

**Packet_classifier_cfg_reg8 0x58**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv4_add3 | R/W | 0x0 | This field holds the third of three IPv4 addresses for the device. The other addresses are held in register Packet_classifier_cfg_reg0 and Packet_classifier_cfg_reg1. Relevant only for packets received from the Ethernet port. If a third IPv4 address is not needed, this field must be configured to the same value as Ipv4_add1. |

**Packet_classifier_cfg_reg9 0x5C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Mef_ether_type | R/W | 0x88d8 | Ethertype for MEF packets. Must be set to a value greater than 0x5DC. See section 10.6.13.5. |
| [15:0] | Mef_oam_ether_type | R/W | 0x0800 | Ethertype for MEF OAM packets. Must be set to a value greater than 0x5DC. See section 10.6.13.3. |

**Packet_classifier_cfg_reg10 0x60**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add1[127:96] | R/W | 0x0 | This field holds bits 127:96 of the first of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg14. Relevant only for packets received from the Ethernet port. |

**Packet_classifier_cfg_reg11 0x64**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add1[95:64] | R/W | 0x0 | This field holds bits 95:64 of the first of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg14. Relevant only for packets received from the Ethernet port. |

**Packet_classifier_cfg_reg12 0x68**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add1[63:32] | R/W | 0x0 | This field holds bits 63:32 of the first of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg14. Relevant only for packets received from the Ethernet port. |

**Packet_classifier_cfg_reg13 0x6C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add1[31:0] | R/W | 0x0 | This field holds bits 31:0 of the first of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg14. Relevant only for packets received from the Ethernet port. |

**Packet_classifier_cfg_reg14 0x70**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add2[127:96] | R/W | 0x0 | This field holds bits 127:96 of the second of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg10. Relevant only for packets received from the Ethernet port. If a second IPv6 address is not needed, this field must be configured to the same value as Ipv6_add1. |

**Packet_classifier_cfg_reg15 0x74**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add2[95:64] | R/W | 0x0 | This field holds bits 95:64 of the second of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg10. Relevant only for packets received from the Ethernet port. If a second IPv6 address is not needed, this field must be configured to the same value as Ipv6_add1. |

**Packet_classifier_cfg_reg16 0x78**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add2[63:32] | R/W | 0x0 | This field holds bits 63:32 of the second of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg10. Relevant only for packets received from the Ethernet port. If a second IPv6 address is not needed, this field must be configured to the same value as Ipv6_add1. |

**Packet_classifier_cfg_reg17 0x7C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ipv6_add2[31:0] | R/W | 0x0 | This field holds bits 31:0 of the second of two IPv6 addresses for the device. The other address is held in registers starting with Packet_classifier_cfg_reg10. Relevant only for packets received from the Ethernet port. If a second IPv6 address is not needed, this field must be configured to the same value as Ipv6_add1. |

**Packet_classifier_cfg_reg18 0x80**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | VCCV_oam_mask_n | R/W | 0x0000 | Indicates which of the 16 most significant bits of the control word should be compared to identify VCCV OAM packets. The values of the bits to be compared are stored |

**Packet_classifier_cfg_reg18 0x80**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | in the VCCV_oam_value field below. See section 10.6.13.3. |
| [15:0] | VCCV_oam_value | R/W | 0x0000 | Indicates the value of the 16 most significant bits of the control word for identifying VCCV OAM packets. The combination of this field and VCCV_oam_mask_n above specifies how the device does VCCV OAM identification. For example, to identify VCCV OAM packets when the 4 most significant bits of the control word are equal to 0x1, then set this field to 0x1000 and set VCCV_oam_mask_n to 0xF000. See section 10.6.13.3. |

**CPU_rx_arb_max_fifo_level_reg 0xD4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:25] | Tx_arb_max_fifo_level | R/W | 0x00 | Indicates the maximum level, which the TX_FIFO has reached (given in dwords) since the last time this register was read (or since reset). The value of the field is automatically reset when this register is read by the CPU. |
| [24:10] | Reserved | - | 0x0000 | Must be set to zero |
| [9:0] | Rx_arb_max_fifo_level | R/W | 0x000 | Indicates the maximum level, which the RX_FIFO has reached (given in dwords) since the last time this register was read (or since reset). The value of the field is automatically reset when this register is read by the CPU. |

#### 11.4.1.2 TDMoP Status Registers

The General_stat_reg register has latched status registers that indicate hardware events. For each bit, the value 1 indicates that the event occurred. Writing 1 to a bit clears it to 0. Writing 0 to a bit does not change its value.

**General_stat_reg 0xE0**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:10] | Reserved | - | 0x0 | Must be set to zero |
| [9] | MAC_Rx_fifo_overrun | R/W | 0x0 | MAC Rx FIFO overflowed |
| [8] | Ipver_err_status | R/W | 0x0 | Indicates that a packet was discarded due to IP version error |
| [7] | Rx_fifo_sof_err | R/W | 0x0 | Rx FIFO was flushed due to bundle configuration error |
| [6] | TDM_CPU_buff_err | R/W | 0x0 | Frames received from TDM discarded due to lack of buffers at TDM TO CPU pool |
| [5] | Rx_fifo_full | R/W | 0x0 | Packet received from Ethernet discarded because Rx FIFO is full |
| [4] | MPLS_err | R/W | 0x0 | Received MPLS packet with more than three labels |
| [3] | OAM_ETH_to_CPU_q_full | R/W | 0x0 | OAM packet received from Ethernet and destined to CPU discarded because ETH TO CPU queue is full. |
| [2] | OAM_SW_buff_err | R/W | 0x0 | OAM packet received from Ethernet and destined to CPU discarded due to lack of SW buffers |
| [1] | Non_OAM_ETH_to_CPU_q_full | R/W | 0x0 | Non-OAM packet received from Ethernet and destined to CPU discarded because ETH TO CPU queue is full. |
| [0] | Non_OAM_SW_buff_err | R/W | 0x0 | Non-OAM packet received from Ethernet and destined to CPU discarded due to lack of SW buffers. |

**Version_reg 0xE4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Chip_version_reg | R/O | 0xABCD EF01 | Contains the chip version for the TDMoP block |

The Port[n]_sticky_reg1 register has latched status bits that indicate port hardware events. For each bit, the value 1 indicates that the event occurred. Writing 1 to a bit clears it to 0. Writing 0 to a bit does not change its value. The index **n** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101.

**Port[n]_sticky_reg1 0xE4+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7] | Dpll_ovrflow | R/W | 0x0 | Port clock recovery DPLL overflowed |
| [6] | Cdc_detected | R/W | 0x0 | Port clock recovery detected constant delay change in the network |
| [5] | Smart_self_test_failed | R/W | 0x0 | Provided for debug purposes |
| [4] | Smart_timeout_expired | R/W | 0x0 | Provided for debug purposes |
| [3] | Sticky_filter_ovrflow | R/W | 0x0 | Port clock recovery loop filter overflowed |
| [2] | Virtual_jitter_buffer_or_ur | R/W | 0x0 | Port clock recovery virtual jitter buffer reached overrun/ underrun state |
| [1] | Reacquisition_alarm | R/W | 0x0 | Provided for debug purposes |
| [0] | Adapt_freeze_state | R/W | 0x0 | Port clock recovery mechanism is in freeze state |

The Port[n]_stat_reg1 register has real-time (not latched) status fields. The index **n** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101.

**Port[n]_stat_reg1 0x100+n*8**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:25] | Reserved | - | 0x0 | Must be set to zero |
| [24] | Smart_disabled | RO | 0x0 | not documented |
| [23:5] | DPLL_level | RO | 0x0 | not documented |
| [4:2] | Adapt_current_state | RO | 0x0 | Port n clock recovery current state:<br>0 = Idle<br>2 = Acquisition<br>3 = Tracking1<br>4 = Tracking2<br>5 = Recover from Underrun/Overrun |
| [1] | RTS | RO | 0x0 | When the Port[n]_cfg_reg.Int_type field specifies a serial interface, the value of the TDMn_RSIG_RTS pin--which behaves as RTS (Request To Send)—can be read from this bit. |
| [0] | TSA_int_act_blk | RO | 0x0 | Indicates which bank is active:<br>0 = Port n TSA bank1 is active<br>1 = Port n TSA bank2 is active |

The Port[n]_stat_reg2 register has real-time (not latched) status fields. The index **n** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101.

**Port[n]_stat_reg2 0x104+n*8**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:29] | Bw_tunn | RO | 0x0 | not documented |
| [28:4] | Curr_pdv_std | RO | 0x0 | not documented |
| [3:0] | Convergence_counter | RO | 0x0 | not documented |

### 11.4.2 Bundle Configuration Tables

The base address for the TDMoP bundle configuration tables is **0x8,000**. Bundle configurations are 160 bits long and therefore span five 32-bit words. The least-significant 32-bit word of a bundle configuration is located at address offset 0x000 + BundleNumber x 4. The most-significant 32-bit word is located at address offset 0x400 + BundleNumber x 4. There are 64 bundles numbered 0 to 63. In the register descriptions in this section the index **n** indicates bundle number: 0 to 63.

Each bundle can be one of three different types: AAL1, HDLC or SAToP/CESoPSN. Subsections 11.4.2.1 through 11.4.2.3 describe the bundle configuration fields for each of the four types. Some fields are common to two or more of the bundle types. The payload type is specified in the Payload_type_machine field, bits 21:20 of xxxx_Bundle[n]_cfg[63:32].

#### 11.4.2.1 AAL1 Bundle Configuration

In the register descriptions below, the index **n** indicates the bundle number: 0 to 63.

**AAL1_Bundle[n]_cfg[31:0] 0x000+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Rx_bundle_identifier | R/W | None | Holds the Rx bundle number |

**AAL1_Bundle[n]_cfg[63:32] 0x100+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:22] | Rx_max_buff_size | R/W | None | The size of the jitter buffer. See section 10.6.10. Also the maximum time interval for which data is stored. The resolution is determined by the interface type as follows:<br>For framed E1/T1: 0.5 ms.<br>For unframed E1/T1 or serial bundles: 1024 bit periods<br>For high-speed interface: 4096 bit periods.<br>Allowed values:<br>For T1-SF: Rx_max_buff_size ≤ 0x2FC<br>For T1-ESF: Rx_max_buff_size ≤ 0x2F9<br>For E1-MF: Rx_max_buff_size ≤ 0x3FB<br>For all interface types, the Rx_max_buff_size must be greater than Rx_PDVT + PCT (Packet Creation Time).<br>**Note**: For unframed, the Rx_max_buff_size resolution is different than PDVT resolution. |
| [21:20] | Payload_type_machine | R/W | None | 00 = HDLC<br>01 = AAL1<br>10 = Reserved<br>11 = SAToP/CESoPSN |
| [19] | Tx_RTP<br>(Tx is toward Ethernet MAC) | R/W | None | 0 = RTP header does not exist in transmitted packets<br>1 = RTP header exists in transmitted packets |
| [18] | Control_Word_exists | R/W | None | 0 = Control word does not exist<br>1 = Control word exists (default, standard mode) |
| [17:16] | Tx_dest | R/W | None | Destination of packets:<br>00 = Reserved<br>01 = Ethernet<br>10 = CPU<br>11 = TDM (Cross-connect). See section 10.6.11.10. |
| [15:9] | Rx_max_lost_packets | R/W | None | The maximum number of Rx packets inserted upon detection of lost packets |
| [8:4] | Number_of_ts | R/W | None | One less than number of assigned timeslots per bundle. When Rx_AAL1_bundle_type='00' (unstructured) then Number_of_ts=31; this applies also to high speed mode. |
| [3] | Rx_ discard_sanity_fail | R/W | None | 0 = Discard AAL1 packets which fail the sanity check<br>1 = Don't discard the above packets<br>See section 10.6.13.8. |
| [2:1] | Header_type | R/W | None | 00 = MPLS<br>01 = UDP over IP<br>10 = L2TPv3 over IP<br>11 = MEF |
| [0] | Tx_R_bit | R/W | None | 0 = Don't set R bit in header of transmitted packets<br>1 = Set R bit |

**AAL1_Bundle[n]_cfg[95:64] 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | Reserved | R/W | None | Must be set to zero |
| [30] | Tx_cond_data | R/W | None | 0 = Regular operation<br>1 = Use conditioning octet specified by Tx_cond_octet_type for transmitted packets |
| [29] | Tx_dest_framing | R/W | None | Only applies to T1 framed traffic. See section 10.6.5.<br>0 = Destination framer operates in SF framing<br>1 = Destination framer operates in ESF framing |
| [28] | Tx_CAS_source | R/W | None | Source of transmit CAS bits:<br>0 = TDMoP block's RSIG input<br>1 = Tx software CAS table (section 11.4.9) |
| [27:13] | Reserved | - | None | Must be set to zero |
| [12:11] | Tx_AAL1_bundle_type | R/W | None | Bundle type of transmitted payload: |

**AAL1_Bundle[n]_cfg[95:64] 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | 00 = Unstructured<br>01 = Structured<br>10 = Structured with CAS<br>11 = Reserved |
| [10:6] | Reserved | R/W | None | Must be set to zero |
| [5:4] | Tx_cond_octet_type | R/W | None | Selects the ETH_cond_octet from ETH_cond_data_reg to be transmitted towards packet network:<br>00 = ETH_cond_octet_a<br>01 = ETH_cond_octet_b<br>10 = ETH_cond_octet_c<br>11 = ETH_cond_octet_d |
| [3:2] | Rx_AAL1_bundle_type | R/W | None | Bundle type of received packets:<br>00 = Unstructured<br>01 = Structured<br>10 = Structured with CAS<br>11 = Reserved |
| [1:0] | Protection_mode | R/W | None | 00 = Stop sending packets<br>01 = Send each packet once with the first header<br>10 = Send each packet once with the second header<br>11 = Send each packet twice: once with the first header and once with the second header<br>See section 10.6.16. |

**AAL1_Bundle[n]_cfg[127:96] 0x300+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | Reserved | R/W | None | Must be set to zero |
| [30:16] | Rx_PDVT | R/W | None | Packet delay variation time value for AAL1 bundles. See section 10.6.10. Bits [30:26] are used only when unframed. The resolution is determined by the interface type as follows:<br>For framed E1/T1: 0.5 ms<br>For unframed E1/T1 or serial bundles: 32 bit periods<br>For high speed interface: 128 bit periods<br>Allowed values:<br>Minimum allowed value: 3 (for all interfaces types)<br>For T1 SF, ESF: Rx_PDVT < 0x300 |
| [15] | Rx_CAS_src | R/W | None | Source of signaling conditioning towards TDM:<br>0 = SDRAM signaling jilter buffer<br>1 = Rx SW CAS table (section 11.4.13) |
| [14] | Rx_cell_chk_ignore | R/W | None | 0 = Discard AAL1 SAR PDUs with header parity/CRC errors<br>1 = Ignore AAL1 SAR PDU header (CRC /parity) checks Including AAL1 pointer parity error |
| [13] | Reserved | R/W | None | Must be set to zero |
| [12] | OAM_ID_in_CW | R/W | None | 0 = Ignore the OAM packet indication in the control word<br>1 = Check the OAM packet indication in the control word<br>See section 10.6.13.3. |
| [11] | Rx_discard | R/W | None | 0 = Pass through all incoming packets<br>1 = Discard all incoming packets |
| [10] | Rx_dest | R/W | None | 0 = TDM<br>1 = CPU |
| [9:8] | Tx_MPLS_labels_l2tpv3_cookies | R/W | None | For MPLS:<br>00 = Reserved<br>01 = One label in the TX MPLS stack<br>10 = Two labels in the TX MPLS stack<br>11 = Three labels in the TX MPLS stack<br>For L2TPv3: |

**AAL1_Bundle[n]_cfg[127:96] 0x300+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | 00 = No cookies in the TX L2TPv3 header<br>01 = One cookie in the TX L2TPv3 header<br>10 = Two cookies in the TX L2TPv3 header<br>11 = Reserved |
| [7:4] | Port_num | R/W | None | The port number which the bundle is assigned to:<br>0000 = Port 1, 0111=Port 8 |
| [3:2] | Tx_VLAN_stack | R/W | None | 00 = No VLAN tag in header<br>01 = One VLAN tag exists in header<br>10 = Two VLAN tags exist in header<br>11 = Reserved<br>Not valid for Rx. Not used by Tx AAL1 but by Ethernet transmitter block |
| [1] | Rx_bundle_identifier_valid | R/W | None | 0 = Rx_bundle_identifier entry isn't valid: If the incoming frame bundle identifier isn't found in the whole packet classifier table, the incoming frame is handled according to packet classifier discard switches in Packet_classifier_cfg_reg3.<br>1 = Rx_Bundle_Identifier entry is valid |
| [0] | Reserved | R/W | None | Must be set to zero |

**AAL1_Bundle[n]_cfg[159:128] 0x400+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:23] | Reserved | R/W | None | Must be set to zero |
| [22] | Rx_RTP | R/W | None | 0 = RTP header does not exist in received packets<br>1 = RTP header exists in received packets |
| [21:20] | Rx_L2TPV3_cookies | R/W | None | For MPLS:<br>00 = Reserved<br>01 = One label in the received MPLS stack<br>10 = Two label in the received MPLS stack<br>11 = Three label in the received MPLS stack<br>For L2TPv3:<br>00 = No cookies in the received L2TPv3 header<br>01 = One cookie in the received L2TPv3 header<br>10 = Two cookies in the received L2TPv3 header<br>11 = Reserved |
| [19:15] | Reserved | R/W | None | Must be set to zero. |
| [14:10] | Packet_size_in_cells | R/W | None | AAL1 SAR PDUs per frame: 1 - 30 |
| [9:5] | Tx_bundle_identifier | R/W | None | Tx bundle Identifier upper bits<br>Used only for TX_AAL1 old format |
| [4:0] | Reserved | R/W | None | Must be set to zero |

#### *11.4.2.2 HDLC Bundle Configuration*

In the register descriptions below, the index **n** indicates the bundle number: 0 to 63.

**HDLC_Bundle[n]_cfg[31:0] 0x000+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Rx_bundle_identifier | R/W | None | Holds the Rx bundle number |

**HDLC_Bundle[n]_cfg[63:32] 0x100+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:22] | Reserved | R/W | None | Must be set to zero |

**HDLC_Bundle[n]_cfg[63:32] 0x100+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [21:20] | Payload_type_machine | R/W | None | 00 = HDLC<br>01 = AAL1<br>10 = Reserved<br>11 = SAToP/CESoPSN |
| [19] | Tx_RTP | R/W | None | 0 = RTP header does not exist in transmitted packets<br>1 = RTP header exists in transmitted packets |
| [18] | Control_Word_exists | R/W | None | 0 = Control word does not exist<br>1 = Control word exists (default, standard mode) |
| [17:16] | Tx_dest | R/W | None | Destination of packets:<br>00 = Reserved<br>01 = Ethernet<br>10 = CPU<br>11 = Reserved |
| [15:11] | Reserved | R/W | None | Must be set to zero. |
| [10:9] | Packet_SN_mode | R/W | None | Transmitted and expected sequence number is:<br>00 = Always 0<br>01 = Incremented normally in wrap-around manner<br>10 = Reserved<br>11 = Incremented in wrap-around manner but skips 0 |
| [8:3] | Reserved | R/W | None | Must be set to zero. |
| [2:1] | Header_type | R/W | None | 00 = MPLS<br>01 = UDP over IP<br>10 = L2TPv3 over IP<br>11 = MEF |
| [0] | Tx_R_bit | R/W | None | 0 = Don't set R bit in header of transmitted packets<br>1 = Set R bit |

**HDLC_Bundle[n]_cfg[95:64] 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | R/W | None | Must be set to zero |
| [15:13] | Reserved | R/W | None | Must be set to zero |
| [12:2] | Tx_max_frame_size | R/W | None | Tx HDLC maximum transmitted packet size in bytes. This does not include FCS. |
| [1:0] | Reserved | R/W | None | Must be set to zero |

**HDLC_Bundle[n]_cfg[127:96] 0x300+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | Reserved | R/W | None | Must be set to zero |
| [27] | Tx_stop | R/W | None | 0 = Send one packet with the 1st header<br>1 = Stop transmission |
| [26:13] | Reserved | | None | Must be set to zero |
| [12] | OAM_ID_in_CW | R/W | None | 0 = Ignore the OAM packet indication in the control word<br>1 = Check the OAM packet indication in the control word |
| [11] | Rx_discard | R/W | None | 0 = Pass through all incoming packets<br>1 = Discard all incoming packets |
| [10] | Rx_dest | R/W | None | 0 = TDM<br>1 = CPU |
| [9:8] | Tx_MPLS_lables_l2tpv3_cookies | R/W | None | For MPLS:<br>00 = Reserved<br>01 = One label in the TX MPLS stack<br>10 = Two labels in the TX MPLS stack<br>11 = Three labels in the TX MPLS stack<br>For L2TPv3: |

**HDLC_Bundle[n]_cfg[127:96] 0x300+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | 00 = No cookies in the TX L2TPv3 header<br>01 = One cookie in the TX L2TPv3 header<br>10 = Two cookies in the TX L2TPv3 header<br>11 = Reserved |
| [7:4] | Port_num | R/W | None | The port number which the bundle is assigned to:<br>0000 = Port 1, 0111=Port 8 |
| [3:2] | Tx_VLAN_stack | R/W | None | 00 = No VLAN tag in header<br>01 = One VLAN tag exists in header<br>10 = Two VLAN tags exist in header<br>11 = Reserved<br>Not valid for Rx. Not used by Tx AAL1 but by Ethernet MAC transmit block |
| [1] | Rx_Bundle_Identifier_valid | R/W | None | 0 = Rx_bundle_identifier entry isn't valid: If the incoming frame bundle identifier isn't found in the whole packet classifier table, the incoming frame is handled according to discard switches in (Packet_classifier_cfg_reg3)<br>1 = Rx_Bundle_Identifier entry is valid |
| [0] | Reserved | R/W | None | Must be set to zero |

**HDLC_Bundle[n]_cfg[159:128] 0x400+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:22] | Reserved | | 0x000 | Must be set to zero |
| [21:20] | Rx_L2TPV3_cookies | R/W | None | For MPLS:<br>00 = Reserved<br>01 = One label in the received MPLS stack<br>10 = Two label in the received MPLS stack<br>11 = Three label in the received MPLS stack<br>For L2TPv3:<br>00 = No cookies in the received L2TPv3 header<br>01 = One cookie in the received L2TPv3 header<br>10 = Two cookies in the received L2TPv3 header<br>11 = Reserved |
| [19:16] | Reserved | R/W | None | |
| [15:0] | Tx_IP_checksum | R/W | None | IP header checksum for IP total length equal to zero<br>Explain more. Also, why isn't this in AAL1? |

#### *11.4.2.3 SAToP/CESoPSN Bundle Configuration*

In the register descriptions below, the index **n** indicates bundle number: 0 to 63.

**SAToP/CESoPSN_Bundle[n]_cfg[31:0] 0x000+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Rx_bundle_identifier | R/W | None | Holds the Rx bundle number |

**SAToP/CESoPSN_Bundle[n]_cfg[63:32] 0x100+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:22] | Rx_max_buff_size | R/W | None | The size of the jitter buffer. See section 10.6.10. Also the maximum time interval for which data is stored. The resolution is determined by the interface type as follows:<br>For framed E1/T1: 0.5 ms. |

**SAToP/CESoPSN_Bundle[n]_cfg[63:32] 0x100+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | For unframed E1/T1 or serial bundles: 1024 bit periods<br>For high speed interface: 4096 bit periods.<br>Allowed values:<br>For T1-SF: RX_max_buff_size ≤ 2FChex<br>For T1-ESF: RX_max_buff_size ≤ 0x2F9<br>For E1-MF: RX_max_buff_size ≤ 0x3FB<br>For all interface types the RX_max_buff_size must be greater than Rx_PDVT + PCT (Packet Creation Time).<br>Note: For unframed, the RX_max_buff_size resolution is different than the Rx_PDVT resolution. |
| [21:20] | Payload_type_machine | R/W | None | 00 = HDLC<br>01 = AAL1<br>10 = Reserved<br>11 = SAToP/CESoPSN |
| [19] | Tx_RTP | R/W | None | 0 = RTP header does not exist in transmitted packets<br>1 = RTP header exists in transmitted packets |
| [18] | Control_Word_exists | R/W | None | 0 = Control word does not exist<br>1 = Control word exists (default, standard mode) |
| [17:16] | Tx_dest | R/W | None | Destination of packets:<br>01 = Ethernet<br>10 = CPU<br>11 = TDM-Rx (cross-connect)<br>00 = Reserved |
| [15:9] | Rx_max_lost_packets | R/W | None | The maximum number of Rx packets inserted upon detection of lost packets |
| [8:4] | Number_of_ts | R/W | None | One less than number of assigned timeslots per bundle. Not relevant for unstructured bundles, or when working in high speed mode. |
| [3] | Rx_ discard_sanity_fail | R/W | None | 0 = Don't discard the above packets<br>1 = Discard SAToP/CESoPSN packets which fail the sanity check<br>See section 10.6.13.8. |
| [2:1] | Header_type | R/W | None | 00 = MPLS<br>01 = UDP over IP<br>10 = L2TPv3 over IP<br>11 = MEF |
| [0] | Tx_R_bit | R/W | None | 0 = Don't set R bit in header of transmitted packets<br>1 = Set R bit |

**SAToP/CESoPSN_Bundle[n]_cfg[95:64] 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | Reserved | R/W | None | Must be set to zero |
| [30] | Tx_cond_data | R/W | None | 0 = Regular operation<br>1 = Use conditioning octet specified by Tx_cond_octet_type for transmitted packets |
| [29] | Tx_dest_framing | R/W | None | Only applies to T1 framed traffic<br>0 = Destination framer operates in SF framing<br>1 = Destination framer operates in ESF framing |
| [28] | Tx_CAS_source | R/W | None | Source of transmit CAS bits:<br>0 = TDMoP block's RSIG input<br>1 = Tx software CAS table<br>See sections See section 10.6.5 and 11.4.9. |
| [27] | Reserved | R/W | None | Must be set to zero |
| [26:16] | TDM_frames_in_packet or TDM_bytes_in_packet | R/W | None | For structured and structured with CAS CESoPSN bundles: number of TDM frames included in each packet.<br>For SAToP bundles: number of TDM bytes included in each packet. |

**SAToP/CESoPSN_Bundle[n]_cfg[95:64] 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | Note: For Structured with CAS bundles the allowed values are:<br>E1 MF: 16, 8, 4, 2, 1<br>T1 SF/ESF: 24, 12, 8, 6, 4, 3, 2, 1 |
| [15:13] | Reserved | R/W | None | Must be set to zero |
| [12:11] | Tx_SATOP_bundle_type | R/W | None | Bundle type of transmitted payload:<br>00 = Unstructured<br>01 = Structured<br>10 = Structured with CAS<br>11 = Reserved |
| [10:6] | Reserved | R/W | None | Must be set to zero. |
| [5:4] | Tx_cond_octet_type | R/W | None | Selects the ETH_cond_octet from ETH_cond_data_reg to be transmitted towards packet network:<br>00 = ETH_cond_octet_a<br>01 = ETH_cond_octet_b<br>10 = ETH_cond_octet_c<br>11 = ETH_cond_octet_d |
| [3:2] | Rx_SAToP/CESoPSN_ bundle_type | R/W | None | Bundle type of received packets:<br>00 = Unstructured<br>01 = Structured<br>10 = Structured with CAS<br>11 = Reserved |
| [1:0] | Protection_mode | R/W | None | 00 = Stop sending packets<br>01 = Send each packet once with the first header<br>10 = Send each packet once with the second header<br>11 = Send each packet twice: one with the first header and one with the second header |

**SAToP/CESoPSN_Bundle[n]_cfg[127:96] 0x300+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | Reserved | R/W | None | Must be set to zero. |
| [30:16] | Rx_PDVT | R/W | None | Packet delay variation time value for SAToP/CESoPSN bundles. See section 10.6.10. Bits[30:26] are used only when unframed. The resolution is determined by the interface type as follows:<br>For framed E1/T1: 0.5 ms<br>For unframed E1/T1 or serial bundles: 32 bit periods<br>For high speed interface: 128 bit periods<br>Allowed values:<br>Minimum allowed value: 3 (for all interface types)<br>For T1 SF, ESF: Rx_PDVT < 0x300 |
| [15] | Rx_CAS_src | R/W | None | Source of signaling towards TDM:<br>0 = SDRAM signaling jitter buffer<br>1 = Rx SW CAS tables (section 11.4.13) |
| [14] | Rx_enable_reorder | R/W | None | 0 = Disable reorder<br>1 = Enable reorder |
| [13] | Reserved | R/W | None | Must be set to zero |
| [12] | OAM_ID_in_CW | R/W | None | 0 = Ignore the OAM packet indication in the control word<br>1 = Check the OAM packet indication in the control word |
| [11] | Rx_discard | R/W | None | 0 = Pass through all incoming packets<br>1 = Discard all incoming packets |
| [10] | Rx_dest | R/W | None | 0 = TDM<br>1 = CPU |
| [9:8] | Tx_MPLS_lables_l2tpv3_cookies | R/W | None | For MPLS:<br>00 = Reserved<br>01 = One label in the TX MPLS stack<br>10 = Two labels in the TX MPLS stack |

**SAToP/CESoPSN_Bundle[n]_cfg[127:96] 0x300+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | 11 = Three labels in the TX MPLS stack<br>For L2TPv3:<br>00 = No cookies in the TX L2TPv3 header<br>01 = One cookie in the TX L2TPv3 header<br>10 = Two cookies in the TX L2TPv3 header<br>11 = Reserved |
| [7:4] | Port_num | R/W | None | The port number which the bundle is assigned to:<br>0000 = Port 1, 0111=Port 8 |
| [3:2] | Tx_VLAN_stack | | | 00 = No VLAN tag in header<br>01 = One VLAN tag exists in header<br>10 = Two VLAN tags exist in header<br>11 = Reserved<br>Not valid for Rx. Not used by Tx AAL1 but by Ethernet MAC transmitter block |
| [1] | Rx_Bundle_Identifier_valid | R/W | None | 0 = Rx_bundle_identifier entry isn't valid: If the incoming frame bundle identifier isn't found in the whole packet classifier table, the incoming frame is handled according to packet classifier discard switches in Packet_classifier_cfg_reg3<br>1 = Rx_Bundle_Identifier entry is valid |
| [0] | Reserved | R/W | None | Must be set to zero |

**SAToP/CESoPSN_Bundle[n]_cfg[159:128] 0x400+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:24] | Reserved | | 0x00 | Must be set to zero |
| [23] | Last_value_insertion | R/W | None | Enables the insertion of the last received timeslot value in case packet loss was detected. This insertion is only performed if 3 frames or less of data per timeslot is lost. If more than 3 frames of data are lost, the insertion is not performed and, instead, conditioning is inserted as usual).<br>0 = last value insertion disabled<br>1 = last value insertion enabled |
| [22] | Rx_RTP | R/W | None | 0 = RTP header doesn't exist in received packets<br>1 = RTP header exists in received packets |
| [21:20] | Rx_L2TPV3_cookies | R/W | None | For MPLS:<br>00 = Reserved<br>01 = One label in the received MPLS stack<br>10 = Two label in the received MPLS stack<br>11 = Three label in the received MPLS stack<br>For L2TPv3:<br>00 = No cookies in the received L2TPv3 header<br>01 = One cookie in the received L2TPv3 header<br>10 = Two cookies in the received L2TPv3 header<br>11 = Reserved |
| [19:16] | Reserved | R/W | None | Must be set to zero |
| [15:0] | Tx_IP_checksum | R/W | None | IP header checksum for IP total length equal to zero |

### 11.4.3 Counters

Each counter can be read from two different addresses. Reading from the first address—**0x10,000** + offset—does not affect the counter value. Reading from the second address—**0x11,000** + offset—causes the counter to be cleared after it is read.

**Table 11-6. Counters Types**

| Address | Counter Type | Read/Write | Reset Value |
|---|---|---|---|
| 10,000 | Counters – no clear on read | Read Only | None |
| 11,000 | Counters – clear on read | Read Only-Clear on Read | None |

When reading from counters wider than 16 bits in 16-bit mode, use the following procedure:

1. Read from address 2, i.e. H_AD[1]=1. All 32 bits are internally latched and bits 15:0 are output on H_D[15:0].
2. Read from address 0, i.e. H_AD [1]=0. Bits 31:16 are output on H_D[15:0].

#### 11.4.3.1 Per Bundle Counters

In the register descriptions in this section, the index **n** indicates the bundle number: 0 to 63.

**Ethernet Rx Good Packets Counter 0x000+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Good_packets_received | R | None | Good packets received from Ethernet. Counter wraps around to 0 from its maximum value. |

**Ethernet Tx Good Packets Counter 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Good_packets_transmitted | R | None | Good packets transmitted to Ethernet. Counter wraps around to 0 from its maximum value. |

**Ethernet Rx Lost/Jump Event Packets Counter 0x300+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | None | Must be set to zero |
| [15:0] | Lost_AAL1_packets_Rxd / Lost_HDLC_packets_Rxd / Jumped_SAToP/CESoPSN_ packets_Rxd | R | None | Number of lost/jumped packets encountered by RX_AAL1, RX_HDLC or RX_SATOP payload machine:<br><br>AAL1 and SAToP/CESoPSN – The counter is increased by the gap between the received packet sequence number and the expected packet sequence number (except when this gap is higher than the configured Rx_max_lost_packets value).<br><br>HDLC – The counter is increased by the difference between the received packet sequence number and the expected packet sequence number only when this difference is smaller than 32768.<br><br>SAToP/CESoPSN – the CPU can calculate the number of lost packets using the following equation:<br>lost packets = (jumped packets –Rxd reordered packets) |

**Ethernet Rx AAL1 Lost Cells / Rx SAToP/CESoPSN Discarded Packets Counter 0x400+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | None | Must be set to zero |
| [15:0] | Lost_AAL1_Rxd_cells /<br><br>Discarded_SAToP/CESoPSN_ Rxd_packets | R | None | AAL1 – Number of lost AAL1 SAR PDUs<br><br>SAToP/CESoPSN – Number of received packets that were discarded by SAToP/CESoPSN hardware machine. |

**TDM Tx HDLC Frames with Error Counter 0x500+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | None | Must be set to zero |
| [15:0] | TDM_HDLC_err_frames | R | None | Number of HDLC frames from TDM with any error, including CRC/alignment/abort/short/long. Counter sticks at its maximum value and does not roll over to 0. |

**TDM Tx HDLC Good Frames Counter 0x600+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | None | Must be set to zero |
| [31:0] | TDM_HDLC_good_frames | R | None | HDLC good frames received from TDM (passed CRC). Counter wraps around to 0 from its maximum value. |

**TDM Rx SAToP/CESoPSN Reordered Packets / HDLC/AAL1 Packet SN Error Outside Window Counter 0x100+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | SAToP/CESoPSN_Rxd_re_ordered_packets /<br><br>HDLC_packet_sn_oo_window /<br><br>AAL1_packet_sn_oo_window | R | None | SAToP/CESoPSN – Number of received misordered packets that were successfully reordered by SAToP/CESoPSN hardware machine. The counter is incremented each time a miss-ordered packet is received and saved in the SDRAM.<br><br>HDLC – Counter incremented by 1 when SN error outside window is detected (window of 32,768).<br><br>AAL1 – Counter incremented by 1 when SN error outside window is detected (window configured by Rx_max_lost_packets).<br><br>Counter sticks at its maximum value and does not roll over to 0. |

#### 11.4.3.2 Per Jitter Buffer Index Counters

In the register description in this section, the index **n** indicates the jitter buffer number: 0 to 255.

**Jitter Buffer Underrun/Overrun Events Counter 0x800+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | None | Must be set to zero |
| [7:0] | JBC_events | R | None | Number of jitter buffer underrun/overrun events.<br><br>AAL1/SAToP/CESoPSN bundles – count of underrun events. AAL1 counter does not include underruns caused by pointer mismatches.<br><br>HDLC bundles – count of overrun events.<br><br>Counter sticks at its maximum value and does not roll over to 0. |

#### 11.4.3.3 General Counters

**Received Ethernet Bytes Counter 0xE00**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | ETH_bytes_received | R | 0x0000 0000 | Total bytes received from Ethernet (good packets which passed CRC check only). CRC bytes are not counted. Counter wraps around to 0 from its maximum value. |

**Transmitted Ethernet Bytes Counter 0xE04**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | ETH_bytes_transmitted | R | 0x0000 0000 | Total bytes transmitted to Ethernet (good packets which passed CRC check only). CRC bytes are not counted. Counter wraps around to 0 from its maximum value. |

**Classified Packets Counter 0xE08**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Classified_packets | R | 0x0000 0000 | Counts all packets that pass the packet classifier towards TDM or CPU and are not discarded.<br>Counter wraps around to 0 from its maximum value. |

**Received IP Checksum Errors Counter 0xE0C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | 0x0000 | Must be set to zero |
| [15:0] | IP_checksum_err_packets | R | 0x0000 | Counts packets, detected by the packet classifier, as packets with IP checksum errors.<br>Counter sticks at its maximum value and does not roll over to 0. |

### 11.4.4 Status Tables

The TDMoP status tables hold indications of hardware events. Except where noted, these are latched status bits. For each bit, the value 1 indicates that the event occurred. A bit set to 1 maintains its value unless the host CPU changes it. Writing 1 to a bit clears it to 0. Writing 0 to a bit does not change its value. The base address for the TDMoP status tables is **0x12,000**.

#### 11.4.4.1 Per Bundle Status Tables

In the register descriptions in this section, the index **n** indicates the bundle number: 0 to 63.

**Rx Payload Type Machine Status 0x000+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:5] | Reserved | - | None | Must be set to zero |
| [4] | Rx_SAToP/CESoPSN_frame_ count_err | | | SAToP/CESoPSN – packets that belong to structured-with-CAS bundles were received with incorrect number of frames. |
| [3] | Rx_AAL1_cell_hdr_err / Rx_SAToP/CESoPSN_jump_ove rflow_err / | R/W | None | AAL1 – AAL1 SAR PDUs received with incorrect SN (sequence number), protection fields (CRC/parity), corrected and not corrected AAL1 SAR PDU header.<br><br>SAToP/CESoPSN – Packets received with incorrect sequence number (higher than the expected sequence number and within the window allowed by the configured Rx_max_lost_packets value) and could not be inserted into the jitter buffer due to insufficient space. |
| [2] | Rx_AAL1_packet_sn_oo_ window / Rx_HDLC_packet_sn_oo_ window / Rx_SAToP/CESoPSN_packet_ sn_oo_window | R/W | None | HDLC – Packet SN (Sequence Number) error outside window (window of 32768)<br><br>SAToP/CESoPSN/AAL1 – Packets discarded due to incorrect Sequence Number (SN equal to the former or gap between them exceeds limit determined by Rx_max_lost_packets parameter). |
| [1] | Rx_AAL1_packet_sn_in_ window / Rx_HDLC_packet_sn_in_ window / Rx_SAToP/CESoPSN_ overrunn_discard | R/W | None | AAL1– Packet sequence number error within window (determined by Rx_max_lost_packets parameter)<br><br>HDLC – Packet sequence number error within window (window of 32768)<br><br>SAToP/CESoPSN – Packets discarded because the Jitter Buffer reached or was in the over-run state. |
| [0] | Rx_AAL1_ptr_mismatch / Rx_SAToP/CESoPSN_miss_ ordered_discard | R/W | None | AAL1 –SAR PDUs received with pointer mismatch<br><br>SAToP/CESoPSN – Packets discarded because they were considered duplicated, or because they were received too late to be inserted into the Jitter Buffer. |

**Tx Payload Type Machine Status 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:5] | Reserved | - | None | Must be set to zero |
| [4] | Tx_HDLC_abort | R/W | None | HDLC – received frame from TDM with abort indication |
| [3] | Tx_HDLC_short | R/W | None | HDLC – received frame from TDM shorter than 4 bytes (including CRC bytes) |
| [2] | Tx_HDLC_long | R/W | None | HDLC – received frame from TDM longer than maximum allowed length (Tx_max_frame_size) |

**Tx Payload Type Machine Status 0x200+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [1] | Tx_HDLC_align_err | R/W | None | HDLC – received frame from TDM with alignment error |
| [0] | Tx_AAL1_framing_mismatch / Tx_HDLC_CRC_err / Tx_SAToP/CESoPSN_framing_ mismatch | R/W | None | AAL1 – Start of TDM frame or start of TDM multiframe mismatch<br>HDLC – received frame from TDM with CRC error<br>SAToP/CESoPSN – Start of TDM frame or start of TDM multiframe mismatch |

**Tx Buffers Status 0x400+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:1] | Reserved | - | None | Must be set to zero |
| [0] | TDM_to_ETH_buff_err | R/W | None | Frames received from TDM were discarded due to lack of Tx buffers |

**Packet Classifier Status 0x600+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | None | Must be set to zero |
| [7] | Packet_length_error | R/W | None | Packet discarded due to mismatch between IP_length/ Control_word_length (for MPLS/MEF) and the actual length according to the following rules:<br>IP packets – If IP_length > (actual payload + ip_hdr + CW + RTP)<br>MPLS/MEF packets – If Control_word_length > actual payload length + CW + RTP |
| [6] | Rx_sync_loss | RO | None | received packet with “L” indication |
| [5] | Rx_remote_fail | RO | None | received packet with “R” indication |
| [4:3] | Rx_Lbit_modifier | RO | None | received packet with “M” indication |
| [2:1] | Fragmentation_bits | RO | None | Relevant for SAToP/CESoPSN payload type machine:<br>00 = Entire (unfragmented) multi-frame structure is carried in a single packet<br>01 = Packet carrying the first fragment<br>10 = Packet carrying the last fragment<br>11 = Packet carrying an intermediate fragment |
| [0] | Rx_length_mismatch_discard | R/W | None | Packet discarded due to mismatch between the packet length and the configuration (for AAL1 and SAToP/ CESoPSN bundles only) |

#### *11.4.4.2 Per JBC Index Tables*

In the register descriptions in this section, the index **n** indicates the jitter buffer number: 0 to 255.

**Rx JBC Status 0xC00+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:1] | Reserved | - | None | Must be set to zero |
| [0] | JBC_overrun | R/W | None | AAL1 – overrun has occurred<br>HDLC – overrun has occurred<br>SAToP/CESoPSN – overrun has occurred |

### 11.4.5 Timeslot Assignment Tables

Each port has two banks of timeslot assignment (TSA) tables, bank 1 and bank 2. While one bank is actively used by the TDMoP block, the other bank can be written by the CPU. The active bank for the port is specified by the TSA_act_blk field in the Port[n]_cfg_reg register.

The base address for the TDMoP status tables is **0x18,000**. From this base address:

- Bank 1 TSA tables are located at offset **0x000** for ports 1 to 4 and **0x400** for ports 5 to 8.
- Bank 2 TSA tables are located at offset **0x200** for ports 1 to 4 and **0x600** for ports 5 to 8.

In the register descriptions in this section, the index **port** indicates the port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101. The index **ts** is the timeslot number: 0 to 31.

**Bank1 Timeslot Assignment Registers** **Ports 1 to 4: 0x000+(port-1)*0x80+ts*4**
**Ports 5 to 8: 0x400+(port-5)*0x80+ts*4**

**Bank2 Timeslot Assignment Registers** **Ports 1 to 4: 0x200+(port-1)*0x80+ts*4**
**Ports 5 to 8: 0x600+(port-5)*0x80+ts*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:21] | Reserved | - | None | Must be set to zero |
| [20] | Remote_loop | R/W | None | When set, establishes a loop (per timeslot) between the data received from the Ethernet port and the data transmitted towards the Ethernet port.<br><br>Notes:<br>*Usually the remote loop is activated on all timeslots assigned to a bundle.*<br>*Only the TDM data is looped back. CAS information is not looped back.*<br>*Available only when interface is configured to single clock mode (Port[n]_cfg_reg.Two_clocks=0).* |
| [19] | Local_loop | R/W | None | When set, establishes a loop (per timeslot) between the data received from the TDM port and the data transmitted towards the TDM port. The data transmitted towards the TDM port is delayed by one TDM frame vs. the received data.<br><br>Notes:<br>*Usually the local loop is activated on all timeslots assigned to a bundle.*<br>*Only the TDM data is looped back. CAS information is not looped back.*<br>*Available only when interface is configured to single clock* mode (Port[n]_cfg_reg.Two_clocks=0). |
| [18] | Structured_type | R/W | | Must be set for timeslots that are part of AAL1/CESoPSN bundles whose type is structured or structured-with-CAS. |
| [17:16] | Timeslot_width | R/W | None | 00 = Reserved<br>01 =2 bits (only for HDLC bundles)<br>10 = 7 bits (only for HDLC bundles)<br>11 = 8 bits<br>See section 10.6.4 for additional details. |
| [15] | First_in_bundle | R/W | None | Must be set for the first timeslot of an AAL1 or CESoPSN bundle. Must be cleared for HDLC bundles. |
| [14] | Rx_assigned | R/W | None | 0 = timeslot is not assigned for the Rx path<br>1 = timeslot is assigned for the Rx path |
| [13] | Transmit_assigned | R/W | None | 0 = timeslot is not assigned for the transmit path<br>1 = timeslot is assigned for the transmit path |
| [12:7] | Bundle_number | R/W | None | Number of the bundle that the timeslot is assigned to. |
| [6:5] | Reserved | R/W | None | Must be set to zero |
| [4:0] | Jitter_buffer_index | R/W | None | Jitter buffer index. This field indicates which jitter buffer is being used for the timeslot or bundle. It is also the index into the Jitter Buffer Status Table (section 11.4.8).<br>If a timeslot is assigned to a bundle, the jitter buffer index must be configured to the number of the first timeslot assigned to the bundle. Otherwise, it must be configured to the timeslot number. See section 10.6.10. |

### 11.4.6 CPU Queues

The pools and queue referred to in this section are shown in the block diagram in Figure 10-49. Whenever a queue or pool level exceeds the associated threshold register, a latched status bit is set in the CPU_Queues_change register which generates an interrupt unless masked by the associated mask bit in the CPU_Queues_mask register.

In this section the address offsets in parentheses apply when the CPU data bus is 16 bits wide (pin DAT_32_16_N=0). The base address for the TDMoP CPU queues is **0x20,000**.

**Table 11-7. CPU Queues**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x00 (0x02) | TDM_to_CPU_pool_insert | Write to insert a buffer ID into the TDM-to-CPU Pool | 124 |
| 0x04 (0x06) | TDM_to_CPU_pool_level | Number of buffers stored in the TDM-to-CPU Pool | 125 |
| 0x08 (0x0A) | TDM_to_CPU_pool_thresh | TDM-to-CPU Pool interrupt threshold | 125 |
| 0x0C (0x0E) | TDM_to_CPU_q_read | Read to get a buffer ID from the TDM-to-CPU Queue | 125 |
| 0x10 (0x12) | TDM_to_CPU_q_level | Number of buffers in the TDM-to-CPU Queue | 125 |
| 0x14 (0x16) | TDM_to_CPU_q_thresh | TDM-to-CPU Queue interrupt threshold | 125 |
| 0x18 (0x1A) | CPU_to_ETH_q_insert | Write to insert a buffer ID into the CPU-to-ETH Queue | 125 |
| 0x1C (0x1E) | CPU_to_ETH_q_level | Number of buffers in the CPU-to-ETH Queue | 126 |
| 0x20 (0x22) | CPU_to_ETH_q_thresh | CPU-to-ETH Queue interrupt threshold | 126 |
| 0x24 (0x26) | ETH_to_CPU_pool_insert | Write to insert a buffer ID into the ETH-to-CPU Pool | 126 |
| 0x28 (0x2A) | ETH_to_CPU_pool_level | Number of buffers stored in the ETH-to-CPU Pool | 126 |
| 0x2C (0x2E) | ETH_to_CPU_pool_thresh | ETH-to-CPU Queue interrupt threshold. | 126 |
| 0x30 (0x32) | ETH_to_CPU_q_read | Read to get a buffer ID from the ETH-to-CPU Queue | 127 |
| 0x34 (0x36) | ETH_to_CPU_q_level | Number of buffers in the ETH-to-CPU Queue. | 127 |
| 0x38 (0x3A) | ETH_to_CPU_q_thresh | ETH-to-CPU Queue interrupt threshold | 127 |
| 0x54 (0x56) | CPU_to_TDM_q_insert | Write to insert a buffer ID into the CPU-to-TDM Queue | 127 |
| 0x58 (0x5A) | CPU_to_TDM_q_level | Number of buffers stored in the CPU-to-TDM Queue | 127 |
| 0x5C (0x5E) | CPU_to_TDM_q_thresh | CPU-to-TDM Queue interrupt threshold | 127 |
| 0x60 (0x62) | Tx_return_q_read | Read to get a buffer ID from the CPU-tx-return Queue | 128 |
| 0x64 (0x66) | Tx_return_q_level | Number of buffers stored in the CPU-tx-return Queue | 128 |
| 0x68 (0x6A) | Tx_return_q_thresh | CPU-tx-return Queue interrupt threshold | 128 |
| 0x6C (0x6E) | Rx_return_q_read | Read to get a buffer ID from the CPU-rx-return Queue | 128 |
| 0x70 (0x72) | Rx_return_q_level | Number of buffers stored in the CPU-rx-return Queue | 129 |
| 0x74 (0x76) | Rx_return_q_thresh | CPU-rx-return Queue interrupt threshold | 129 |

#### 11.4.6.1 TDM-to-CPU Pool

**TDM_to_CPU_pool_insert 0x00 (0x02)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | WO | None | Writing to this address causes a single 13-bit buffer ID to be inserted to the TDM-to-CPU pool. Only bits [12:0] are written. The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of the 24 SDRAM address bits). |

**TDM_to_CPU_pool_level 0x04 (0x06)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Level | RO | 0x0 | Number of buffers currently stored in the pool. These are the buffers that are still available to the Tx payload type machines. Range: 0 to 128. |

**TDM_to_CPU_pool_thresh 0x08 (0x0A)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Threshold | RO | 0x0 | If the number of buffers in the pool is ≤ this threshold, an interrupt is generated. Range: 0 to 128. |

#### *11.4.6.2 TDM-to-CPU Queue*

**TDM_to_CPU_q_read 0x0C (0x0E)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | RO | None | Reading from this address extracts the first buffer ID from the TDM-to-CPU queue (bits [12:0]). The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of 24 SDRAM address bits). |

**TDM_to_CPU_q_level 0x10 (0x12)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Level | RO | 0x0 | Number of buffers currently stored in the queue. These are the buffers still waiting to be handled by the CPU. Range: 0 to 128. |

**TDM_to_CPU_q_thresh 0x14 (0x16)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Threshold | RO | 0x0 | If the number of buffers in the queue is ≥ this threshold, an interrupt is generated. Range: 0-128 |

#### *11.4.6.3 CPU-to-ETH Queue*

**CPU_to_ETH_q_insert 0x18 (0x1A)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | WO | None | Writing to this address causes a single 13-bit buffer ID to be inserted to the CPU-to-ETH queue. Only bits [12:0] are written. The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of the 24 SDRAM address bits). |

**CPU_to_ETH_q_level 0x1C (0x1E)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Level | RO | 0x0 | Number of buffers currently stored in the queue. Range: 0 to 32. |

**CPU_to_ETH_q_thresh 0x20 (0x22)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Threshold | RO | 0x0 | If the number of buffers in the queue is ≤ this threshold, an interrupt is generated. Range: 0 to 32. |

#### *11.4.6.4 ETH-to-CPU Pool*

**ETH_to_CPU_pool_insert 0x24 (0x26)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | WO | None | Writing to this address causes a single 13-bit buffer ID to be inserted to the ETH-to-CPU pool. Only bits [12:0] are written. The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of the 24 SDRAM address bits). |

**ETH_to_CPU_pool_level 0x28 (0x2A)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Level | RO | 0x0 | Number of buffers currently stored in the pool. These are the buffers that are still available to the Rx arbiter. Range: 0 to 128. |

**ETH_to_CPU_pool_thresh 0x2C (0x2E)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Threshold | RO | 0x0 | If the number of buffers in the pool is ≤ this threshold, an interrupt is generated and only OAM packets are inserted in the ETH-to-CPU queue (non-OAM packets are discarded). Range: 0 to 128. |

#### 11.4.6.5 ETH- to-CPU Queue

**ETH_to_CPU_q_read 0x30 (0x32)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | RO | None | Reading from this address extracts the first buffer ID from the ETH-to-CPU queue (bits [12:0]). The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of 24 SDRAM address bits). |

**ETH_to_CPU_q_level 0x34 (0x36)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Level | RO | 0x0 | Number of buffers currently stored in the queue. These are the buffers still waiting to be handled by the CPU. Range: 0 to 128. |

**ETH_to_CPU_q_thresh 0x38 (0x3A)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Threshold | RO | 0x0 | If the number of buffers in the queue is ≥ this threshold, an interrupt is generated. Range: 0 to 128. |

#### 11.4.6.6 CPU-to-TDM Queue

**CPU_to_TDM_q_insert 0x54 (0x56)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | WO | None | Writing to this address causes a single 13-bit buffer ID to be inserted to the CPU-to-TDM queue. Only bits [12:0] are written. The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of the 24 SDRAM address bits). |

**CPU_to_TDM_q_level 0x58 (0x5A)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Level | RO | 0x0 | Number of buffers currently stored in the queue. Range: 0 to 32. |

**CPU_to_TDM_q_thresh 0x5C (0x5E)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Threshold | RO | 0x0 | If the number of buffers in the queue is ≥ this threshold, |

**CPU_to_TDM_q_thresh 0x5C (0x5E)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | an interrupt is generated. Range: 0 to 32. |

#### 11.4.6.7 Tx Return Queue

**Tx_return_q_read 0x60 (0x62)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | RO | None | Reading from this address extracts the first buffer ID from the CPU Tx return queue (bits [12:0]). The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of 24 SDRAM address bits). |

**Tx_return_q_level 0x64 (0x62)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Level | RO | 0x0 | Number of buffers currently stored in the queue. Range: 0 to 32. |

**Tx_return_q_thresh 0x68 (0x6A)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Threshold | RO | 0x0 | If the number of buffers in the queue is ≥ this threshold, an interrupt is generated. Range: 0 to 32. |

#### 11.4.6.8 Rx Return Queue

**Rx_return_q_read 0x6C (0x6E)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved | - | 0x0 | Must be set to zero |
| [12:0] | Buffer ID | RO | None | Reading from this address extracts the first buffer ID from the CPU Rx return queue (bits [12:0]). The buffer ID serves as the 13 MSbs of the buffer address in the SDRAM (i.e. corresponds to H_AD[23:11] out of 24 SDRAM address bits). |

**Rx_return_q_level 0x70 (0x72)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Level | RO | 0x0 | Number of buffers currently stored in the queue. Range: 0 to 32. |

**Rx_return_q_thresh 0x74 (0x76)**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5:0] | Threshold | RO | 0x0 | If the number of buffers in the queue is ≥ this threshold, an interrupt is generated. Range: 0 to 32. |

### 11.4.7 Transmit Buffers Pool

The base address for the TDMoP transmit buffers pool is **0x28,000**. See section 10.6.11.7 for details.

#### 11.4.7.1 Per-Bundle Head Pointers

In the register descriptions in this section, the index **n** indicates the bundle number: 0 to 63.

The RAM should be initialized by CPU software to hold the heads of the linked lists for all open bundles. See section 10.6.11.7.

**Per-Bundle Head[n] 0x800+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:10] | Reserved | | None | Must be set to zero |
| [9] | Buffer_valid | R/W | None | 0 = The head contains non-valid information (i.e. the pool is empty).<br>1 = The head points to a valid free buffer. |
| [8:0] | Buffer_id | R/W | None | The full address of the buffer consists of the Tx buffer base address (specified in General_cfg_reg1. Tx_buf_base_add) concatenated with the buffer ID and eleven 0s. |

#### 11.4.7.2 Per-Buffer Next-Buffer Pointers

A pointer to the next buffer in the linked list.

In the register descriptions in this section, the index **n** indicates the buffer number: 0 to 511.

The RAM should be initialized by CPU software to hold the linked lists for all the bundles. See section 10.6.11.7.

**Per Buffer Next Buffer[n] 0x000+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:9] | Reserved | - | None | Must be set to zero |
| [8:0] | Buffer_offset | R/W | None | The offset (ID) of the next buffer in the linked list in the SDRAM area dedicated to the Tx payload-type machines. The full address of the buffer consists of the Tx buffer base address (specified in General_cfg_reg1. Tx_buf_base_add) concatenated with the buffer offset and eleven 0s. |

### 11.4.8 Jitter Buffer Control

The base address for the TDMoP jitter buffer control is **0x30,000**.

In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34T108, 1-4 for DS34T104, 1-2 for DS34T102, 1 only for DS34T101. The index **ts** indicates timeslot number: 0 to 31. The index **n** indicates the bundle number: 0 to 63. See section 10.6.10 for more information.

**Table 11-8. Jitter Buffer Status Tables**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x000 | Status_and_level[1, 0] | Jitter buffer port 1 timeslot 0 status and fill level | 130 |
| 0x004 | Min_and_max_level[1, 0] | Jitter buffer port 1 timeslot 0 min / max levels | 131 |
| (port-1)*0x100+ts*8 | Status_and_level[port, ts] | Jitter buffer status and fill level | 130 |
| (port-1)*0x100+ts*8+4 | Min_and_max_level[port, ts] | Jitter buffer min / max levels | 131 |
| 0x7F8 | Status_and_level[8, 31] | Jitter buffer port 8 timeslot 31 status and fill level | 130 |
| 0x7FC | Min_and_max_level[8, 31] | Jitter buffer port 8 timeslot 31 min / max levels | 131 |

Note 1: In high speed mode, Hs_status_and_level and Hs_min_and_max_level reside in Status_and_level0 and Min_and_max_level0 registers, respectively.

Note 2: The CPU should never try to read Min_and_max_level from an HDLC bundle. When the CPU performs an access to these registers, it causes some bits to be changed – bits that are used for other purposes in HDLC bundles and thus may cause severe problems.

**Table 11-9. Bundle Timeslot Tables**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0xF00 | Bundle_ts0 | Assigned timeslots in bundle 0 | 130 |
| 0xF00+n*4 | Bundle_ts[n] | Assigned timeslots in bundle n | 130 |
| 0xFFC | Bundle_ts63 | Assigned timeslots in bundle 63 | 130 |

#### 11.4.8.1 Status_and_level Registers

The status_and_level registers have different fields depending on the bundle type: HDLC, Structured AAL1/CESoPSN, Unstructured AAL1/SAToP or High Speed AAL1/SAToP. The subsections below describe the status_and_level register fields for each type. In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34T108, 1-4 for DS34T104, 1-2 for DS34T102, 1 only for DS34T101. The index **ts** indicates timeslot number: 0 to 31.

##### 11.4.8.1.1 HDLC

**Status_and_level (port-1)*0x100+ts*8**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:2] | Reserved | RO | 0x0 | Always zero |
| [1:0] | Status | RO | None | The status of the bundle's jitter buffer:<br>00 = jitter buffer is empty<br>01 = jitter buffer is OK<br>10 = jitter buffer is full<br>11 = Reserved |

#### 11.4.8.1.2 Structured AAL1/CESoPSN

**Status_and_level (port-1)*0x100+ts*8**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:26] | Reserved | RO | 0x0 | Always zero |
| [25:16] | Current_level | RO | None | The current jitter buffer level for the bundle. The resolution is 0.5ms. |
| [15:2] | Reserved | RO | 0x0 | Always zero |
| [1:0] | Status | RO | None | The status of the bundle's jitter buffer:<br>00 = jitter buffer is empty<br>01 = jitter buffer is OK<br>10 = jitter buffer is full<br>11 = Reserved |

#### 11.4.8.1.3 Unstructured AAL1/SAToP

**Status_and_level (port-1)*0x100**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | Reserved | RO | 0x0 | Always zero |
| [30:16] | Current_level | RO | None | The current jitter buffer level for the bundle.<br>The resolution is 32 interface bit periods. |
| [15:2] | Reserved | RO | 0x0 | Always zero |
| [1:0] | Status | RO | None | The status of the bundle's jitter buffer:<br>00 = jitter buffer is empty<br>01 = jitter buffer is OK<br>10 = jitter buffer is full<br>11 = Reserved |

#### 11.4.8.1.4 High Speed AAL1/SAToP

**Status_and_level 0x000**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Current_level | RO | 0x0000 | The 16 MSbs of the current jitter buffer level (the level is 17 bits wide). The resolution is 64 interface bit periods. |
| [15:2] | Reserved | RO | 0x0 | Always zero |
| [1:0] | Status | RO | 0x0 | The status of the bundle's jitter buffer:<br>00 = jitter buffer is empty<br>01 = jitter buffer is OK<br>10 = jitter buffer is full<br>11 = Reserved |

### 11.4.8.2 Min_and_max_level

In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34T108, 1-4 for DS34T104, 1-2 for DS34T102, 1 only for DS34T101. The index **ts** indicates timeslot number: 0 to 31..

#### 11.4.8.2.1 Structured AAL1/CESoPSN

**Min_and_max_level (port-1)*0x100+ts*8+4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:26] | Reserved | RO | 0x0 | Always zero |
| [25:16] | Minimal_level | RO | None | The minimal level that the jitter buffer has reached since the last time this register was read. After this register is read the TDMoP block resets this field to all ones. When underrun is reached, the value of this field remains zero until it is read by the CPU. The resolution is 0.5 ms.. |

**Min_and_max_level (port-1)*0x100+ts*8+4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [15:10] | Reserved | RO | 0x00 | These bits are always zero |
| [9:0] | Maximal_level | RO | None | The maximal level that the jitter buffer has reached since the last time this register was read. After this register is read the TDMoP block resets this field to zero. When overrun is reached, the value remains equal to Rx_max_buff_size until it is read by the CPU. The resolution is 0.5 ms. |

#### *11.4.8.2.2 Unstructured AAL1/SAToP*

**Min_and_max_level (port-1)*0x100+4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31] | Reserved | RO | 0x0 | This bit is always zero |
| [30:16] | Minimal_level | RO | None | The minimal level that the jitter buffer has reached since the last time this register was read. After this register is read the TDMoP block resets this field to all ones. When underrun is reached, the value of this field remains zero until it is read by the CPU. The resolution is 32 interface bit periods. |
| [15] | Reserved | RO | 0x0 | This bit is always zero |
| [14:0] | Maximal_level | RO | None | The maximal level that the jitter buffer has reached since the last time this register was read. After this register is read the TDMoP block resets this field to zero. When overrun is reached, the value remains equal to Rx_max_buff_size until it is read by the CPU. The resolution is 32 interface bit periods. |

#### *11.4.8.2.3 High Speed AAL1/SAToP*

**Min_and_max_level 0x004**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Minimal_level | RO | 0xFFFF | The 16 MSbs of the minimal level that the jitter buffer has reached since the last time this register was read. After this register is read the TDMoP block resets this field to all ones. When underrun is reached, the value of this field remains zero until it is read by the CPU. The level is 17 bits wide. The resolution is 64 interface bit periods. |
| [15:0] | Maximal_level | RO | 0x0000 | The 16 MSbs of the maximal level that the jitter buffer has reached since the last time this register was read. After this register is read the TDMoP block resets this field to zero. When overrun is reached, the value remains equal to Rx_max_buff_size until it is read by the CPU. The level is 17 bits wide. The resolution is 64 interface bit periods. |

### *11.4.8.3 Bundle Timeslot Registers*

In this section, the index **n** indicates the bundle number: 0 to 63.

**Bundle_ts[n] 0xF00+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Ts_assigned | R/W | None | Assigned timeslots of the bundle. See section 10.6.10.<br>1 = Timeslot is assigned to the bundle<br>0 = Timeslot is not assigned to the bundle |

**Bundle_ts[n] 0xF00+n*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | Note: When the interface type is Nx64k this field should be set to all 1s. |

### 11.4.9 Transmit Software CAS

The base address for the TDMoP transmit software CAS register space is **0x38,000**. For the CAS information transmitted in packets in the TDM-to-Ethernet direction, the CAS signaling information stored in these registers can be used instead of CAS bits coming into the TDMoP block on the TDMn_RSIG_RTS signals. This is configured on a per-bundle basis using the Tx_CAS_source field in the Bundle Configuration Tables. In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101.

**Table 11-10. Transmit Software CAS Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| **Port 1** | | | |
| 0x00 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 1 | 135 |
| 0x04 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 1 | 135 |
| 0x08 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 1 | 135 |
| 0x0C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 1 | 135 |
| **Port 2** | | | |
| 0x10 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 2 | 135 |
| 0x14 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 2 | 135 |
| 0x18 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 2 | 135 |
| 0x1C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 2 | 135 |
| **Port 3** | | | |
| 0x20 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 3 | 135 |
| 0x24 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 3 | 135 |
| 0x28 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 3 | 135 |
| 0x2C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 3 | 135 |
| **Port 4** | | | |
| 0x30 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 4 | 135 |
| 0x34 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 4 | 135 |
| 0x38 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 4 | 135 |
| 0x3C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 4 | 135 |
| **Port 5** | | | |
| 0x40 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 5 | 135 |
| 0x44 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 5 | 135 |
| 0x48 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 5 | 135 |
| 0x4C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 5 | 135 |
| **Port 6** | | | |
| 0x50 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 6 | 135 |
| 0x54 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 6 | 135 |
| 0x58 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 6 | 135 |
| 0x5C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 6 | 135 |
| **Port 7** | | | |
| 0x60 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 7 | 135 |
| 0x64 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 7 | 135 |
| 0x68 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 7 | 135 |
| 0x6C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 7 | 135 |
| **Port 8** | | | |
| 0x70 | Tx_SW_CAS_TS7_TS0 | CAS signaling for TS7 to TS0 for Port 8 | 135 |
| 0x74 | Tx_SW_CAS_TS15_TS8 | CAS signaling for TS15 to TS8 for Port 8 | 135 |
| 0x78 | Tx_SW_CAS_TS23_TS16 | CAS signaling for TS23 to TS16 for Port 8 | 135 |
| 0x7C | Tx_SW_CAS_TS31_TS24 | CAS signaling for TS31 to TS24 for Port 8 | 135 |

**Tx_SW_CAS_TS7_TS0 0x000+(port-1)*0x10**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | TS7_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 7 |
| [27:24] | TS6_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 6 |
| [23:20] | TS5_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 5 |
| [19:16] | TS4_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 4 |
| [15:12] | TS3_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 3 |
| [11:8] | TS2_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 2 |
| [7:4] | TS1_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 1 |
| [3:0] | TS0_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 0 |

**Tx_SW_CAS_TS15_TS8 0x004+(port-1)*0x10**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | TS15_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 15 |
| [27:24] | TS14_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 14 |
| [23:20] | TS13_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 13 |
| [19:16] | TS12_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 12 |
| [15:12] | TS11_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 11 |
| [11:8] | TS10_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 10 |
| [7:4] | TS9_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 9 |
| [3:0] | TS8_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 8 |

**Tx_SW_CAS_TS23_TS16 0x008+(port-1)*0x10**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | TS23_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 23 |
| [27:24] | TS22_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 22 |
| [23:20] | TS21_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 21 |
| [19:16] | TS20_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 20 |
| [15:12] | TS19_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 19 |
| [11:8] | TS18_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 18 |
| [7:4] | TS17_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 17 |
| [3:0] | TS16_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 16 |

**Tx_SW_CAS_TS31_TS24 0x00C+(port-1)*0x10**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | TS31_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 31 |
| [27:24] | TS30_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 30 |
| [23:20] | TS29_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 29 |
| [19:16] | TS28_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 28 |
| [15:12] | TS27_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 27 |
| [11:8] | TS26_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 26 |
| [7:4] | TS25_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 25 |
| [3:0] | TS24_CAS_nibble | R/W | None | CAS signaling (ABCD) for timeslot 24 |

### 11.4.10 Receive Line CAS

The base address for the TDMoP Rx line CAS register space is **0x40,000**. These read-only registers allow the CPU to examine the state of the CAS signaling recovered from received packets and transmitted out of the TDMoP block on the TDMn_TSIG signals. See section 10.6.5.2 for more details. When Rx line CAS bits change, an interrupt is generated. The Rx_CAS_change registers in the Interrupt Controller indicate which timeslots have changed CAS bits.

In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101. The index **ts** indicates timeslot number: 0 to 31.

**Table 11-11. Receive Line CAS Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| **Port 1** | | | |
| 0x000 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 1 | 136 |
| 0x000+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 1 | 136 |
| 0x07C | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 1 | 136 |
| **Port 2** | | | |
| 0x080 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 2 | 136 |
| 0x080+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 2 | 136 |
| 0x0FC | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 2 | 136 |
| **Port 3** | | | |
| 0x100 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 3 | 136 |
| 0x100+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 3 | 136 |
| 0x17C | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 3 | 136 |
| **Port 4** | | | |
| 0x180 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 4 | 136 |
| 0x180+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 4 | 136 |
| 0x1FC | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 4 | 136 |
| **Port 5** | | | |
| 0x200 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 5 | 136 |
| 0x200+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 5 | 136 |
| 0x27C | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 5 | 136 |
| **Port 6** | | | |
| 0x280 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 6 | 136 |
| 0x280+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 6 | 136 |
| 0x2FC | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 6 | 136 |
| **Port 7** | | | |
| 0x300 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 7 | 136 |
| 0x300+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 7 | 136 |
| 0x37C | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 7 | 136 |
| **Port 8** | | | |
| 0x380 | Rx_Line_CAS_TS0 | CAS signaling for timeslot 0 for Port 8 | 136 |
| 0x380+ts*4 | Rx_Line_CAS_TS[ts] | CAS signaling for timeslot ts for Port 8 | 136 |
| 0x3FC | Rx_Line_CAS_TS31 | CAS signaling for timeslot 31 for Port 8 | 136 |

**Rx_Line_CAS 0x000+(port-1)*0x80+ts*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:4] | Reserved | - | 0x0 | Must be set to zero |
| [3:0] | Rx_CAS | RO | None | CAS signaling (ABCD) towards TDMn_TSIG |

### 11.4.11 Clock Recovery

The base address for the TDMoP clock recovery register space is **0x48,000**. Most of the registers in this section of the TDMoP block are not documented. The HAL (Hardware Abstraction Layer) software manages these registers.

In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101.

**Table 11-12. Clock Recovery Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| **Port 1** | | | |
| 0x0000 | Control_Word_P1 | Port1 clock recovery control bits | 137 |
| 0x0004-00A0 | Clk_recovery_cfg_reg1-40 | Port1 clock recovery configuration registers (not documented) | --- |
| **Port 2** | | | |
| 0x0400 | Control_Word_P2 | Port2 clock recovery control bits | 137 |
| 0x0404-04A0 | Clk_recovery_cfg_reg1-40 | Port2 clock recovery configuration registers (not documented) | --- |
| **Port 3** | | | |
| 0x0800 | Control_Word_P3 | Port3 clock recovery control bits | 137 |
| 0x0804-08A0 | Clk_recovery_cfg_reg1-40 | Port3 clock recovery configuration registers (not documented) | --- |
| **Port 4** | | | |
| 0x0C00 | Control_Word_P4 | Port4 clock recovery control bits | 137 |
| 0x0C04-0CA0 | Clk_recovery_cfg_reg1-40 | Port4 clock recovery configuration registers (not documented) | --- |
| **Port 5** | | | |
| 0x1000 | Control_Word_P5 | Port5 clock recovery control bits | 137 |
| 0x1004-10A0 | Clk_recovery_cfg_reg1-40 | Port5 clock recovery configuration registers (not documented) | --- |
| **Port 6** | | | |
| 0x1400 | Control_Word_P6 | Port6 clock recovery control bits | 137 |
| 0x1404-14A0 | Clk_recovery_cfg_reg1-40 | Port6 clock recovery configuration registers (not documented) | --- |
| **Port 7** | | | |
| 0x1800 | Control_Word_P7 | Port7 clock recovery control bits | 137 |
| 0x1804-18A0 | Clk_recovery_cfg_reg1-40 | Port7 clock recovery configuration registers (not documented) | --- |
| **Port 8** | | | |
| 0x1C00 | Control_Word_P8 | Port8 clock recovery control bits | 137 |
| 0x1C04-1CA0 | Clk_recovery_cfg_reg1-40 | Port8 clock recovery configuration registers (not documented) | --- |

When using the clock recovery mechanism of a certain port, its Rx_PDVT parameter in the bundle configuration must also be configured.

**Clk_Recovery_Control_Word 0x000+(port-1)*0x400**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:1] | Reserved | - | 0x0 | Set according to the HAL function |
| [0] | System_Reset | W/O | 0x0 | 1 = Reset the clock recovery system |

### 11.4.12 Receive SW Conditioning Octet Select

The base address for the TDMoP Rx software conditioning octet select register space is **0x50,000**. These registers specify which of four conditioning bytes (TDM_cond_octet_a through TDM_cond_octet_d in TDM_cond_data_reg) the TDMoP block transmits on the TDMn_TX signals during an unassigned timeslot. The specified value is also the conditioning octet that is inserted into the jitter buffer for lost packet compensation.

In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101. The index **ts** indicates timeslot number: 0 to 31.

**Table 11-13. Receive SW Conditioning Octet Select Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| **Port 1** | | | |
| 0x000 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 1 | 138 |
| 0x000+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 1 | 138 |
| 0x07C | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 1 | 138 |
| **Port 2** | | | |
| 0x080 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 2 | 138 |
| 0x080+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 2 | 138 |
| 0x0FC | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 2 | 138 |
| **Port 3** | | | |
| 0x100 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 3 | 138 |
| 0x100+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 3 | 138 |
| 0x17C | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 3 | 138 |
| **Port 4** | | | |
| 0x180 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 4 | 138 |
| 0x180+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 4 | 138 |
| 0x1FC | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 4 | 138 |
| **Port 5** | | | |
| 0x200 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 5 | 138 |
| 0x200+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 5 | 138 |
| 0x27C | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 5 | 138 |
| **Port 6** | | | |
| 0x280 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 6 | 138 |
| 0x280+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 6 | 138 |
| 0x2FC | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 6 | 138 |
| **Port 7** | | | |
| 0x300 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 7 | 138 |
| 0x300+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 7 | 138 |
| 0x37C | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 7 | 138 |
| **Port 8** | | | |
| 0x380 | Rx_SW_cond_TS0 | Rx software conditioning for timeslot 0 for Port 8 | 138 |
| 0x380+ts*4 | Rx_SW_cond_TS[ts] | Rx software conditioning for timeslot ts for Port 8 | 138 |
| 0x3FC | Rx_SW_cond_TS31 | Rx software conditioning for timeslot 31 for Port 8 | 138 |

**Rx_SW_cond 0x000+(port-1)*0x80+ts*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:2] | Reserved | - | 0x0 | Must be set to zero |
| [1:0] | Cond_octet_sel | R/W | None | 00 = TDM_cond_octet_a<br>01 = TDM_cond_octet_b<br>10 = TDM_cond_octet_c<br>11 = TDM_cond_octet_d |

### 11.4.13 Receive SW CAS

The base address for the TDMoP Rx software CAS register space is **0x58,000**. These registers specify the CAS signaling bits the TDMoP block transmits on the TDMn_TSIG signals during unassigned timeslots and during timeslots where CAS is not assigned. See section 10.6.5.2 for more details.

In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101. The index **ts** indicates timeslot number: 0 to 31.

**Table 11-14. Receive SW CAS Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| **Port 1** | | | |
| 0x000 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 1 | 139 |
| 0x000+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 1 | 139 |
| 0x07C | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 1 | 139 |
| **Port 2** | | | |
| 0x080 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 2 | 139 |
| 0x080+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 2 | 139 |
| 0x0FC | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 2 | 139 |
| **Port 3** | | | |
| 0x100 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 3 | 139 |
| 0x100+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 3 | 139 |
| 0x17C | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 3 | 139 |
| **Port 4** | | | |
| 0x180 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 4 | 139 |
| 0x180+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 4 | 139 |
| 0x1FC | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 4 | 139 |
| **Port 5** | | | |
| 0x200 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 5 | 139 |
| 0x200+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 5 | 139 |
| 0x27C | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 5 | 139 |
| **Port 6** | | | |
| 0x280 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 6 | 139 |
| 0x280+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 6 | 139 |
| 0x2FC | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 6 | 139 |
| **Port 7** | | | |
| 0x300 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 7 | 139 |
| 0x300+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 7 | 139 |
| 0x37C | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 7 | 139 |
| **Port 8** | | | |
| 0x380 | Rx_SW_CAS_TS0 | Rx software conditioning for timeslot 0 for Port 8 | 139 |
| 0x380+ts*4 | Rx_SW_CAS_TS[ts] | Rx software conditioning for timeslot ts for Port 8 | 139 |
| 0x3FC | Rx_SW_CAS_TS31 | Rx software conditioning for timeslot 31 for Port 8 | 139 |

**Rx_SW_CAS 0x000+(port-1)*0x80+ts*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:4] | Reserved | - | 0x0 | Must be set to zero |
| [3:0] | Rx_CAS | R/W | None | CAS signaling (ABCD) transmitted towards TDMn_TSIG when Rx_CAS_src=1 in Bundle Configuration Tables. Must be different from 0000. |

### 11.4.14 Interrupt Controller

The base address for the interrupt controller register space is **0x68,000**.

The Intpend register and the "change" registers listed below have latched status bits that indicate various TDMoP hardware events. For each bit, the value 1 indicates that the event occurred. Writing 1 to a bit clears it to 0. Writing 0 to a bit does not change its value.

The Intmask register and the other "mask" registers listed below have an interrupt mask bit corresponding to each bit in the associated "change" register. Each mask bit masks the interrupt when set to 1 and does not mask the interrupt when set to 0.

The Intpend register is the master interrupt status register. "Change" bits in Intpend indicate that one or more events of a specific type have occurred. More details about which ports or bundles had that type of event can be found by reading the change register(s) for that event type.

In the register descriptions in this section, the index **port** indicates port number: 1-8 for DS34S108, 1-4 for DS34S104, 1-2 for DS34S102, 1 only for DS34S101.

**Table 11-15. Interrupt Controller Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x000 | Intpend | Interrupts pending register | 141 |
| 0x004 | Intmask | Interrupt mask register | 142 |
| 0x040 | Rx_CAS_change_P1 | Rx CAS change for timeslots in Port 1 | 143 |
| 0x044 | Rx_CAS_change_P2 | Rx CAS change for timeslots in Port 2 | 143 |
| 0x048 | Rx_CAS_change_P3 | Rx CAS change for timeslots in Port 3 | 143 |
| 0x04C | Rx_CAS_change_P4 | Rx CAS change for timeslots in Port 4 | 143 |
| 0x050 | Rx_CAS_change_P5 | Rx CAS change for timeslots in Port 5 | 143 |
| 0x054 | Rx_CAS_change_P6 | Rx CAS change for timeslots in Port 6 | 143 |
| 0x058 | Rx_CAS_change_P7 | Rx CAS change for timeslots in Port 7 | 143 |
| 0x05C | Rx_CAS_change_P8 | Rx CAS change for timeslots in Port 8 | 143 |
| 0x080 | JBC_underrun_P1 | JBC underrun in Port 1. | 143 |
| 0x088 | JBC_underrun_P2 | JBC underrun in Port 2 | 143 |
| 0x090 | JBC_underrun_P3 | JBC underrun in Port 3 | 143 |
| 0x098 | JBC_underrun_P4 | JBC underrun in Port 4 | 143 |
| 0x0A0 | JBC_underrun_P5 | JBC underrun in Port 5 | 143 |
| 0x0A8 | JBC_underrun_P6 | JBC underrun in Port 6 | 143 |
| 0x0B0 | JBC_underrun_P7 | JBC underrun in Port 7 | 143 |
| 0x0B8 | JBC_underrun_P8 | JBC underrun in Port 8 | 143 |
| 0x084 | JBC_underrun_mask_P1 | JBC underrun mask for Port 1 | 143 |
| 0x08C | JBC_underrun_mask_P2 | JBC underrun mask for Port 2 | 143 |
| 0x094 | JBC_underrun_mask_P3 | JBC underrun mask for Port 3 | 143 |
| 0x09C | JBC_underrun_mask_P4 | JBC underrun mask for Port 4 | 143 |
| 0x0A4 | JBC_underrun_mask_P5 | JBC underrun mask for Port 5 | 143 |
| 0x0AC | JBC_underrun_mask_P6 | JBC underrun mask for Port 6 | 143 |
| 0x0B4 | JBC_underrun_mask_P7 | JBC underrun mask for Port 7 | 143 |
| 0x0BC | JBC_underrun_mask_P8 | JBC underrun mask for Port 8 | 143 |
| 0x0C0 | Tx_CAS_change_P1 | Tx CAS change for timeslots in Port 1 | 144 |
| 0x0C8 | Tx_CAS_change_P2 | Tx CAS change for timeslots in Port 2 | 144 |
| 0x0D0 | Tx_CAS_change_P3 | Tx CAS change for timeslots in Port 3 | 144 |
| 0x0D8 | Tx_CAS_change_P4 | Tx CAS change for timeslots in Port 4 | 144 |
| 0x0E0 | Tx_CAS_change_P5 | Tx CAS change for timeslots in Port 5 | 144 |
| 0x0E8 | Tx_CAS_change_P6 | Tx CAS change for timeslots in Port 6 | 144 |
| 0x0F0 | Tx_CAS_change_P7 | Tx CAS change for timeslots in Port 7 | 144 |
| 0x0F8 | Tx_CAS_change_P8 | Tx CAS change for timeslots in Port 8 | 144 |
| 0x0C4 | Tx_CAS_change_mask_P1 | Tx CAS change mask for Port 1 | 144 |
| 0x0CC | Tx_CAS_change_mask_P2 | Tx CAS change mask for Port 1 | 144 |
| 0x0D4 | Tx_CAS_change_mask_P3 | Tx CAS change mask for Port 1 | 144 |
| 0x0DC | Tx_CAS_change_mask_P4 | Tx CAS change mask for Port 1 | 144 |

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x0E4 | Tx_CAS_change_mask_P5 | Tx CAS change mask for Port 1 | 144 |
| 0x0EC | Tx_CAS_change_mask_P6 | Tx CAS change mask for Port 1 | 144 |
| 0x0F4 | Tx_CAS_change_mask_P7 | Tx CAS change mask for Port 1 | 144 |
| 0x0FC | Tx_CAS_change_mask_P8 | Tx CAS change mask for Port 1 | 144 |
| 0x100 | RTS_change | RTS change register for Ports 1 to 8 | 144 |
| 0x104 | RTS_mask | RTS change mask for Ports 1 to 8 | 144 |
| 0x140 | CW_bits_change_low_bundles | CW bits change for bundles 0 to 31 | 144 |
| 0x144 | CW_bits_mask_low_bundles | CW bits change mask for bundles 31 to 0 | 144 |
| 0x148 | CW_bits_change_high_bundles | CW bits change for bundles 32 to 63 | 145 |
| 0x14C | CW_bits_mask_high_bundles | CW bits change mask for bundles 63 to 32 | 145 |
| 0x180 | CW_bits_change_mask | Which CW fields (L, R, M, FRG) cause interrupts on change | 145 |
| 0x1C0 | CPU_Queues_change | Which CPU pools and queues went above/below thresholds | 145 |
| 0x1C4 | CPU_Queues_mask | CPU Queues changed mask | 146 |

**Intpend 0x000**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | Reserved | - | 0x0 | Must be set to zero |
| [27] | ETH_MAC | R/W | 0x0 | Ethernet MAC interrupt. Read the MAC_interrupt_status register to determine the interrupt source(s). |
| [26] | CPU Queues | R/W | 0x0 | The fill level of one or more of the CPU queues and pools has gone beyond the configured threshold. Read the CPU_Queues_change register to determine the interrupt source(s). |
| [25] | CW_bits_change | R/W | 0x0 | At least one of the L, R, M or FRG control Word fields has changed in one or more bundles. Read the CW_bits_change_low_bundles and CW_bits_change_high_bundles registers to determine the interrupt source(s).<br>The CW_bits_change_mask register indicates which of the four CW fields can cause an interrupt when changed. |
| [24] | RTS_changes | R/W | 0x0 | 1 = The state of the RTS pin (TDMn_RSIG_RTS) for one or more ports has changed. This only applies for port in asynchronous serial interface mode (Port[n]_cfg_reg. Int_type=00). Read the RTS_change register to determine the interrupt source(s). |
| [23] | Tx_CAS_change_P8 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port8. Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [22] | Tx_CAS_change_P7 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port7. Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [21] | Tx_CAS_change_P6 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port6. Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [20] | Tx_CAS_change_P5 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port5. Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [19] | Tx_CAS_change_P4 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port4. Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [18] | Tx_CAS_change_P3 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port3. Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [17] | Tx_CAS_change_P2 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port2. Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [16] | Tx_CAS_change_P1 | R/W | 0x0 | A change has occurred in the CAS signaling bits for Port1. |

**Intpend 0x000**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | Read the Port7 Tx_CAS_change register to determine the interrupt source(s). |
| [15] | JBC_underrun_P8 | R/W | 0x0 | One of the Port8 Jitter Buffers is in underrun state. Read the Port8 JBC_underrun register to determine the interrupt source(s). |
| [14] | JBC_underrun_P7 | R/W | 0x0 | One of the Port7 Jitter Buffers is in underrun state. Read the Port7 JBC_underrun register to determine the interrupt source(s). |
| [13] | JBC_underrun_P6 | R/W | 0x0 | One of the Port6 Jitter Buffers is in underrun state. Read the Port6 JBC_underrun register to determine the interrupt source(s). |
| [12] | JBC_underrun_P5 | R/W | 0x0 | One of the Port5 Jitter Buffers is in underrun state. Read the Port5 JBC_underrun register to determine the interrupt source(s). |
| [11] | JBC_underrun_P4 | R/W | 0x0 | One of the Port4 Jitter Buffers is in underrun state. Read the Port4 JBC_underrun register to determine the interrupt source(s). |
| [10] | JBC_underrun_P3 | R/W | 0x0 | One of the Port3 Jitter Buffers is in underrun state. Read the Port3 JBC_underrun register to determine the interrupt source(s). |
| [9] | JBC_underrun_P2 | R/W | 0x0 | One of the Port2 Jitter Buffers is in underrun state. Read the Port2 JBC_underrun register to determine the interrupt source(s). |
| [8] | JBC_underrun_P1 | R/W | 0x0 | One of the Port1 Jitter Buffers is in underrun state. Read the Port1 JBC_underrun register to determine the interrupt source(s). |
| [7] | Rx_CAS_change_P8 | R/W | 0x0 | A change has occurred in Port8 Receive Line CAS table. Read the Port8 Rx_CAS_change register to determine the interrupt source(s). |
| [6] | Rx_CAS_change_P7 | R/W | 0x0 | A change has occurred in Port7 Receive Line CAS table. Read the Port7 Rx_CAS_change register to determine the interrupt source(s). |
| [5] | Rx_CAS_change_P6 | R/W | 0x0 | A change has occurred in Port6 Receive Line CAS table. Read the Port6 Rx_CAS_change register to determine the interrupt source(s). |
| [4] | Rx_CAS_change_P5 | R/W | 0x0 | A change has occurred in Port5 Receive Line CAS table. Read the Port5 Rx_CAS_change register to determine the interrupt source(s). |
| [3] | Rx_CAS_change_P4 | R/W | 0x0 | A change has occurred in Port4 Receive Line CAS table. Read the Port4 Rx_CAS_change register to determine the interrupt source(s). |
| [2] | Rx_CAS_change_P3 | R/W | 0x0 | A change has occurred in Port 3 Receive Line CAS table. Read the Port3 Rx_CAS_change register to determine the interrupt source(s). |
| [1] | Rx_CAS_change_P2 | R/W | 0x0 | A change has occurred in Port 2 Receive Line CAS table. Read the Port2 Rx_CAS_change register to determine the interrupt source(s). |
| [0] | Rx_CAS_change_P1 | R/W | 0x0 | A change has occurred in Port 1 Receive Line CAS table. Read the Port1 Rx_CAS_change register to determine the interrupt source(s). |

**Intmask 0x004**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:28] | Reserved | - | 0x0 | Must be set to zero |
| [27] | ETH_MAC | R/W | 0x1 | Mask Ethernet MAC interrupt. |
| [26] | CPU Queues | R/W | 0x1 | Mask CPU Queues change interrupt. |

**Intmask 0x004**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [25] | CW_Bits_change | R/W | 0x1 | Mask Control Word bits change interrupt. |
| [24] | RTS_changes | R/W | 0x1 | Mask RTS change interrupt. |
| [23] | Tx_CAS_change_P8 | R/W | 0x1 | Mask Tx_CAS_change_P8 interrupt. |
| [22] | Tx_CAS_change_P7 | R/W | 0x1 | Mask Tx_CAS_change_P7 interrupt. |
| [21] | Tx_CAS_change_P6 | R/W | 0x1 | Mask Tx_CAS_change_P6 interrupt. |
| [20] | Tx_CAS_change_P5 | R/W | 0x1 | Mask Tx_CAS_change_P5 interrupt. |
| [19] | Tx_CAS_change_P4 | R/W | 0x1 | Mask Tx_CAS_change_P4 interrupt. |
| [18] | Tx_CAS_change_P3 | R/W | 0x1 | Mask Tx_CAS_change_P3 interrupt. |
| [17] | Tx_CAS_change_P2 | R/W | 0x1 | Mask Tx_CAS_change_P2 interrupt. |
| [16] | Tx_CAS_change_P1 | R/W | 0x1 | Mask Tx_CAS_change_P1 interrupt. |
| [15] | JBC_underrun_P8 | R/W | 0x1 | Mask JBC_underrun_P8 interrupt. |
| [14] | JBC_underrun_P7 | R/W | 0x1 | Mask JBC_underrun_P7 interrupt. |
| [13] | JBC_underrun_P6 | R/W | 0x1 | Mask JBC_underrun_P6 interrupt. |
| [12] | JBC_underrun_P5 | R/W | 0x1 | Mask JBC_underrun_P5 interrupt. |
| [11] | JBC_underrun_P4 | R/W | 0x1 | Mask JBC_underrun_P4 interrupt. |
| [10] | JBC_underrun_P3 | R/W | 0x1 | Mask JBC_underrun_P3 interrupt. |
| [9] | JBC_underrun_P2 | R/W | 0x1 | Mask JBC_underrun_P2 interrupt. |
| [8] | JBC_underrun_P1 | R/W | 0x1 | Mask JBC_underrun_P1 interrupt. |
| [7] | Rx_CAS_change_P8 | R/W | 0x1 | Mask Rx_CAS_change_P8 interrupt. |
| [6] | Rx_CAS_change_P7 | R/W | 0x1 | Mask Rx_CAS_change_P7 interrupt. |
| [5] | Rx_CAS_change_P6 | R/W | 0x1 | Mask Rx_CAS_change_P6 interrupt. |
| [4] | Rx_CAS_change_P5 | R/W | 0x1 | Mask Rx_CAS_change_P5 interrupt. |
| [3] | Rx_CAS_change_P4 | R/W | 0x1 | Mask Rx_CAS_change_P4 interrupt. |
| [2] | Rx_CAS_change_P3 | R/W | 0x1 | Mask Rx_CAS_change_P3 interrupt. |
| [1] | Rx_CAS_change_P2 | R/W | 0x1 | Mask Rx_CAS_change_P2 interrupt. |
| [0] | Rx_CAS_change_P1 | R/W | 0x1 | Mask Rx_CAS_change_P1 interrupt. |

**Rx_CAS_change 0x40+(port-1)*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Rx_CAS_change | R/W | 0x0000 0000 | Bit 31 represents timeslot 31 and bit 0 represents timeslot 0 for the port. When a bit is set it indicates a change in received CAS (from the Ethernet port) in the corresponding timeslot. The current CAS bits can be read from the appropriate Rx_Line_CAS register (section 11.4.10). See section.10.6.5.2 |

**JBC_underrun 0x80+(port-1)*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | JBC_underrun | R/W | 0x0000 0000 | Bit 31 represents timeslot 31 and bit 0 represents timeslot 0 for the port. When a bit is set it indicates a jitter buffer underrun for the corresponding timeslot. |

**JBC_underrun_mask 0x84+(port-1)*8**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | JBC_underrun_mask | R/W | 0xFFFF FFFF | Each bit masks an interrupt caused by the corresponding bit in the JBC_underrun register. |

**Tx_CAS_change 0xC0+(port-1)*8**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Tx_CAS_change | R/W | 0x0000 0000 | Bit 31 represents timeslot 31 and bit 0 represents timeslot 0 for the port. When a bit is set it indicates a change in transmit (toward the Ethernet port) CAS bits in the corresponding timeslot. The current CAS bits can be read from the signaling registers in the neighboring framer IC. See section 10.6.5.1. |

**Tx_CAS_change_mask 0xC4+(port-1)*8**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Tx_CAS_change_maxk | R/W | 0xFFFF FFFF | Each bit masks interrupts caused by the corresponding bit in the Tx_CAS_change register. See section 10.6.5.1. |

**RTS_change 0x100**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7] | RTS8_ change | R/W | 0x0 | TDM8_RTS input level changed. |
| [6] | RTS7_ change | R/W | 0x0 | TDM7_RTS input level changed. |
| [5] | RTS6_ change | R/W | 0x0 | TDM6_RTS input level changed. |
| [4] | RTS5_ change | R/W | 0x0 | TDM5_RTS input level changed. |
| [3] | RTS4_ change | R/W | 0x0 | TDM4_RTS input level changed. |
| [2] | RTS3_ change | R/W | 0x0 | TDM3_RTS input level changed. |
| [1] | RTS2_ change | R/W | 0x0 | TDM2_RTS input level changed. |
| [0] | RTS1_ change | R/W | 0x0 | TDM1_RTS input level changed. |

**RTS_mask 0x104**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | RTS_mask | R/W | 0xFF | Each bit masks interrupts caused by the corresponding bit in the RTS_change register. |

**CW_bits_change_low_bundles 0x140**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | CW_bits_change | R/W | 0xFFFF FFFF | Bit 31 represents bundle 31 and bit 0 represents bundle 0. When a bit is set it indicates the corresponding bundle had a change in one of the bundle's control word fields: L, R, M or FRG. The CW_bits_change_mask register specifies which of the four Control Word fields can cause an interrupt when changed. The current state of the four fields can be read from the Packet Classifier Status register in the per-bundle status tables (section 11.4.4.1). |

**CW_bits_mask_low_bundles 0x144**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | CW_bits_mask | R/W | 0xFFFF FFFF | Bit 31 represents bundle 31 and bit 0 represents bundle 0. Mask the interrupt from the corresponding bit in the |

**CW_bits_mask_low_bundles 0x144**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| | | | | CW_bits_change_low_bundles register. |

**CW_bits_change_high_bundles 0x148**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | CW_bits_change | R/W | 0xFFFF FFFF | Bit 31 represents bundle 63 and bit 0 represents bundle 32. When a bit is set it indicates the corresponding bundle had a change in one of the bundle's control word fields: L, R, M or FRG. The CW_bits_change_mask register specifies which of the four Control Word fields can cause an interrupt when changed. The current state of the four fields can be read from the Packet Classifier Status register in the per-bundle status tables (section 11.4.4.1). |

**CW_bits_mask_high_bundles 0x14C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | CW_bits_mask | R/W | 0xFFFF FFFF | Bit 31 represents bundle 63; bit 0 represents bundle 32. Mask the interrupt from the corresponding bit in the CW_bits_change_high_bundles register. |

**CW_bits_change_mask 0x180**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:6] | Reserved | - | 0x0 | Must be set to zero |
| [5] | Rx_sync_loss | R/W | None | Mask interrupts caused by L field changing in Control Word |
| [4] | Rx_remote_fail | R/W | None | Mask interrupts caused by R field changing in Control Word |
| [3:2] | Rx_Lbit_modifier | R/W | None | Mask interrupts caused by M field changing in Control Word |
| [1:0] | Fragmentation_bits | R/W | None | Mask interrupts caused by FRG field changing in Control Word |

**CPU_Queues_change 0x1C0**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:10] | Reserved | - | 0x0 | Must be set to zero |
| [9] | TDM_to_CPU_pool_thresh | R/W | 0x0 | TDM to CPU pool level ≤ threshold. |
| [8] | TDM_to_CPU_q_thresh | R/W | 0x0 | TDM to CPU queue level ≥ threshold. |
| [7] | CPU_to_ETH_q_thresh | R/W | 0x0 | CPU to Ethernet queue level ≤ threshold. |
| [6] | ETH_to_CPU_pool_thresh | R/W | 0x0 | Ethernet to CPU pool level ≤ threshold. |
| [5] | ETH_to_CPU_q_thresh | R/W | 0x0 | Ethernet to CPU queue level ≥ threshold |
| [4:3] | Reserved | R/W | 0x0 | Must be set to zero |
| [2] | CPU_to_TDM_q_thresh | R/W | 0x0 | CPU to TDM queue level ≥ threshold. |
| [1] | Tx_return_q_thresh | R/W | 0x0 | CPU TX return queue level ≥ threshold. |
| [0] | Rx_ return_q_thresh | R/W | 0x0 | CPU RX return queue level ≥ threshold. |

**CPU_Queues_mask 0x1C4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:10] | Reserved | - | 0x0 | Must be set to zero |
| [9] | TDM_to_CPU_pool_thresh | R/W | 0x1 | Mask TDM_to_CPU_pool_thresh interrupts |
| [8] | TDM_to_CPU_q_thresh | R/W | 0x1 | Mask TDM_to_CPU_q_thresh interrupts |
| [7] | CPU_to_ETH_q_thresh | R/W | 0x1 | Mask CPU_to_ETH_q_thresh interrupts |
| [6] | ETH_to_CPU_pool_thresh | R/W | 0x1 | Mask ETH_to_CPU_pool_thresh interrupts |
| [5] | ETH_to_CPU_q_thresh | R/W | 0x1 | Mask ETH_to_CPU_q_thresh interrupts |
| [4:3] | Reserved | R/W | 0x1 | Must be set to zero |
| [2] | CPU_to_TDM_q_thresh | R/W | 0x1 | Mask CPU_to_TDM_q_thresh interrupts |
| [1] | Tx_return_q_thresh | R/W | 0x1 | Mask Tx_return_q_thresh interrupts |
| [0] | Rx_return_q_thresh | R/W | 0x1 | Mask Rx_return_q_thresh interrupts |

### 11.4.15 Packet Classifier

The base address for the packet classifier register space is **0x70,000**. In the register descriptions in this section the index **n** indicates register number: 1 to 8. These registers can store eight possible OAM bundle numbers.

**Table 11-16. Packet Classifier OAM Identification Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x000 | OAM Identification1 | 1st Identification for control packets | 147 |
| 0x004 | OAM Identification2 | 2nd Identification for control packets | 147 |
| 0x008 | OAM Identification3 | 3rd Identification for control packets | 147 |
| 0x00C | OAM Identification4 | 4th Identification for control packets | 147 |
| 0x010 | OAM Identification5 | 5th Identification for control packets | 147 |
| 0x014 | OAM Identification6 | 6th Identification for control packets | 147 |
| 0x018 | OAM Identification7 | 7th Identification for control packets | 147 |
| 0x01C | OAM Identification8 | 8th Identification for control packets | 147 |
| 0x080 | OAM Identification Validity1 | 1st Identification validity for control packets | 147 |
| 0x084 | OAM Identification Validity2 | 2nd Identification validity for control packets | 147 |
| 0x088 | OAM Identification Validity3 | 3rd Identification validity for control packets | 147 |
| 0x08C | OAM Identification Validity4 | 4th Identification validity for control packets | 147 |
| 0x090 | OAM Identification Validity5 | 5th Identification validity for control packets | 147 |
| 0x094 | OAM Identification Validity6 | 6th Identification validity for control packets | 147 |
| 0x098 | OAM Identification Validity7 | 7th Identification validity for control packets | 147 |
| 0x09C | OAM Identification Validity8 | 8th Identification validity for control packets | 147 |

**OAM_Identification[n] 0x000+(n-1)*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | OAM Identification | R/W | None | OAM Identification **n**. If the corresponding validity bit (below) is set then the packet classifier compares the bundle identifier of received packets with the value stored in this register. If they match then the packet classifier considers the received packet to be an OAM packet. See section 10.6.13.3. |

**OAM_Identification_validity[n] 0x080+(n-1)*4**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:1] | Reserved | - | 0x0 | Must be set to zero |
| [0] | OAM Identification Validity | R/W | 0x0 | 1 = OAM Identification **n** (above) has a valid value. See section 10.6.13.3. |

### 11.4.16 Ethernet MAC

The base address for the Ethernet MAC register space is **0x72,000**.

Configuration and status registers are listed in subsection 11.4.16.1. Counters are listed in subsection 11.4.16.2.

#### 11.4.16.1 Ethernet MAC Configuration and Status Registers

**Table 11-17. Ethernet MAC Registers**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x00 | MAC_network_control | MAC control register | 148 |
| 0x04 | MAC_network_configuration | MAC configuration register | 149 |
| 0x08 | MAC_network_status | MAC network status register | 150 |
| 0x14 | MAC_transmit_status | MAC transmitter status register | 150 |
| 0x24 | MAC_interrupt_status | MAC interrupt status register | 150 |
| 0x28 | MAC_interrupt_enable | MAC interrupt enable register | 150 |
| 0x2C | MAC_interrupt_disable | MAC interrupt disable register | 151 |
| 0x30 | MAC_interrupt_mask | MAC interrupt mask register | 151 |
| 0x34 | MAC_PHY_maintenance | PHY maintenance register | 152 |
| 0x38 | MAC_pause_time | MAC pause time register | 152 |
| 0x98 | MAC_specific_address_lower | MAC specific address register (bits 31:0) | 152 |
| 0x9C | MAC_specific_address_upper | MAC specific address register (bits 47:32) | 152 |
| 0xBC | MAC_transmit_paulse_quantum | MAC transmit pause quantum register | 153 |
| 0xC0 | PHY_SMII_status | PHY SMII status register | 153 |

When reading from Ethernet MAC data elements wider than 16 bits in 16-bit mode, use the following procedure:

1. Read from address 2, i.e. H_AD[1]=1. All 32 bits are internally latched and bits 15:0 are output on H_D[15:0].
2. Read from address 0, i.e. H_AD [1]=0. Bits 31:16 are output on H_D[15:0].

When writing to Ethernet MAC data elements wider than 16 bits in 16-bit mode, use the following procedure:

1. Write to address 2, i.e. H_AD[1]=1. Bits 15:0 are internally latched but not written to the register yet.
2. Write to address 0, i.e. H_AD [1]=0. All 32 bits are written to the register. Bits 31:16 on H_D[15:0] are written to address 0. Bits 15:0 in the internal latch are written to address 2.

**MAC_network_control 0x000**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:13] | Reserved. | RO | 0x0 | Read as zero, ignored on write |
| [12] | Transmit_zero_quantum_pause_ packet | WO | None | Writing a 1 to this bit transmits a pause packet with zero pause quantum at the next available transmitter idle time. |
| [11] | Transmit_pause_packet | WO | None | Writing 1 to this bit transmits a pause packet with the pause quantum in the MAC_transmit_paulse_quantum register — at the next available transmitter idle time. |
| [10:9] | Reserved | | 0x0 | Must be set to zero |
| [8] | Back_pressure | R/W | 0x0 | When set in half duplex mode forces collisions on all received packets. |
| [7] | Write_enable_for_statistics_ registers | R/W | 0x0 | Setting this bit to 1 makes the Ethernet MAC counter registers writable for functional test purposes. |
| [6] | Increment_statistics_reg | WO | 0x0 | Writing 1 increments all statistics registers by one for test purposes. |
| [5] | Clear_statistics_reg | WO | 0x0 | Writing 1 clears the statistics registers. |
| [4] | Management_port_enable | R/W | 0x0 | 0 = Disable PHY management port (MDIO high impedance, MDC forced low.)<br>1 = Enable the PHY management port |

**MAC_network_control 0x000**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [3] | Transmit_enable | R/W | 0x0 | 0 = Stop transmission immediately, clear the transmit FIFO and control registers, and reset the transmit queue pointer register to point to the start of the transmit descriptor list.<br>1 = Enable the MAC transmitter to send data.<br>This bit must be set during normal operation. |
| [2] | Rx_enable | R/W | 0x0 | 0 = Stop packet reception immediately<br>1 = Enable the MAC receiver to Rx data |
| [1:0] | Reserved | - | 0x0 | Must be set to zero |

**MAC_network_configuration 0x004**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:20] | Reserved | - | 0x0 | Read as zero, ignored on write |
| [19] | Ignore_Rx_FCS | R/W | 0x0 | When set, packets with FCS/CRC errors are not rejected and no FCS error statistics are counted. For normal operation, this bit must be set to 0. |
| [18] | Enable_half_duplex_Rx | R/W | 0x0 | Enable packets to be received in half-duplex mode while transmitting. |
| [17] | Reserved | - | 0x0 | Must be set to zero |
| [16] | Rx_length_field_checking_enable | R/W | 0x0 | When set, packets with measured lengths shorter than their length fields are discarded. Packets containing a type ID in bytes 13 and 14 (length/type field not counted as length errors. |
| [15:14] | Reserved | - | 0x0 | Must be set to zero |
| [13] | Pause_enable | R/W | 0x0 | When set, Ethernet packet transmission pauses when a valid pause packet is received. |
| [12] | Retry_test | R/W | 0x0 | Must be set to zero for normal operation. If set to one, the back-off between collisions is always one slot time. Setting this bit to one helps test the 'too many retries condition'. Also used in pause packet tests to reduce the pause counters decrement time from 512 bit times to every CLK_MII_RX cycle. |
| [11:10] | MDC_frequency | R/W | 0x2 | Set according to CLK_SYS speed. This field determines by what number CLK_SYS is divided to generate MDC. For conformance with 802.3 MDC must not exceed 2.5 MHz. (MDC is only active during MDIO read and write operations).<br>Must be set to 0x2. |
| [9] | Reserved | | 0x0 | Must be set to zero |
| [8] | Rx_2000_byte_packets | R/W | 0x0 | Setting this bit means the MAC receives packets up to 2000 bytes in length.<br>Normally the MAC rejects any packet above 1518 bytes |
| [7:5] | Reserved | | 0x0 | Must be set to zero |
| [4] | Reserved | R/W | 0x0 | Must be set to 1 |
| [3:2] | Reserved | | 0x0 | Must be set to zero |
| [1] | Full_duplex | R/W | 0x0 | If set to 1 the transmit block ignores the state of collision and carrier sense and allows Rx while transmitting. |
| [0] | Speed | R/W | 0x0 | 0 = 10 Mbit/s operation<br>1 = 100 Mbit/s operation<br>Used only for RMII and SMII interfaces. |

**MAC_network_status 0x008**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:3] | Reserved | - | 0x0 | Must be set to zero |
| [2] | PHY_access_has_completed | RO | 0x1 | 1 = PHY management logic is idle. |
| [1:0] | Reserved | - | 0x0 | Must be set to zero |

**MAC_transmit_status 0x014**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:7] | Reserved | - | 0x0 | Must be set to zero |
| [6] | Transmit_underrun | R/W | 0x0 | Set when the MAC transmit FIFO was read while was empty. If this happens the transmitter forces bad CRC and forces MII_TX_ERR high. Write 1 to clear this bit. |
| [5:3] | Reserved | - | 0x0 | Must be set to zero |
| [2] | Retry_limit_exceeded | R/W | 0x0 | Set when the retry limit has been exceeded. Write 1 to clear this bit. |
| [1] | Collision_occurred | R/W | 0x0 | Set when a collision occurs. Write 1 to clear this bit. |
| [0] | Reserved | - | 0x0 | Must be set to zero |

The MAC generates a single interrupt, the ETH_MAC bit in the Intpend register. The MAC_interrupt_status register below indicates the source of this interrupt. For test purposes each bit can be set or reset by directly writing to this register regardless of the state of the mask register. Otherwise the corresponding bit in the MAC_interrupt_mask register must be cleared for a bit to be set in the MAC_interrupt_status register. All bits are reset to zero on read. If any bit is set in the MAC_interrupt_status register, the ETH_MAC bit is asserted.

At reset all MAC interrupts are disabled. Writing a one to the relevant bit location in the MAC_interrupt_enable register below enables the associated interrupt. Writing a one to the relevant bit location in the MAC_interrupt_disable register below disables the associated interrupt. MAC_interrupt_enable and MAC_interrupt_disable are not registers but merely mechanisms for setting and clearing bits in the read-only MAC_interrupt_mask register.

**MAC_interrupt_status 0x024**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:14] | Reserved | RO | 0x0 | Read 0, ignored on write |
| [13] | Pause_time_zero | R/W | 0x0 | Set when the MAC_pause_time register decrements to zero. Cleared when read. |
| [12] | Pause_packet_ Rxd | R/W | 0x0 | Indicates a valid pause packet has been received. Cleared when read. |
| [11:6] | Reserved | | 0x0 | Must be set to zero |
| [5] | Retry_limit_exceeded | R/W | 0x0 | Transmit error. Cleared when read. |
| [4] | Ethernet_transmit_underrun | R/W | 0x0 | Set when the MAC transmit FIFO was read while was empty. If this happens the transmitter forces bad CRC and forces MII_TX_ERR high. Cleared when read. |
| [3:1] | Reserved | | 0x0 | Must be set to zero |
| [0] | Management_packet_sent | R/W | 0x0 | The PHY maintenance register has completed its operation. Cleared when read. |

**MAC_interrupt_enable 0x028**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:14] | Reserved | - | 0x0 | Must be set to zero |
| [13] | Pause_time_zero | WO | 0x0 | 1 = Enable Pause_time_zero interrupt |
| [12] | Pause_packet_ Rxd | WO | 0x0 | 1 = Enable Pause_packet_Rxd interrupt |
| [11:6] | Reserved | - | 0x0 | Must be set to zero |

**MAC_interrupt_enable 0x028**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [5] | Retry_limit_exceeded | WO | 0x0 | 1 = Enable Retry_limit_exceeded interrupt |
| [4] | Ethernet_transmit_underrun | WO | 0x0 | 1 = Enable Ethernet_transmit_underrun interrupt |
| [3:1] | Reserved | - | 0x0 | Must be set to zero |
| [0] | Management_packet_sent | WO | 0x0 | 1 = Enable Management_packet_sent interrupt |

**MAC_interrupt_disable 0x02C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:14] | Reserved | - | 0x0 | Must be set to zero |
| [13] | Pause_time_zero | WO | 0x0 | 1 = Disable Pause_time_zero interrupt |
| [12] | Pause_packet_ Rxd | WO | 0x0 | 1 = Disable Pause_packet_Rxd interrupt |
| [11:6] | Reserved | - | 0x0 | Must be set to zero |
| [5] | Retry_limit_exceeded | WO | 0x0 | 1 = Disable Retry_limit_exceeded interrupt |
| [4] | Ethernet_transmit_underrun | WO | 0x0 | 1 = Disable Ethernet_transmit_underrun interrupt |
| [3:1] | Reserved | - | 0x0 | Must be set to zero |
| [0] | Management_packet_sent | WO | 0x0 | 1 = Disable Management_packet_sent interrupt |

**MAC_interrupt_mask 0x030**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:14] | Reserved | - | 0x0 | Must be set to zero |
| [13] | Pause_time_zero | RO | 0x1 | 1 = Mask Pause_time_zero interrupt |
| [12] | Pause_packet_ Rxd | RO | 0x1 | 1 = Mask Pause_packet_Rxd interrupt |
| [11:6] | Reserved | - | 0x0 | Must be set to zero |
| [5] | Retry_limit_exceeded | RO | 0x1 | 1 = Mask Retry_limit_exceeded interrupt |
| [4] | Ethernet_transmit_underrun | RO | 0x1 | 1 = Mask Ethernet_transmit_underrun interrupt |
| [3:1] | Reserved | - | 0x0 | Must be set to zero |
| [0] | Management_packet_sent | RO | 0x1 | 1 = Mask Management_packet_sent interrupt |

The MAC_PHY_maintenance register below enables the MAC to communicate with a PHY by means of the MDIO interface. It is used during auto negotiation to ensure that the MAC and the PHY are configured for the same speed and duplex configuration.

The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signaled as complete when the PHY_access_has_completed bit is set in the MAC_network_status register (about 2000 CLK_SYS cycles later). An interrupt is generated as this bit is set. During this time, the MSB of the register is output on the MDIO pin and the LSB is updated from the MDIO pin with each MDC cycle. In this way a PHY management packet is transmitted on MDIO. See Section 22.2.4.5 of the IEEE 802.3 standard. Reading during the shift operation (not recommended) returns the current contents of the shift register.

At the end of the shift operation, the bits have shifted back to their original locations. For a read operation, the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management packet is produced.

**MAC_PHY_maintenance 0x034**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:30] | Start_of_packet | R/W | 0x0 | Must be written 01 for a valid packet |
| [29:28] | Operation | R/W | 0x0 | 00 = Reserved<br>01 = Write<br>10 = Read<br>11 = Reserved |
| [27:23] | PHY_address | R/W | 0x0 | Specifies the PHY to access |
| [22:18] | Register_address | R/W | 0x0 | Specifies the register in the PHY to access |
| [17:16] | Must_be_written_to_10 | R/W | 0x0 | Read as written |
| [15:0] | PHY_data | R/W | 0x0000 | For a write operation this field is the data to be written to the PHY. After a read operation this field contains the data read from the PHY |

**MAC_pause_time 0x038**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | RO | 0x0000 | Read 0, ignored on write |
| [15:0] | Pause time | RO | 0x0000 | Stores the current value of the pause time register, which is decremented every 512 bit times. |

**MAC_specific_address_lower 0x098**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | MAC Specific Address [31:0] | R/W | 0x0 | Least significant bits of the MAC specific address, i.e. bits 31:0. This field is used for transmission of pause packets as described in section 10.6.12.2. |

**MAC_specific_address_upper 0x09C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | RO | 0x0000 | Read 0, ignored on write |
| [15:0] | MAC Specific Address [47:32] | R/W | 0x0000 | Most significant bits of the MAC specific address, i.e. bits 47:32. See MAC_specific_address_lower for details. |

**MAC_transmit_paulse_quantum 0x0BC**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | 0x0000 | Must be set to zero |
| [15:0] | Pause_time | R/W | 0xFFFF | Transmit pause quantum. Used in hardware generation of transmitted pause packets as value for pause quantum. |

**PHY_SMII_status 0x0C0**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:21] | Reserved | RO | 0x0000 | Must be set to zero |
| [20] | SMII_speed | RO | None | Speed recovered from receive SMII<br>0=10Mbps, 1=100Mbps |
| [19] | SMII_Duplex | RO | None | Duplex recovered from receive SMII<br>0=Half Duplex, 1=Full Duplex |
| [18] | SMII_Link | RO | None | Link recovered from receive SMII<br>0=Link is Down, 1=Link is Up |
| [17] | SMII_Jabber | RO | None | Jabber recovered from receive SMII<br>0=OK, 1=Error |
| [16] | SMII_False_Carrier | RO | None | False carrier recovered from receive SMII<br>0=OK, 1=False carrier detected |
| [15:0] | Reserved | RO | 0x0000 | Must be set to zero |

### *11.4.16.2 Ethernet MAC Counters*

**Table 11-18. Ethernet MAC Counters**

| Addr Offset | Register Name | Description | Page |
|---|---|---|---|
| 0x3C | Pause_packets_Rxd_OK | Pause packets received OK counter | 154 |
| 0x40 | Packets_transmitted_OK | Packets transmitted OK counter | 154 |
| 0x44 | Single_collision_packets | Single collision packets counter | 154 |
| 0x48 | Multiple_collision_packets | Multiple collision packets counter | 154 |
| 0x4C | Packets_Rxd_OK | Packets received OK counter | 154 |
| 0x50 | Packet_check_sequence_errors | Packet check sequence errors counter | 154 |
| 0x54 | Alignment_errors | Alignment errors counter | 155 |
| 0x58 | Deferred_transmission_packets | Deferred transmission packets counter | 155 |
| 0x5C | Late_collisions | Late collisions counter | 155 |
| 0x60 | Excessive_collisions | Excessive collisions counter | 155 |
| 0x64 | Transmit_underrun_errors | Transmit underrun errors counter | 155 |
| 0x68 | Carrier_sense_errors | Carrier sense errors counter | 156 |
| 0x74 | Rx_symbol_errors | Rx symbol errors counter | 156 |
| 0x78 | Excessive_length_errors | Excessive length errors counter | 156 |
| 0x7C | Rx_jabbers | Rx jabbers counter | 156 |
| 0x80 | Undersize_packets | Undersize packets counter | 156 |
| 0x84 | SQE_test_errors | SQE test errors counter | 157 |
| 0x8C | Transmitted_pause_packets | Transmitted pause packets counter | 157 |

These counters stick at their maximum value and do not roll over. They also reset to zero when read and therefore should be read frequently enough to prevent loss of data. The Rx counters are only incremented when the Rx_enable bit is set in the MAC_network_control register.

**Pause_packets_Rxd_OK 0x03C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | 0x0 | Must be set to zero |
| [15:0] | Pause_packets_Rxd_OK | R/W | 0x0 | A 16-bit register counting the number of good pause packets received. A good packet has a length of 64 to 1518 (2000 if Rx_2000_byte_packets is set in the MAC_network_configuration register) and has no FCS, alignment or Rx symbol errors. |

**Packets_transmitted_OK 0x040**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:0] | Packets_transmitted_OK | R/W | 0x0 | A 32-bit register counting the number of packets successfully transmitted, i.e. no underrun and not too many retries. |

**Single_collision_packets 0x044**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | 0x0 | Must be set to zero |
| [15:0] | Single_collision_packets | R/W | 0x0 | A 16-bit register counting the number of packets experiencing a single collision before being successfully transmitted, i.e. no underrun. |

**Multiple_collision_packets 0x048**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | 0x0 | Must be set to zero |
| [15:0] | Multiple_collision_packets | R/W | 0x0 | A 16-bit register counting the number of packets experiencing between two and fifteen collisions prior to being successfully transmitted, i.e. no underrun and not too many retries. |

**Packets_Rxd_OK 0x04C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:24] | Reserved | - | 0x0 | Must be set to zero |
| [23:0] | Packets_Rxd_OK | R/W | 0x0 | A 24-bit register counting the number of good packets received, i.e. packet length is 64 to 1518 bytes (2000 if Rx_2000_byte_packets is set in the MAC_network_configuration register) and has no FCS, alignment or Rx symbol errors. |

**Packet_check_sequence_errors 0x050**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Packet_check_sequence_errors | R/W | 0x0 | An 8-bit register counting packets that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (2000 if Rx_2000_byte_packets is set in the MAC_network_configuration register). |

**Alignment_errors 0x054**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Alignment_errors | R/W | 0x0 | An 8-bit register counting packets that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (2000 if Rx_2000_byte_packets is set in the MAC_network_configuration register). |

**Deferred_transmission_packets 0x058**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | 0x0 | Must be set to zero |
| [15:0] | Deferred_transmission_packets | R/W | 0x0 | A 16-bit register counting the number of packets experiencing deferral due to carrier sense being active on their first attempt at transmission. Packets involved in any collision are not counted nor are packets that experienced a transmit underrun. |

**Late_collisions 0x05C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Late_collisions | R/W | 0x0 | An 8-bit register counting the number of packets that experience a collision after the slot time (512 bits) has expired. A late collision is counted twice i.e. both as a collision and a late collision. |

**Excessive_collisions 0x060**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Excessive_collisions | R/W | 0x0 | An 8-bit register counting the number of packets that failed to be transmitted because they experienced 16 collisions. |

**Transmit_underrun_errors 0x064**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Transmit_underruns | R/W | 0x0 | An 8-bit register counting the number of packets not transmitted due to a transmit FIFO underrun. If this register is incremented, no other Ethernet MAC counter is incremented. |

**Carrier_sense_errors 0x068**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Carrier_sense_errors | R/W | 0x0 | An 8-bit register counting the number of packets transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit packet without collision (no underrun). Only incremented in half-duplex mode. The only effect of a carrier sense error is to increment this register. The behavior of the other Ethernet MAC counters is unaffected by the detection of a carrier sense error. |

**Rx_symbol_errors 0x074**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Rx_symbol_errors | R/W | 0x0 | An 8-bit register counting the number of packets that had MII_RX_ERR asserted during reception. |

**Excessive_length_errors 0x078**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Excessive_length_packets | R/W | 0x0 | An 8-bit register counting the number of packets received exceeding 1518 bytes in length (2000 if Rx_2000_byte_packets is set in the MAC_network_configuration register) but do not have a CRC error, an alignment error nor a Rx symbol error. |

**Rx_jabbers 0x07C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Rx_jabbers | R/W | 0x00 | An 8-bit register counting the number of packets received exceeding 1518 bytes in length (2000 if Rx_2000_byte_packets is set in the MAC_network_configuration register) and have either a CRC error, an alignment error or a Rx symbol error. |

**Undersize_packets 0x080**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | Undersize_packets | R/W | 0x0 | An 8-bit register counting the number of packets received less than 64 bytes in length, that do not have either a CRC error or an alignment error. |

**SQE_test_errors 0x084**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:8] | Reserved | - | 0x0 | Must be set to zero |
| [7:0] | SQE_test_errors | R/W | 0x0 | An 8-bit register counting the number of packets where collision was not asserted within 96 bit times (an interpacket gap) of MII_TX_EN being deasserted in half duplex mode. |

**Transmitted_pause_packets 0x08C**

| Bits | Data Element Name | R/W | Reset Value | Description |
|---|---|---|---|---|
| [31:16] | Reserved | - | 0x0 | Must be set to zero |
| [15:0] | Transmitted_pause_packets | R/W | 0x0 | A 16-bit register counting the number of pause packets transmitted. |

# 12. JTAG Information

For the latest JTAG model, search under http://www.maxim-ic.com/tools/bsdl/.

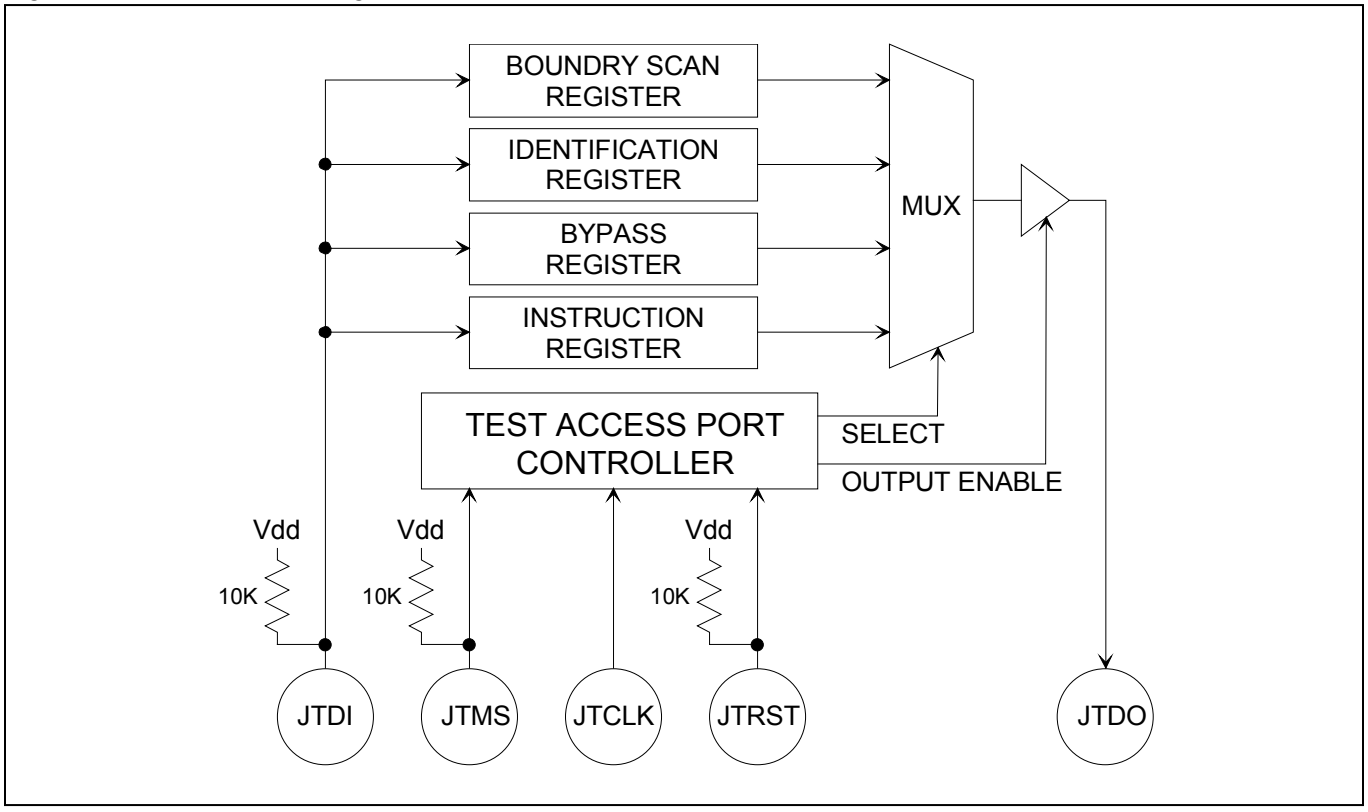
## JTAG Description

The device supports the standard instruction codes SAMPLE/PRELOAD, BYPASS, and EXTEST. Optional public instructions included are HIGHZ, CLAMP and IDCODE. See Figure 12-1 for a block diagram. The device contains the following items which meet the requirements set by the IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture:

| | |
|---|---|
| Test Access Port (TAP) | TAP Controller |
| Instruction Register | Bypass Register |
| Boundary Scan Register | Device Identification Register |

The Test Access Port has the necessary interface pins, namely JTCLK, JTRST_N, JTDI, JTDO, and JTMS. Details on these pins can be found in Table 9-7. Details on the Boundary Scan Architecture and the Test Access Port can be found in IEEE 1149.1-1990, IEEE 1149.1a-1993, and IEEE 1149.1b-1994.

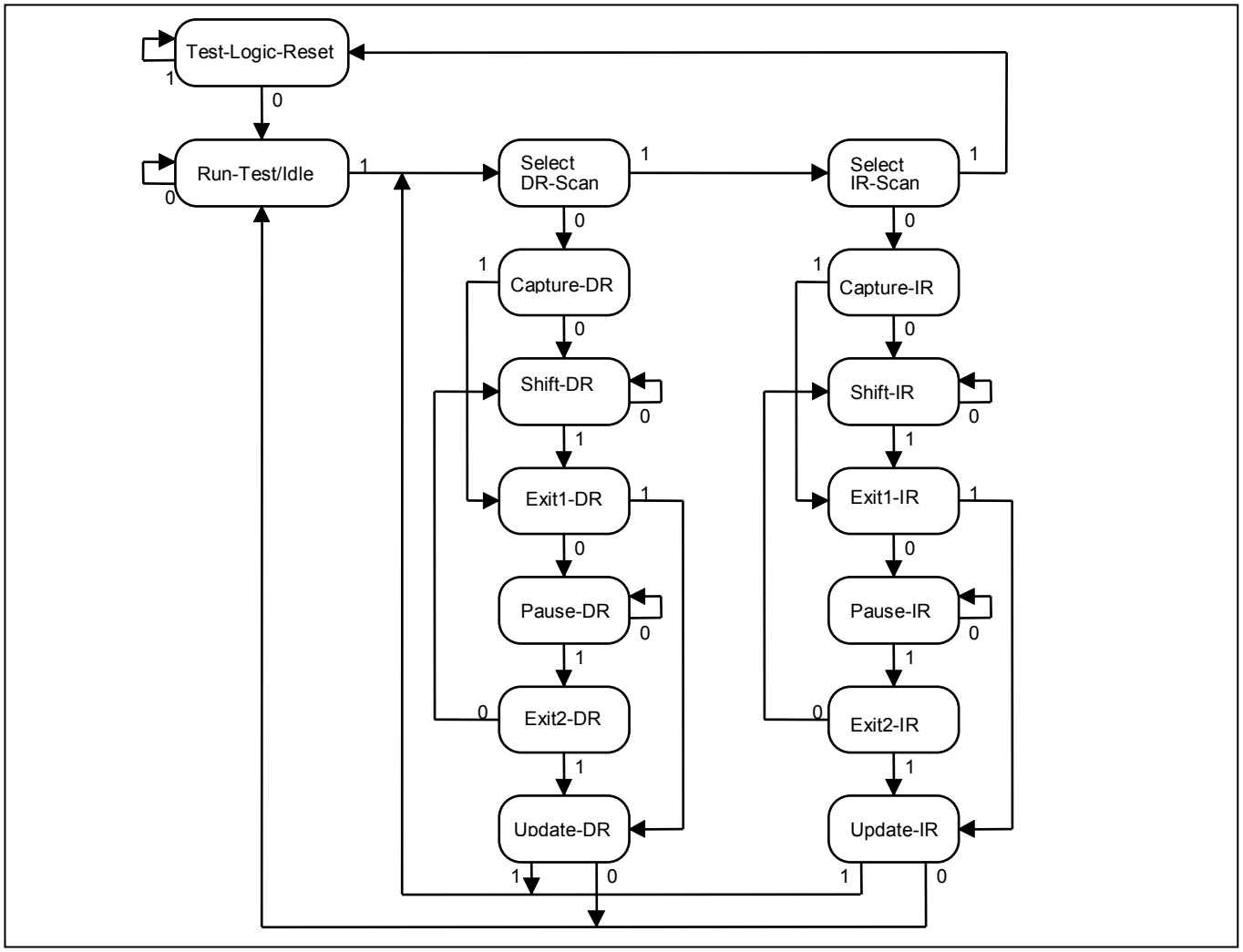
*Figure 12-1. JTAG Block Diagram*




## JTAG TAP Controller State Machine Description

This section covers the details on the operation of the Test Access Port (TAP) Controller State Machine. See Figure 12-2 for details on each of the states described below. The TAP controller is a finite state machine which responds to the logic level at JTMS on the rising edge of JTCLK.

*Figure 12-2. JTAG TAP Controller State Machine*




**Test-Logic-Reset.** Upon power-up of the device, the TAP controller starts in the Test-Logic-Reset state. The Instruction Register contains the IDCODE instruction. All system logic on the device operates normally.

**Run-Test-Idle.** Run-Test-Idle is used between scan operations or during specific tests. The Instruction Register and Test Register remain idle.

**Select-DR-Scan.** All test registers retain their previous state. With JTMS low, a rising edge of JTCLK moves the controller into the Capture-DR state and initiates a scan sequence. JTMS high moves the controller to the Select-IR-SCAN state.

**Capture-DR.** Data may be parallel loaded into the Test Data registers selected by the current instruction. If the instruction does not call for a parallel load or the selected register does not allow parallel loads, the Test Register remains at its current value. On the rising edge of JTCLK, the controller goes to the Shift-DR state if JTMS is low or it to the Exit1-DR state if JTMS is high.

**Shift-DR.** The Test Data Register selected by the current instruction is connected between JTDI and JTDO and shifts data one stage towards its serial output on each rising edge of JTCLK. If a Test Register selected by the current instruction is not placed in the serial path, it maintains its previous state.

**Exit1-DR.** While in this state, a rising edge on JTCLK with JTMS high puts the controller in the Update-DR state which terminates the scanning process. A rising edge on JTCLK with JTMS low puts the controller in the Pause-DR state.

**Pause-DR.** Shifting of the Test registers is halted while in this state. All Test registers selected by the current instruction retain their previous state. The controller remains in this state while JTMS is low. A rising edge on JTCLK with JTMS high puts the controller in the Exit2-DR state.

**Exit2-DR.** While in this state, a rising edge on JTCLK with JTMS high puts the controller in the Update-DR state and terminate the scanning process. A rising edge on JTCLK with JTMS low puts the controller in the Shift-DR state.

**Update-DR.** A falling edge on JTCLK while in the Update-DR state latches the data from the shift register path of the Test registers into the data output latches. This prevents changes at the parallel output due to changes in the shift register. A rising edge on JTCLK with JTMS low, puts the controller in the Run-Test-Idle state. With JTMS high, the controller enters the Select-DR-Scan state.

**Select-IR-Scan.** All Test registers retain their previous state. The Instruction register remains unchanged during this state. With JTMS low, a rising edge on JTCLK moves the controller into the Capture-IR state and initiates a scan sequence for the Instruction register. JTMS high during a rising edge on JTCLK puts the controller back into the Test-Logic-Reset state.

**Capture-IR.** The Capture-IR state is used to load the shift register in the Instruction register with a fixed value. This value is loaded on the rising edge of JTCLK. If JTMS is high on the rising edge of JTCLK, the controller enters the Exit1-IR state. If JTMS is low on the rising edge of JTCLK, the controller enters the Shift-IR state.

**Shift-IR.** In this state, the shift register in the Instruction register is connected between JTDI and JTDO and shifts data one stage for every rising edge of JTCLK towards the serial output. The parallel register, as well as all Test registers remain at their previous states. A rising edge on JTCLK with JTMS high moves the controller to the Exit1-IR state. A rising edge on JTCLK with JTMS low keeps the controller in the Shift-IR state while moving data one stage through the Instruction shift register.

**Exit1-IR.** A rising edge on JTCLK with JTMS low puts the controller in the Pause-IR state. If JTMS is high on the rising edge of JTCLK, the controller enters the Update-IR state and terminate the scanning process.

**Pause-IR.** Shifting of the Instruction register is halted temporarily. With JTMS high, a rising edge on JTCLK puts the controller in the Exit2-IR state. The controller remains in the Pause-IR state if JTMS is low during a rising edge on JTCLK.

**Exit2-IR.** A rising edge on JTCLK with JTMS high put the controller in the Update-IR state. The controller loops back to the Shift-IR state if JTMS is low during a rising edge of JTCLK in this state.

**Update-IR.** The instruction shifted into the Instruction shift register is latched into the parallel output on the falling edge of JTCLK as the controller enters this state. Once latched, this instruction becomes the current instruction. A rising edge on JTCLK with JTMS low, puts the controller in the Run-Test-Idle state. With JTMS high, the controller enters the Select-DR-Scan state.

## JTAG Instruction Register and Instructions

The Instruction register contains a shift register as well as a latched parallel output and is 3 bits in length. When the TAP controller enters the Shift-IR state, the instruction shift register is connected between JTDI and JTDO. While in the Shift-IR state, a rising edge on JTCLK with JTMS low shifts data one stage towards the serial output at JTDO. A rising edge on JTCLK in the Exit1-IR state or the Exit2-IR state with JTMS high moves the controller to the Update-IR state. The falling edge of that same JTCLK latches the data in the instruction shift register to the instruction parallel output. Instructions supported by the device and their respective operational binary codes are shown in Table 12-1.

**Table 12-1. JTAG Instruction Codes**

| Instructions | Selected Register | Instruction Codes |
|---|---|---|
| SAMPLE/PRELOAD | Boundary Scan | 010 |
| BYPASS | Bypass | 111 |
| EXTEST | Boundary Scan | 000 |
| CLAMP | Bypass | 011 |
| HIGHZ | Bypass | 100 |
| IDCODE | Device Identification | 001 |

**SAMPLE/PRELOAD.** A mandatory instruction for the IEEE 1149.1 specification. This instruction supports two functions. The digital I/Os of the device can be sampled at the Boundary Scan register without interfering with the normal operation of the device by using the Capture-DR state. SAMPLE/PRELOAD also allows the device to shift data into the Boundary Scan register via JTDI using the Shift-DR state.

**EXTEST.** EXTEST allows testing of all interconnections to the device. When the EXTEST instruction is latched in the instruction register, the following actions occur. Once enabled via the Update-IR state, the parallel outputs of all digital output pins are driven. The Boundary Scan register is connected between JTDI and JTDO. The Capture-DR samples all digital inputs into the Boundary Scan register.

**BYPASS.** When the BYPASS instruction is latched into the parallel Instruction register, JTDI connects to JTDO through the one-bit Bypass Test register. This allows data to pass from JTDI to JTDO not affecting the device's normal operation.

**IDCODE.** When the IDCODE instruction is latched into the parallel Instruction register, the Identification Test register is selected. The device identification code is loaded into the Identification register on the rising edge of JTCLK following entry into the Capture-DR state. Shift-DR can be used to shift the identification code out serially via JTDO. During Test-Logic-Reset, the identification code is forced into the instruction register's parallel output. The device ID code always has a one in the LSB position. The device ID codes are listed in Table 12-2.

**Table 12-2. JTAG ID Code**

| Device | ID Code (hex) | | |
|---|---|---|---|
| | Rev[31:28] | Device ID [27:12] | Manu[11:0] |
| DS34S101 | 0 | 0098 | 143 |
| DS34S102 | 0 | 0099 | 143 |
| DS34S104 | 0 | 009A | 143 |
| DS34S108 | 0 | 009B | 143 |

**HIGHZ.** All digital outputs are placed into a high impedance state. The Bypass Register is connected between JTDI and JTDO.

**CLAMP.** All digital outputs pins output data from the boundary scan parallel output while connecting the Bypass Register between JTDI and JTDO. The outputs do not change during the CLAMP instruction.

## JTAG Test Registers

IEEE 1149.1 requires a minimum of two Test registers; the Bypass register and the Boundary Scan register. An optional Test register has been included in the device design. This Test register is the Identification register and is used in conjunction with the IDCODE instruction and the Test-Logic-Reset state of the TAP controller.

**Bypass Register.** This is a single one-bit shift register used in conjunction with the BYPASS, CLAMP, and HIGHZ instructions, which provides a short path between JTDI and JTDO.

**Identification Register.** The Identification register contains a 32-bit shift register and a 32-bit latched parallel output. This register is selected during the IDCODE instruction and when the TAP controller is in the Test-Logic-Reset state.

**Boundary Scan Register.** This register contains both a shift register path and a latched parallel output for all control cells and digital I/O cells and is 32 bits in length. The BSDL file found at http://www.maxim-ic.com/tools/bsdl/ shows the entire cell bit locations and definitions.

# 13. DC Electrical Characteristics

## ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Input, Bi-directional or Open Drain
Output Lead with Respect to DVSS .....-0.5V to +5.5V
Supply Voltage (DVDDIO) with Respect to DVSS.....-0.5V to +3.6V
Supply Voltage (DVDDC, ACVDD1, ACVDD2) with Respect to DVSS.....-0.5V to +2.0V
Ambient Operating Temperature Range..... -40°C to +85°C
Junction Operating Temperature Range..... -40°C to +125°C
Storage Temperature Range..... -55°C to +125°C
Soldering Temperature Range ..... See IPC/JEDEC J-STD-020 Specification

*These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods can affect reliability. Ambient Operating Temperature Range is assuming the device is mounted on a JEDEC standard test board in a convection cooled JEDEC test enclosure.*

***Note:*** *The typical values listed below are not production tested.*

**Table 13-1. Recommended DC Operating Conditions**
($T_j$ = -40°C to +85°C.)

| Parameter | Symbol | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Output Logic 1 | $V_{IH}$ | | 2.4 | | 3.465 | V |
| Output Logic 0 | $V_{IL}$ | | -0.3 | | +0.8 | V |
| Power Supply Voltage | DVDDIO | | 3.135 | 3.300 | 3.465 | V |
| Power Supply Voltage | DVDDC, ACVDD1, ACVDD2 | | 1.71 | 1.8 | 1.89 | V |

**Table 13-2. DC Electrical Characteristics**
($T_j$ = -40°C to +85°C.)

| Parameter | Symbol | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| 3.3V Supply Current (@ 3.465V)<br>DS34S108<br>DS34S104<br>DS34S102<br>DS34S101 | $I_{DDIO}$ | Note 1 | | <br>50<br>50<br>TBD<br>TBD | <br>65<br>65<br>TBD<br>TBD | mA |
| 1.8V Supply Current (@1.89V) | $I_{DDC}$ | Note 1 | | 225 | 280 | mA |
| Lead Capacitance | $C_{IO}$ | | | 7 | | pF |
| Input Leakage | $I_{IL}$ | | -10 | | +10 | µA |
| Input Leakage, Internal Pull-Down | $I_{ILP}$ | | -100 | | -10 | µA |
| Output Leakage (when Hi-Z) | $I_{LO}$ | | -10 | | +10 | µA |
| Output Voltage ($I_{OH}$ = -4.0mA) | $V_{OH}$ | 4 mA output | 2.4 | | | V |
| Output Voltage ($I_{OL}$ = +4.0mA) | $V_{OL}$ | 4 mA output | | | 0.4 | V |
| Output Voltage ($I_{OH}$ = -8.0mA) | $V_{OH}$ | 8 mA output | 2.4 | | | V |
| Output Voltage ($I_{OL}$ = -8.0mA) | $V_{OL}$ | 8 mA output | | | 0.4 | V |
| Output Voltage ($I_{OH}$ = -12.0mA) | $V_{OH}$ | 12 mA output | 2.4 | | | V |
| Output Voltage ($I_{OL}$ = +12.0mA) | $V_{OL}$ | 12 mA output | | | 0.4 | V |
| Input Voltage Logic 1 | $V_{IH}$ | | 2.0 | | | V |
| Input Voltage Logic 0 | $V_{IL}$ | | | | 0.8 | V |

**NOTES:**
1. All outputs loaded with rated capacitance; all inputs between DVDDIO and DVSS; inputs with pull-ups connected to DVDDIO.

# 14. AC Timing Characteristics

**Table 14-1. Input Pin Transition Time Requirements**

| PARAMETER | SYMBOL | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| Rise Time | tr | 10 to 90% of DVDDIO | | | 6 | ns |
| Fall Time | tf | 90 to 10% of DVDDIO | | | 6 | ns |

## 14.1 CPU Interface Timing

**Table 14-2. CPU Interface AC characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| RST_SYS_N Active Low Pulse Width | T5 | 50 | | | µs |
| H_CS_N Deasserted or H_R_W_N Low to H_D[31:0] High-Z | T22 | | | 16.2 | ns |
| H_READY_N Active Pull-Up Pulse Width | T26 | 2.9 | | 6.8 | ns |
| Latest of H_WR_BEx_N Asserted or H_CS_N Asserted to H_D[31:0] Valid | T31 | | | 0 | ns |
| H_CS_N Deasserted to H_D[31:0] Not Valid | T32 | | | 0 | ns |
| H_CS_N Asserted to H_AD[24:1] Valid | T33 | | | 0 | ns |
| H_CS_N Deasserted to H_AD[24:1] Not Valid | T34 | | | 0 | ns |
| H_CS_N Asserted to H_R_W_N Valid | T35 | | | 0 | ns |
| H_CS_N Deasserted to H_R_W_N Not Valid | T36 | | | 0 | ns |
| H_CS_N Deasserted to H_READY_N High | T37 | | | 12 | ns |
| H_CS_N Deasserted to H_WR_BEx_N[3:0] Not Valid | T40 | | | 0 | ns |
| Delay Between Two Successive Accesses | T43 | 1.5 | | | Internal CLK_SYS cycles |
| H_D[31:0] Valid before H_READY_N Active Low | T44 | 1.5 | | | ns |

**NOTE:** The output timing specified assumes 50 pF load.

*Figure 14-1. RST_SYS_N Timing*

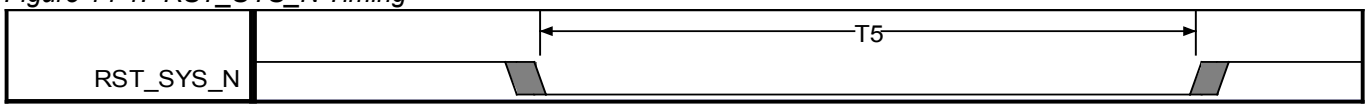


*Figure 14-2. CPU Interface Write Cycle Timing*

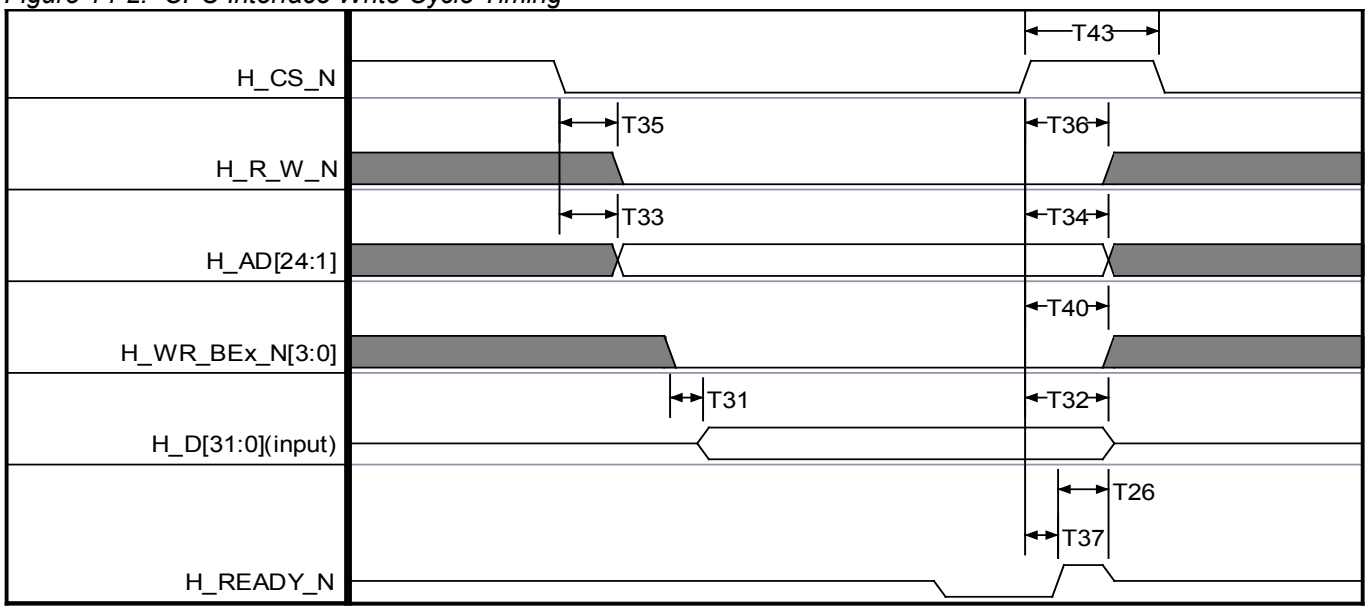
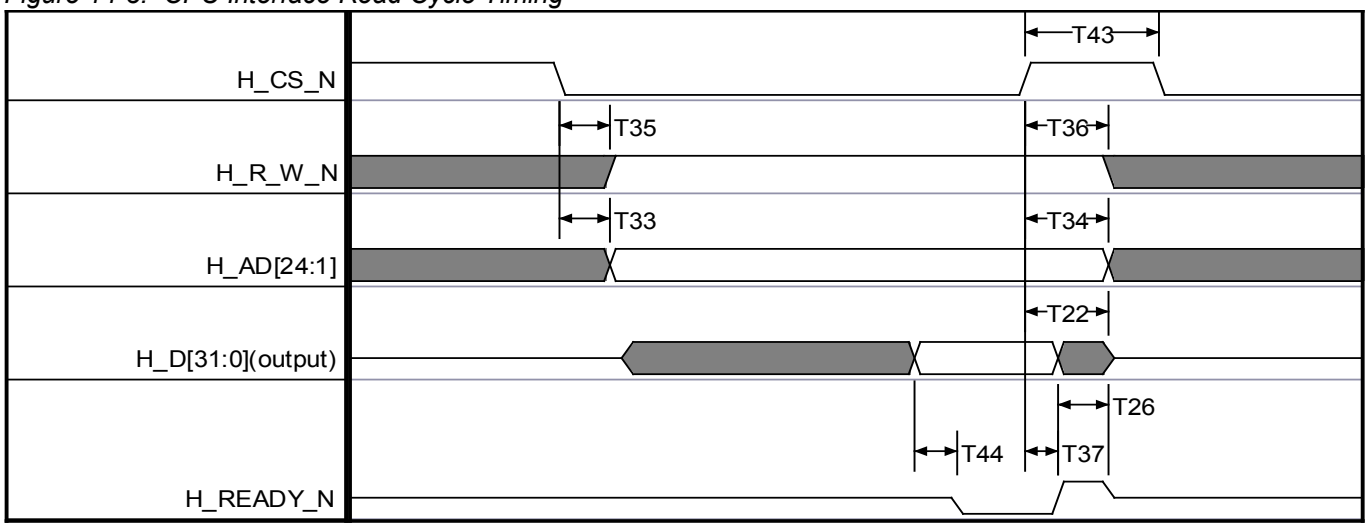



*Figure 14-3. CPU Interface Read Cycle Timing*




## 14.2 SPI Interface Timing

**Table 14-3. SPI Interface AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| SPI_SEL_N Deasserted to SPI_SEL_N Asserted | T230 | 70 | | | ns |
| SPI_CLK Frequency | T231 | | | 12.09 | MHz |
| SPI_CLK Period | T231 | 82.7 | | | ns |
| SPI_CLK to SPI_MISO Output Hold | T232 | 5.3 | | | ns |
| SPI_CLK to SPI_MISO Output Valid | T233 | | | 17.5 | ns |
| SPI_MOSI Input Hold After SPI_CLK Edge | T234 | 5 | | | ns |
| SPI_MOSI Input Setup Prior to SPI_CLK Edge | T235 | 5 | | | ns |
| SPI_SEL_N Asserted to SPI_MISO Active | T236 | | | 15 | ns |
| SPI_SEL_N Deasserted to SPI_MISO High-Z | T237 | | | 12 | ns |

**NOTE:** The output timing specified assumes 50pf load.

*Figure 14-4. SPI interface Timing (SPI_CP = 0)*

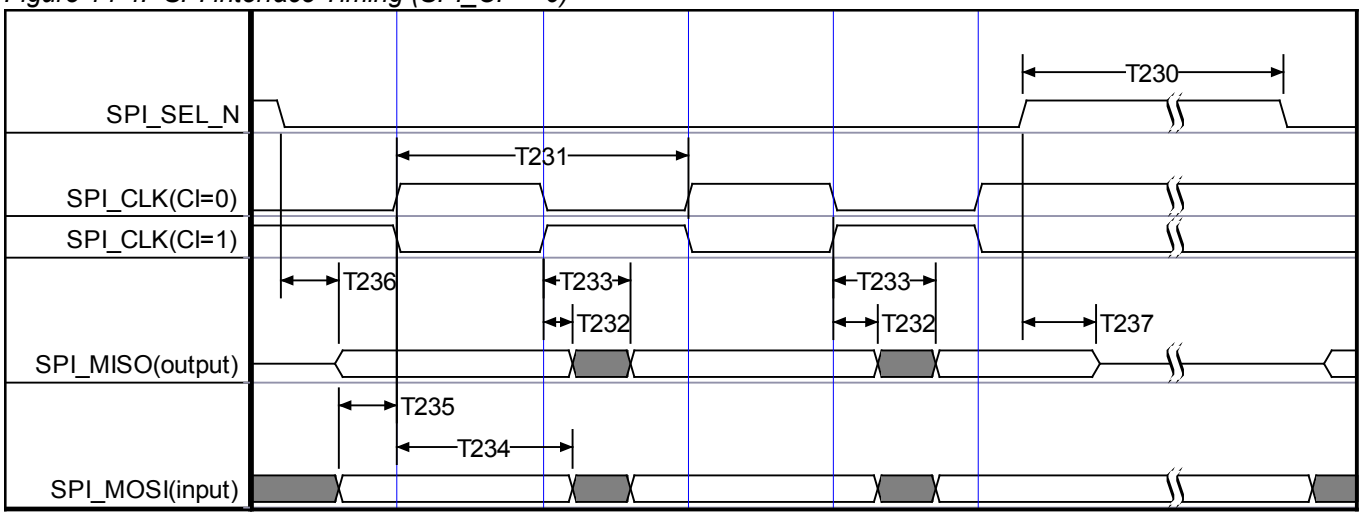
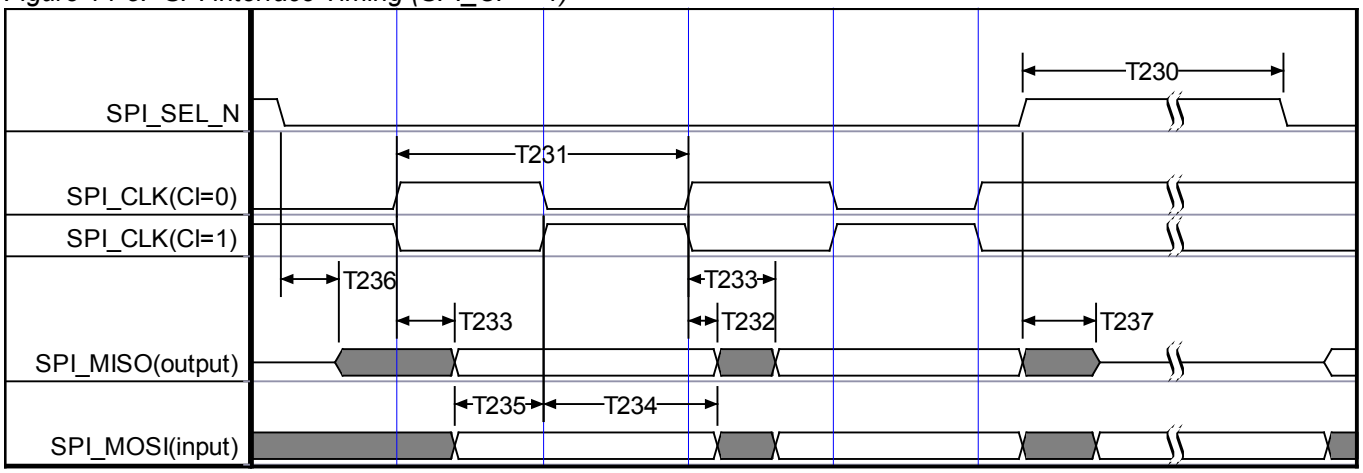



*Figure 14-5. SPI interface Timing (SPI_CP = 1)*




## 14.3 SDRAM Interface Timing

**Table 14-4. SDRAM Interface AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| SD_CLK to SD_CS_N, SD_RAS_N, SD_CAS_N, SD_WE_N, SD_DQM[3:0], SD_A[11:0], SD_BA[1:0] Output Hold | T51 | 1.9 | | | ns |
| SD_CLK to SD_CS_N, SD_RAS_N, SD_CAS_N, SD_WE_N, SD_DQM[3:0], SD_A[11:0], SD_BA[1:0] Output Valid | T52 | | | 8 | ns |
| SD_CLK to SD_D[31:0] Output Hold | T59 | 2 | | | ns |
| SD_CLK to SD_D[31:0] Output Valid | T60 | | | 8 | ns |
| SD_D[31:0] Input Setup Prior to SD_CLK | T69 | 4 | | | ns |
| SD_D[31:0] Input Hold After SD_CLK | T70 | 1 | | | ns |

**NOTE:** The output timing specified assumes 30 pF load.

*Figure 14-6. SDRAM Interface Write Cycle Timing*

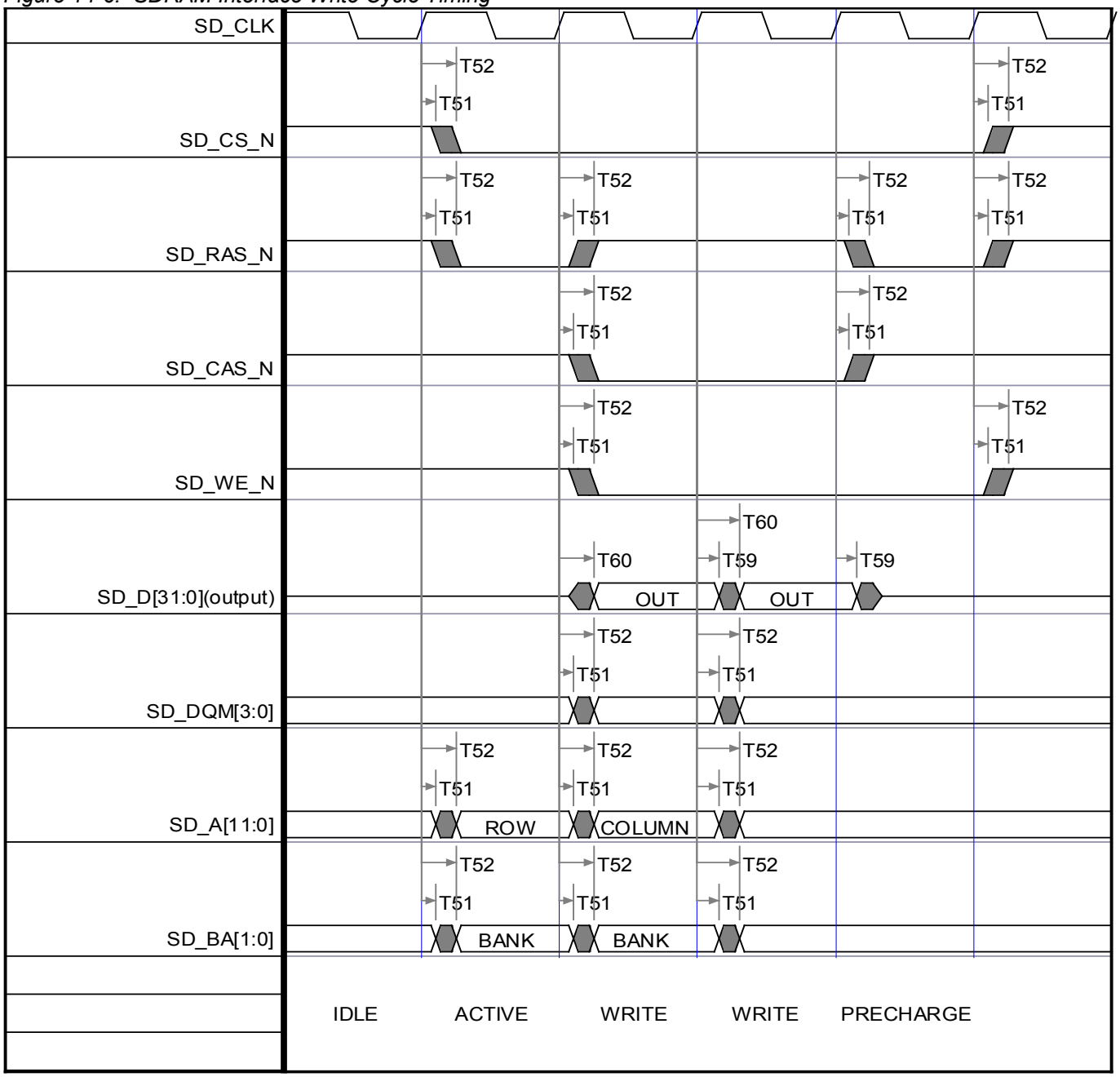
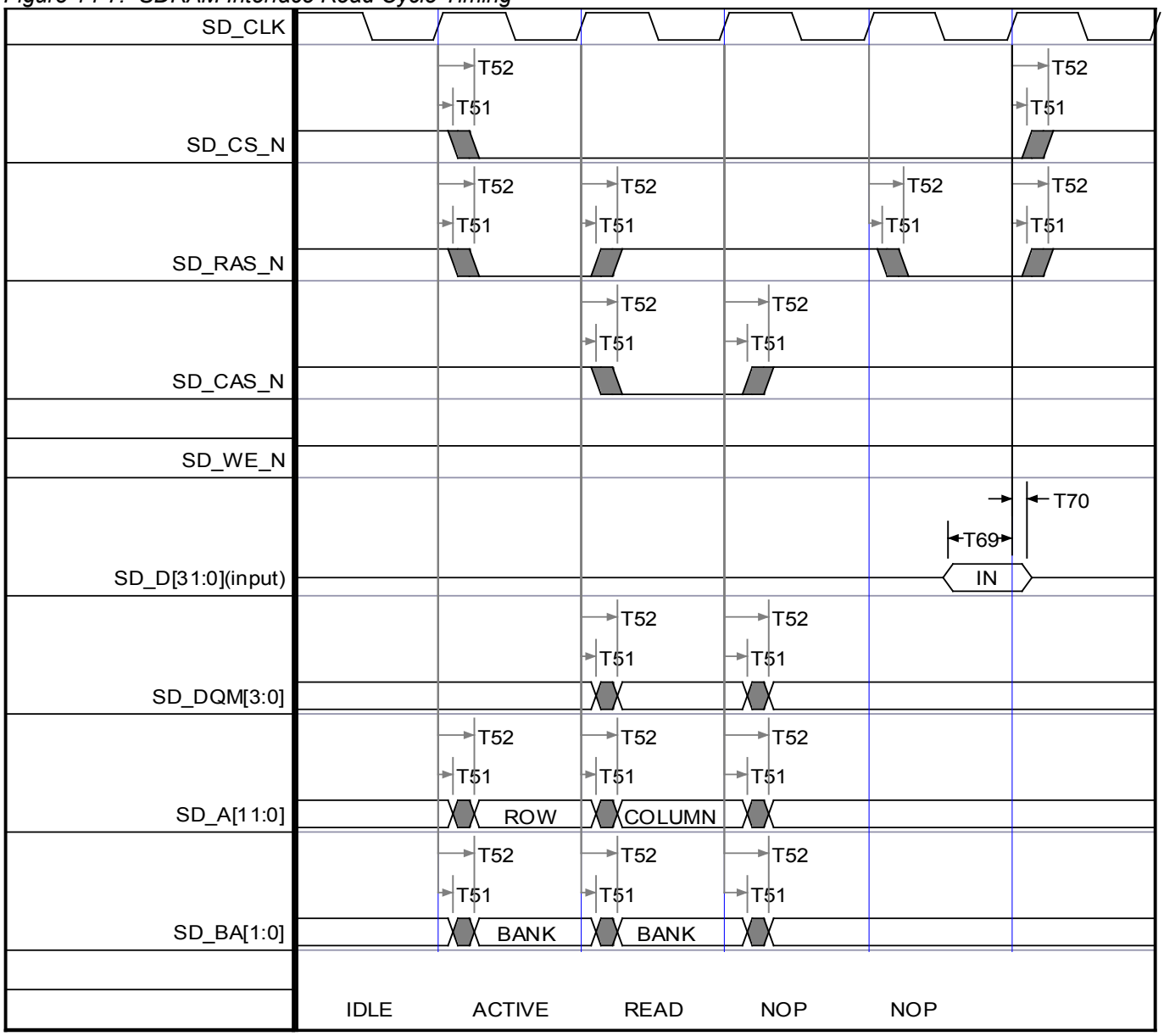


*Figure 14-7. SDRAM Interface Read Cycle Timing*



## 14.4 TDM-over-Packet TDM Interface Timing

**Table 14-5. TDMoP TDM Interface AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| TDMn_TX_SYNC, TDMn_TX_MF_CD, TDMn_RX, TDMn_RX_SYNC, TDMn_RSIG_RTS Input Setup Prior to TDMn_TCLK for E1/T1/Serial Interface | T101 | 1.8 | | | ns |
| TDMn_TX_SYNC, TDMn_TX_MF_CD, TDMn_RX, TDMn_RX_SYNC, TDMn_RSIG_RTS Input Hold After TDMn_TCLK for E1/T1/Serial Interface | T102 | 1.1 | | | ns |
| TDMn_TCLK to TDMn_TX, TDMn_TSIG_CTS Output Hold for E1/T1/Serial Interface | T103 | 2.8 | | | ns |
| TDMn_TCLK to TDMn_TX, TDMn_TSIG_CTS Output Valid for E1/T1/Serial Interface | T104 | | | 13.3 | ns |
| TDM1_TCLK to TDM1_TX Output Hold for High Speed \Interface | T103 | 4.5 (Note 1) | | | ns |
| TDM1_TCLK to TDM1_TX Output Valid for High Speed \Interface | T104 | | | 12.5 (Note 1) | ns |
| TDMn_RX, TDMn_RX_SYNC, TDMn_RSIG_RTS Input Setup Prior to TDMn_RCLK for E1/T1/Serial Interface | T109 | 1.8 | | | ns |
| TDMn_RX, TDMn_RX_SYNC, TDMn_RSIG_RTS Input Hold After TDMn_RCLK for E1/T1/Serial Interface | T110 | 0 | | | ns |
| TDM1_RX Input Setup Prior to TDM1_RCLK for High Speed Interface | T109 | 1.8 | | | ns |
| TDM1_RX Input Hold After TDM1_RCLK for High Speed Interface | T110 | 1.1 | | | ns |

**NOTES:**
1. The output timing specified for TDM1_TX assumes 20 pF load.

**Table 14-6. TDMoP TDM Clock AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| TDMn_TCLK Frequency for E1 Interface | T100 | | 2.048 | | MHz |
| TDMn_TCLK Frequency for T1 Interface | T100 | | 1.544 | | MHz |
| TDMn_RCLK, TDMn_TCLK Frequency for Serial Interface | T106 | 16k | | 4.65M | Hz |
| TDM1_RCLK, TDM1_TCLK Frequency for High Speed Interface | T106 | 16k | | 51.84M | Hz |
| TDMn_RCLK, TDMn_TCLK Duty Cycle for 1/T1 Serial Interface | T107 | 40 | | 60 | % |
| TDM1_RCLK, TDM1_TCLK Duty Cycle for High Speed Interface | T107 | 40 | | 60 | % |

**NOTE:** The output timing specified for TDM interfaces assumes 30 pF load.

*Figure 14-8. TDMoP TDM Timing, One-Clock Mode (Two_clocks=0, Tx_sample=1)*

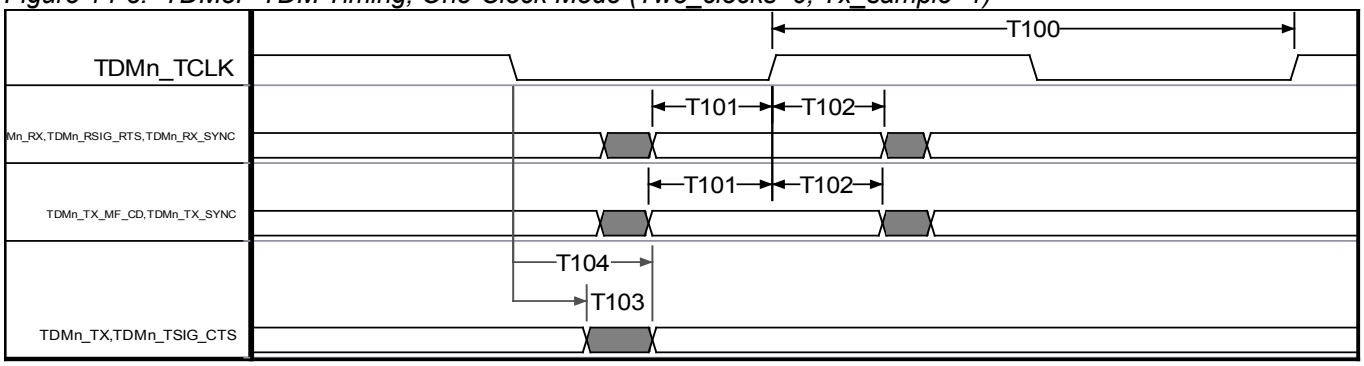


*Figure 14-9. TDMoP TDM Timing, One Clock Mode (Two_clocks=0, Tx_sample=0)*

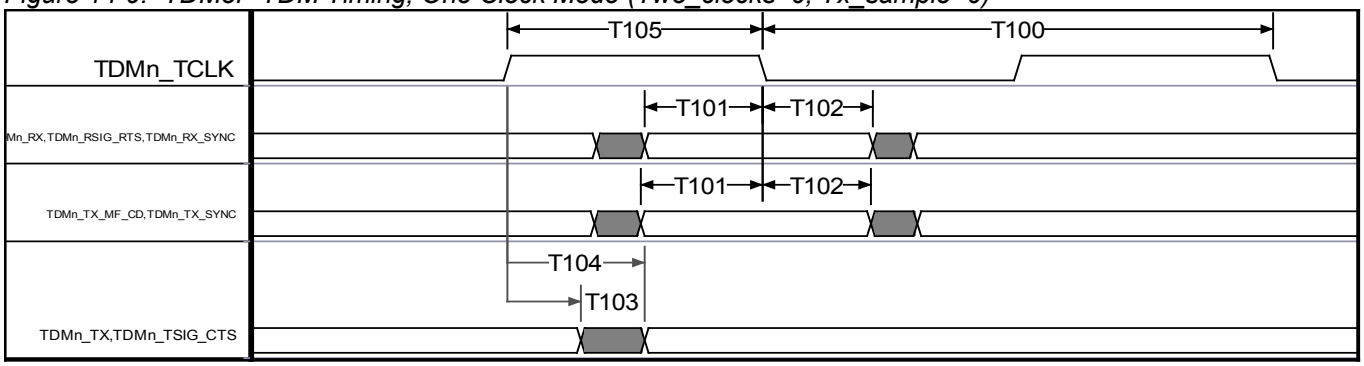



*Figure 14-10. TDMoP TDM Timing, Two Clock Mode (Two_clocks=1, Tx_sample=1, Rx_sample=1)*


*Figure 14-11. TDMoP TDM Timing, Two Clocks Mode (Two_clocks=1, Tx_sample=0, Rx_sample=0)*

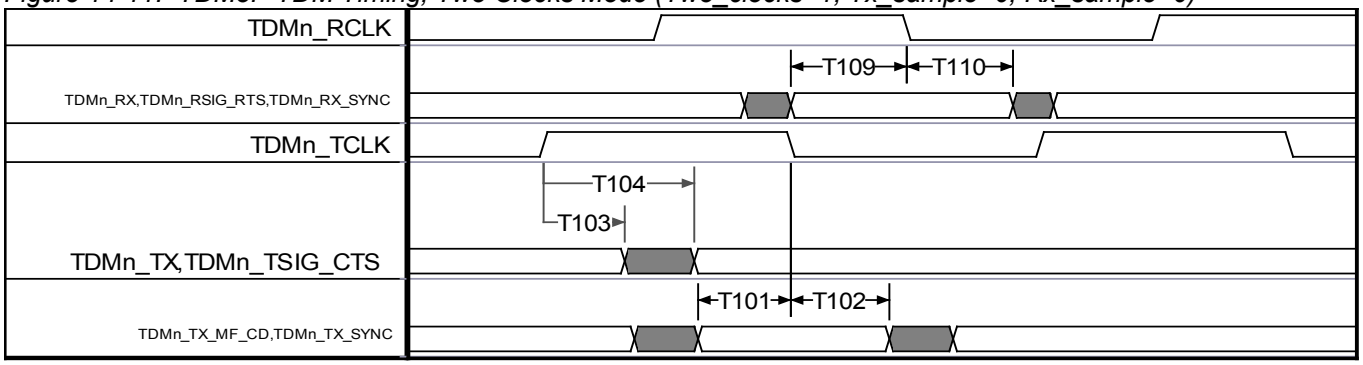


*Figure 14-12. TDMoP TDM Timing, Two Clocks Mode (Two_clocks=1, Tx_sample=0, Rx_sample=1)*

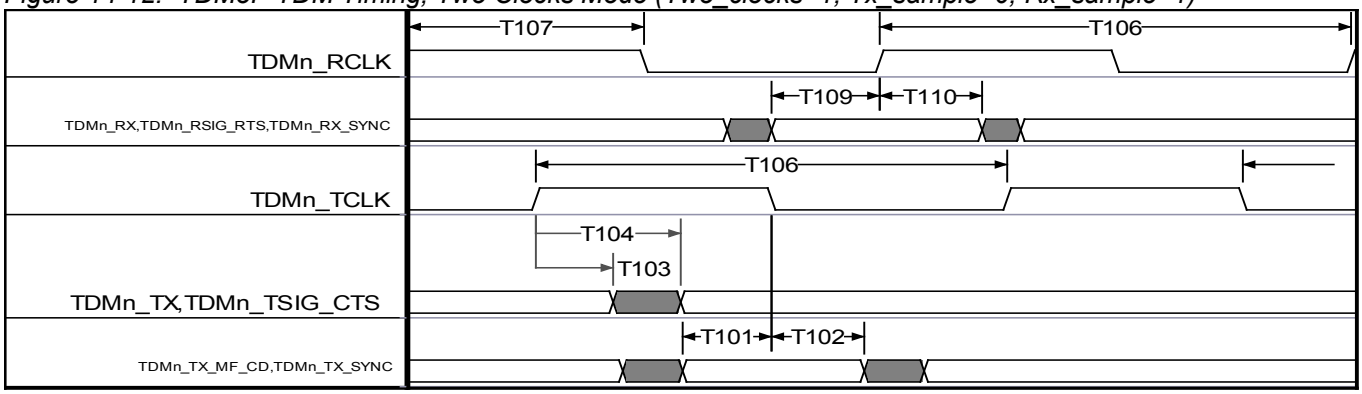
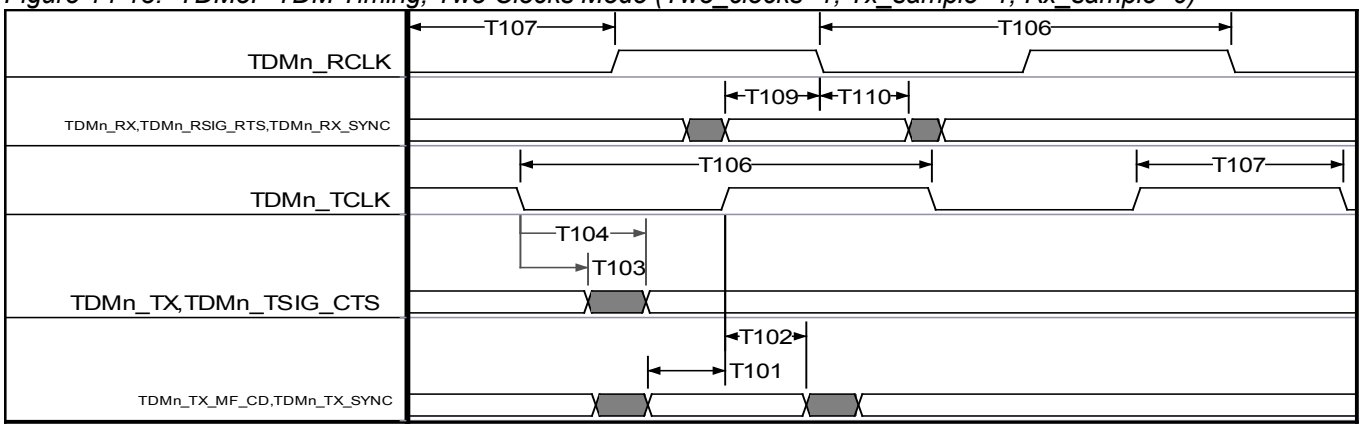



*Figure 14-13. TDMoP TDM Timing, Two Clocks Mode (Two_clocks=1, Tx_sample=1, Rx_sample=0)*



## 14.5 Ethernet MII/RMII/SSMII Interface Timing

**Table 14-7. MII Management Interface AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| MDC Period (Note 1) | T150 | | 320 | | ns |
| MDC to MDIO Output Hold (Note 1) | T151 | 10 | | | ns |
| MDC to MDIO Output Valid (Note 1) | T152 | | | 180 | ns |
| MDIO Input Setup Prior to MDC Rising | T153 | 20 | | | ns |
| MDIO Input Hold After MDC Rising | T154 | 0 | | | ns |

**NOTES:**
1. Valid for 50 MHz CLK_SYS and MDC_frequency = 0x02.

*Figure 14-14. MII Management Interface Timing*

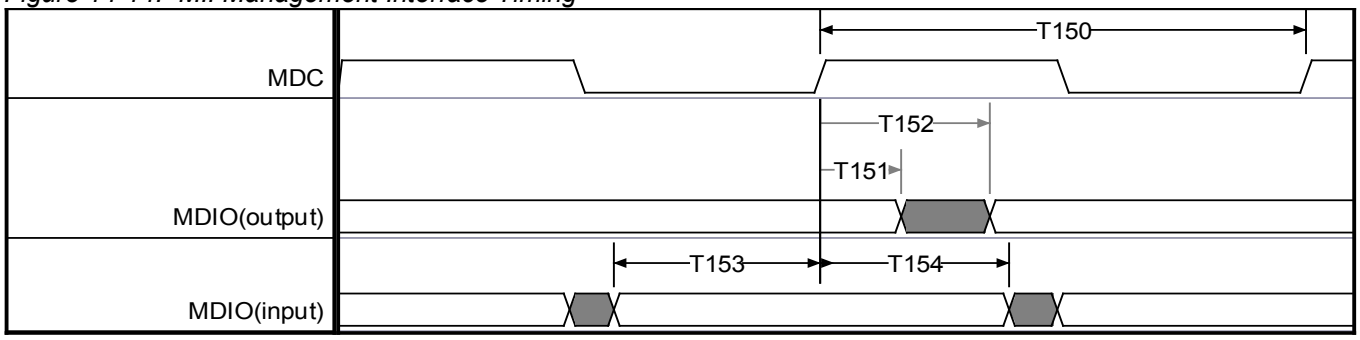



**Table 14-8. MII Interface AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| CLK_MII_TX Rising to MII_TXD, MII_TX_ERR, MII_TX_EN Output Hold | T156 | 0 | | | ns |
| CLK_MII_TX Rising to MII_TXD, MII_TX_ERR, MII_TX_EN Output Valid | T157 | | | 25 | ns |
| MII_RXD, MII_RX_DV, MII_RX_ERR Input Setup Prior to CLK_MII_RX Rising | T159 | 10 | | | ns |
| MII_RXD, MII_RX_DV, MII_RX_ERR Input Hold After to CLK_MII_RX Rising | T160 | 0 | | | ns |

**Table 14-9. MII Clock Timing**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| CLK_MII_TX Frequency | T158 | | 25 | | MHz |
| CLK_MII_RX Frequency | T158 | | 25 | | MHz |
| CLK_MII_TX Duty Cycle | T180 | 40 | | 60 | % |
| CLK_MII_RX duty Cycle | T180 | 40 | | 60 | % |

*Figure 14-15. MII Interface Output Signal Timing*

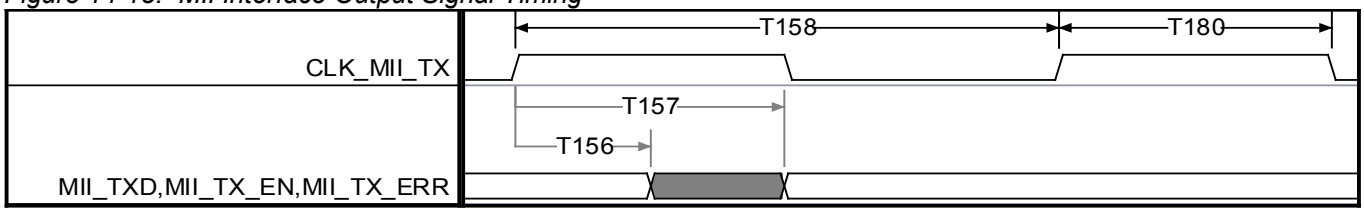


*Figure 14-16. MII Interface Input Signal Timing*

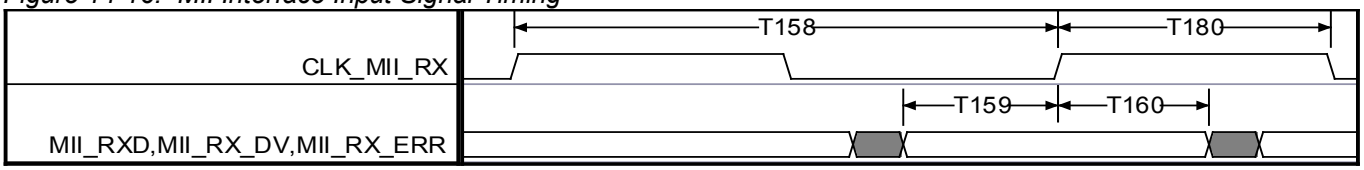



**Table 14-10. RMII Interface AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| CLK_MII_TX Rising to MII_TXD[3:2], MII_TX_EN Output Hold | T162 | 2 | | | ns |
| CLK_MII_TX Rising to MII_TXD[3:2], MII_TX_EN Output Valid | T163 | | | 13.5 | ns |
| MII_RXD[3:2], MII_RX_DV, MII_RX_ERR Input Setup Prior to CLK_MII_TX Rising | T164 | 7 | | | ns |
| MII_RXD(3:2], MII_RX_DV, MII_RX_ERR Input Hold After CLK_MII_TX Rising | T165 | 0 | | | ns |

**Table 14-11. RMII Clock Timing**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| CLK_MII_TX Frequency | T161 | | 50 | | MHz |
| CLK_MII_TX Duty Cycle | T183 | 40 | | 60 | % |

*Figure 14-17. RMII Interface Output Signal Timing*

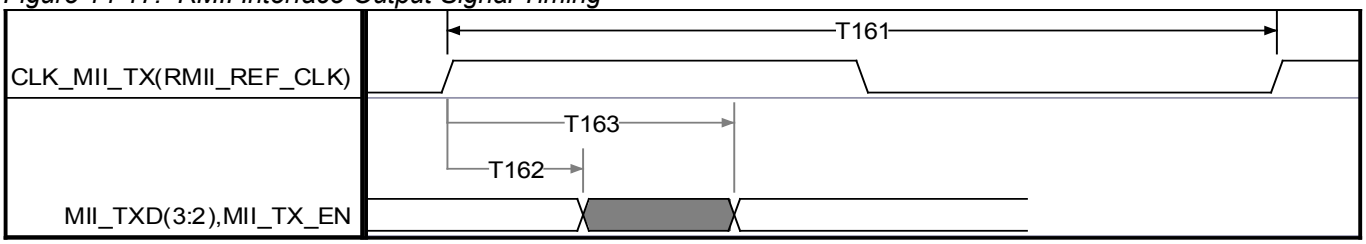



*Figure 14-18. RMII Interface Input Signal Timing*

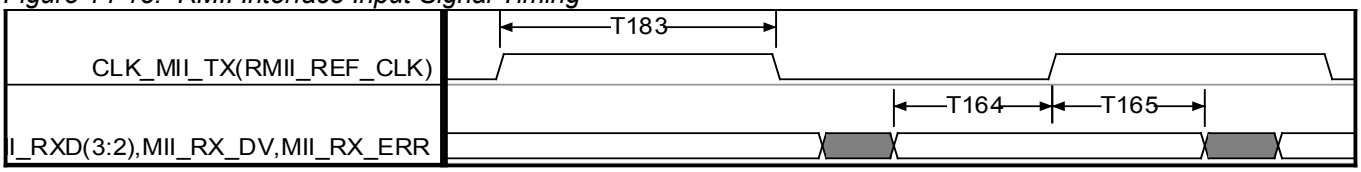



**Table 14-12. SSMII Interface AC Characteristics**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| CLK_SSMII_TX Rising to MII_TXD[1:0] Output | T172 | 1.5 | | 5 | ns |
| MII_RXD[1:0] Input Setup Prior to CLK_MII_RX Rising | T175 | 1.5 | | | ns |
| MII_RXD[1:0] Input Hold After CLK_MII_RX Rising | T176 | 1.3 | | | ns |

**Table 14-13. SSMII Clock Timing**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| CLK_SSMII_TX Frequency | T171 | | 125 | | MHz |
| CLK_SSMII_TX Duty Cycle | T189 | 40 | | 60 | % |
| CLK_MII_RX Frequency | T171 | | 125 | | MHz |
| CLK_MII_RX Duty Cycle | T189 | 40 | | 60 | % |

*Figure 14-19. SSMII Interface Output Signal Timing*

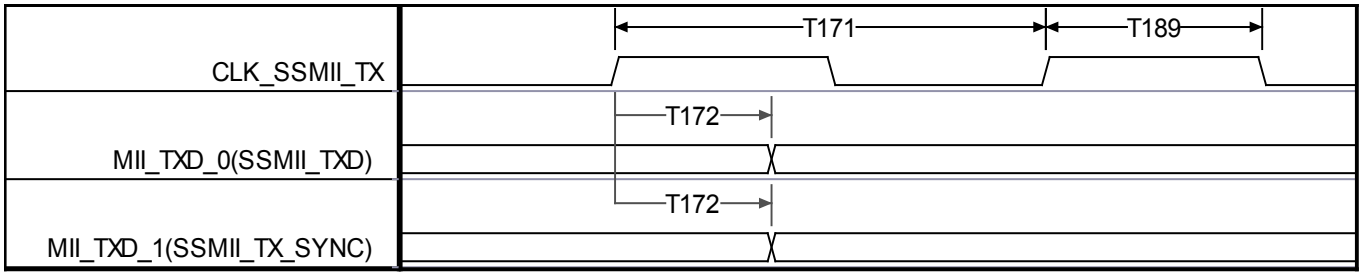
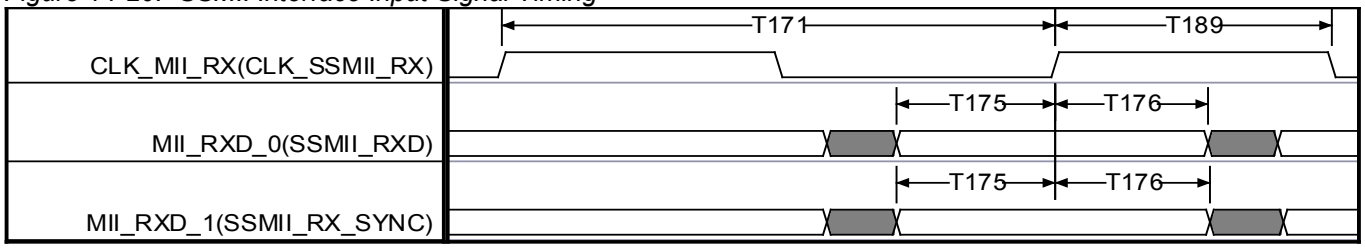



*Figure 14-20. SSMII Interface Input Signal Timing*




**NOTES FOR SECTION 14.5:**
1. The output timing specified for MII/RMII/SSMII interfaces assumes 20pf load for MII_TXD[3:0], MII_TX_EN, and MII_TX_ERR.
2. The output timing specified for MII/RMII/SSMII interfaces assumes 30pf load for MDC and MDIO.
3. The output timing specified for SSMII interface assumes 25pf load for CLK_SSMII_TX.

## 14.6 CLAD and System Clock Timing

**Table 14-14. CLAD1 and CLAD2 Input Clock Specifications**

| PARAMETER | MIN | TYP | MAX | UNITS | ACCURACY |
|---|---|---|---|---|---|
| CLK_SYS Frequency | | 25 or 50 | | MHz | ±50ppm |
| CLK_SYS Duty Cycle | 40 | | 60 | % | |
| CLK_HIGH Frequency | | 10.00<br>19.44<br>38.88<br>77.76 | | MHz | Traceable to Stratum 3E or higher |
| CLK_HIGH Duty Cycle | 40 | | 60 | % | |
| MCLK Frequency | | 1.544<br>2.048 | | MHz | ±32ppm<br>±50ppm |
| MCLK Duty Cycle | 40 | | 60 | % | |
| CLK_SYS Frequency | | 25<br>50<br>75 | | MHz | ±50ppm |
| CLK_SYS Duty Cycle | 40 | | 60 | % | |

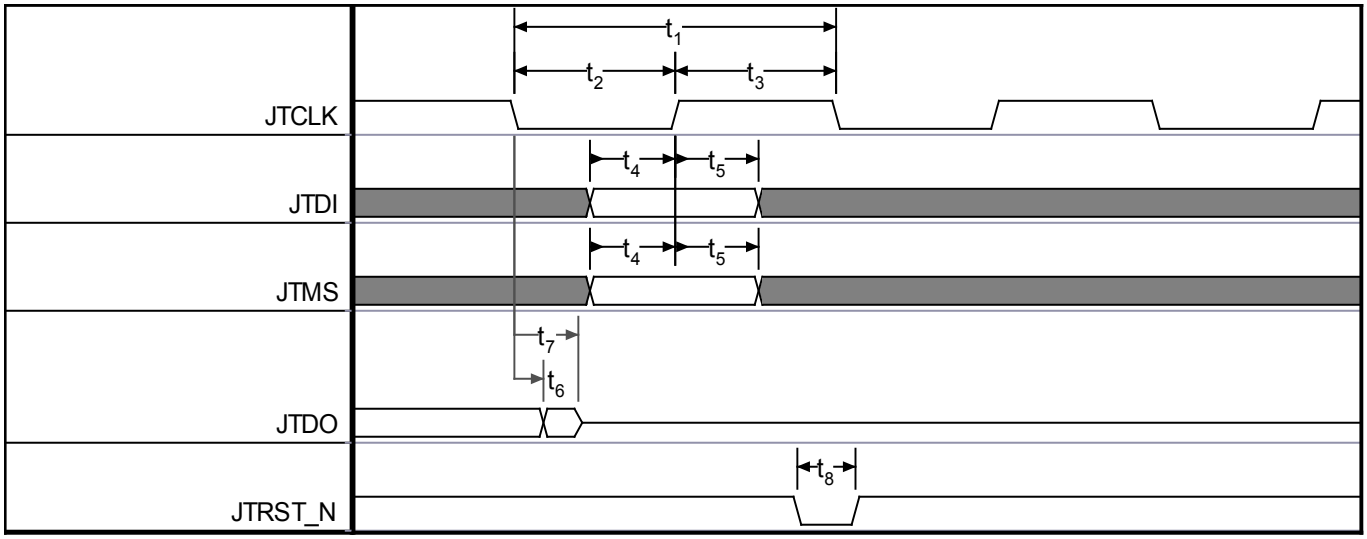
## 14.7 JTAG Interface Timing

**Table 14-15. JTAG Interface Timing**

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS | NOTES |
|---|---|---|---|---|---|---|
| JTCLK Clock Period | t1 | | 1000 | | ns | |
| JTCLK Clock High / Low Time | t2 / t3 | 100 | 500 | | ns | 1 |
| JTCLK to JTDI, JTMS Setup Time | t4 | 5 | | | ns | |
| JTCLK to JTDI, JTMS Hold Time | t5 | 2 | | | ns | |
| JTCLK to JTDO Delay | t6 | 2 | | 50 | ns | |
| JTCLK to JTDO Hi-Z Delay | t7 | 2 | | 50 | ns | 2 |
| JTRST_N Width Low Time | t8 | 100 | | | | |

**NOTES:**
1. Clock can be stopped high or low.
2. Not tested during production test.

*Figure 14-21. JTAG Interface Timing Diagram*



# 15. Applications

## 15.1 Connecting a Serial Interface Transceiver

Figure 15-1 below shows the connection of one port of a DS34S10x chip to a serial interface transceiver such as V.35 or RS-530. The figure shows one port in a DCE (Data Communications Equipment) application. All other ports can be connected in the same way.

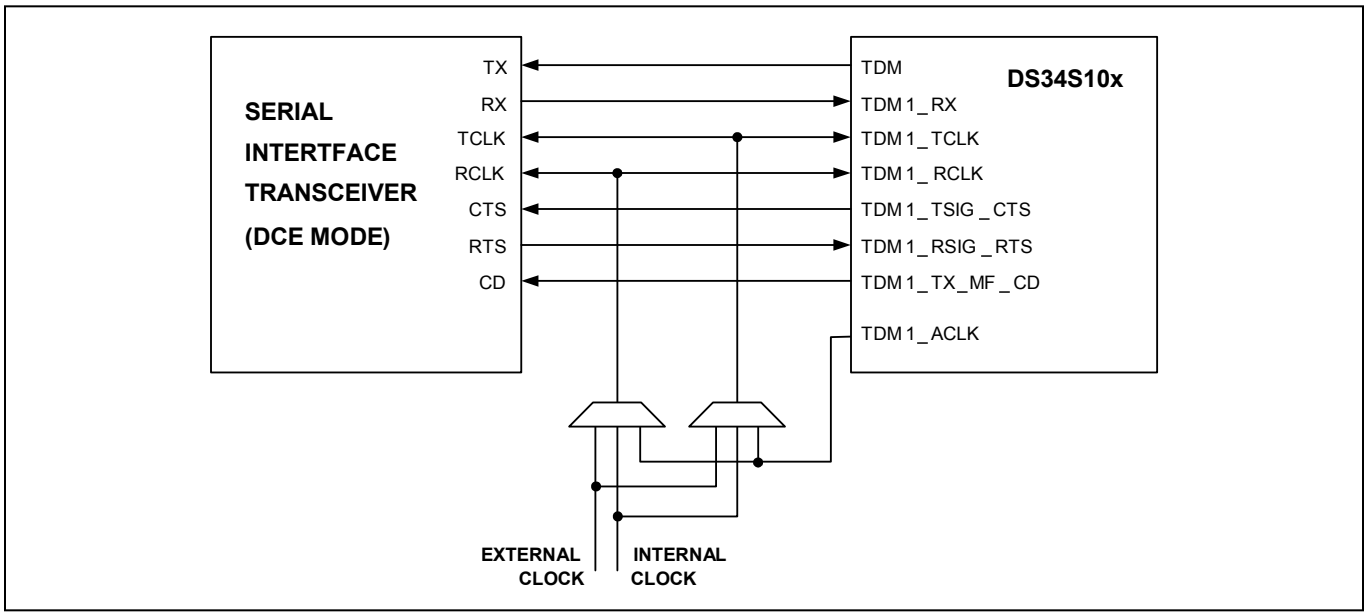
Each direction (Tx and Rx) has its own clock. However, TDM1_RCLK is optional, as the DS34S10x chip may work in one clock mode (GCR1.CLKMODE=0) in which both directions are clocked by TDM1_TCLK. The clock source of TDM1_RCLK or TDM1_TCLK can be:

- Internal (from the local oscillator)
- External
- Recovered from the packet network (provided by the chip on TDM1_ACLK).

The control input signal TDMn_RSIG_RTS does not affect the data reception, but its value can be read by the CPU from register field Port[n]_stat_reg1.RTS.

The TDMn_TSIG_CTS and TDMn_TX_MF_CD outputs can be controlled by software using registers fields CTS and CD in the Port[n]_cfg_reg register.

*Figure 15-1. Connecting Port 1 to a Serial Transceiver*



## 15.2 Connecting an Ethernet PHY or MAC

The figures below show the connection of the Ethernet port to a PHY or MAC device, in MII, RMII, and SSMII modes.

*Figure 15-2. Connecting the Ethernet Port to a PHY in MII Mode*

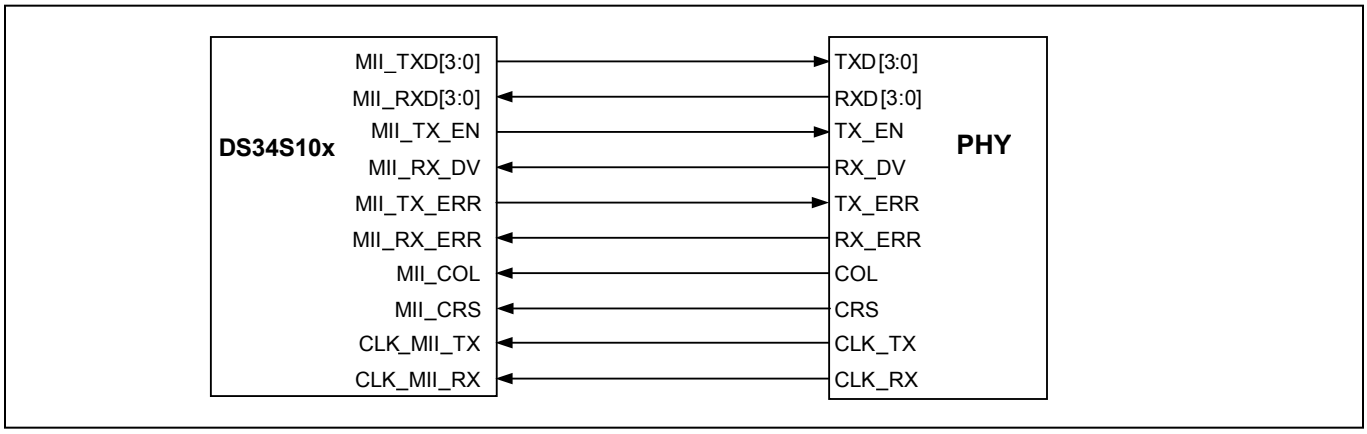



*Figure 15-3. Connecting the Ethernet Port to a MAC in MII Mode*

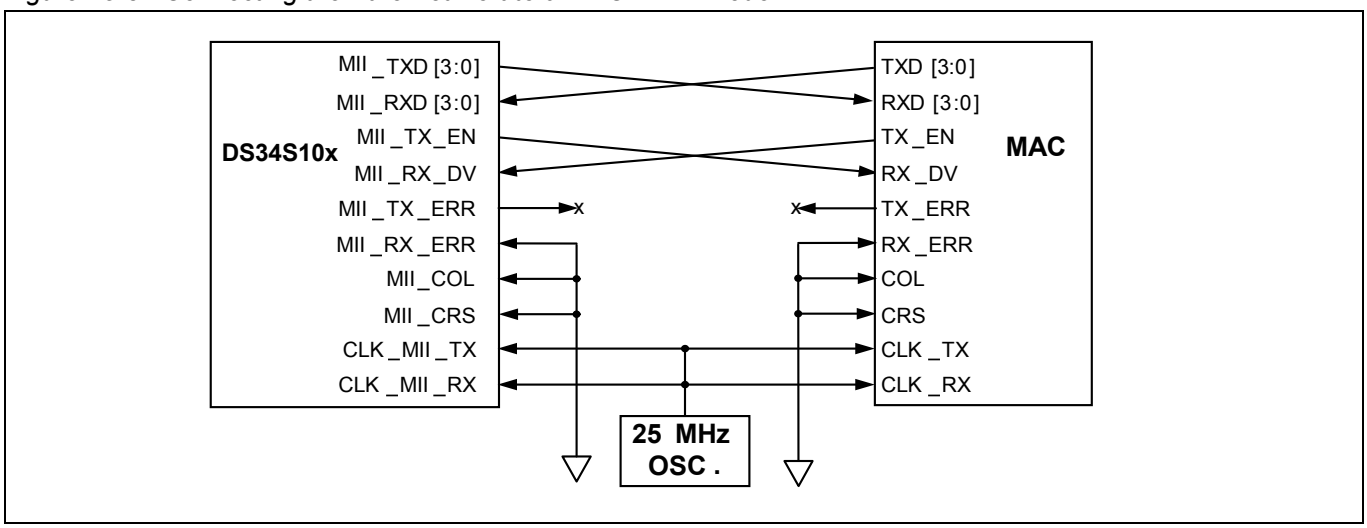



*Figure 15-4. Connecting the Ethernet Port to a PHY in RMII Mode*

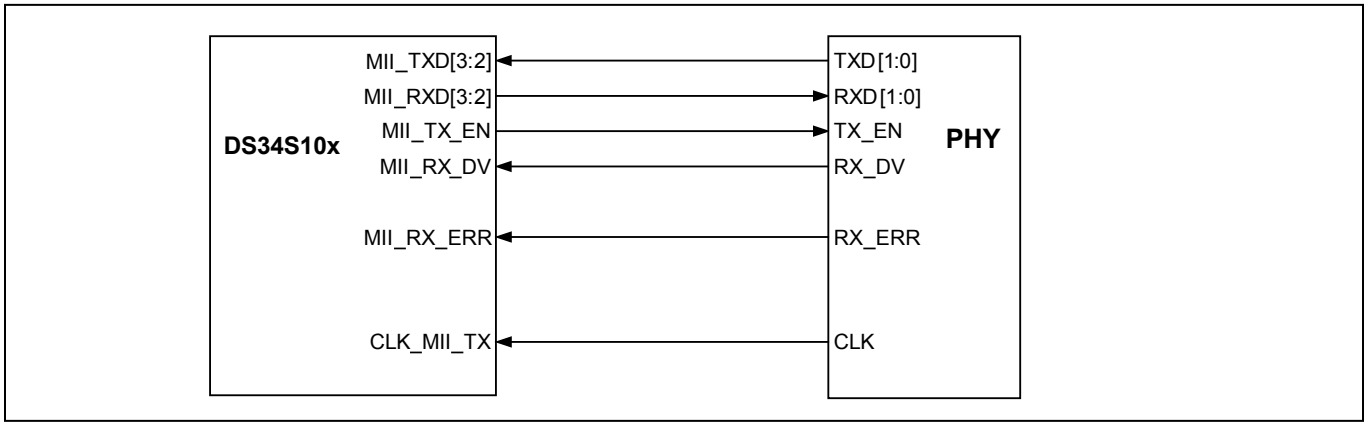


*Figure 15-5. Connecting the Ethernet Port to a MAC in RMII Mode*

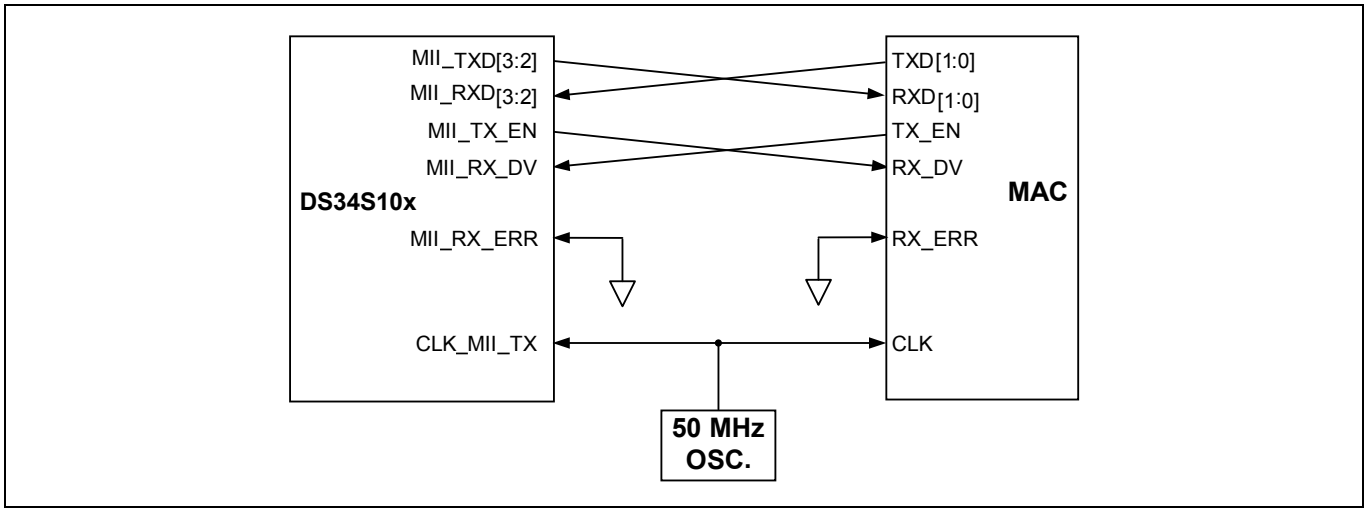



*Figure 15-6. Connecting the Ethernet Port to a PHY in SSMII Mode*

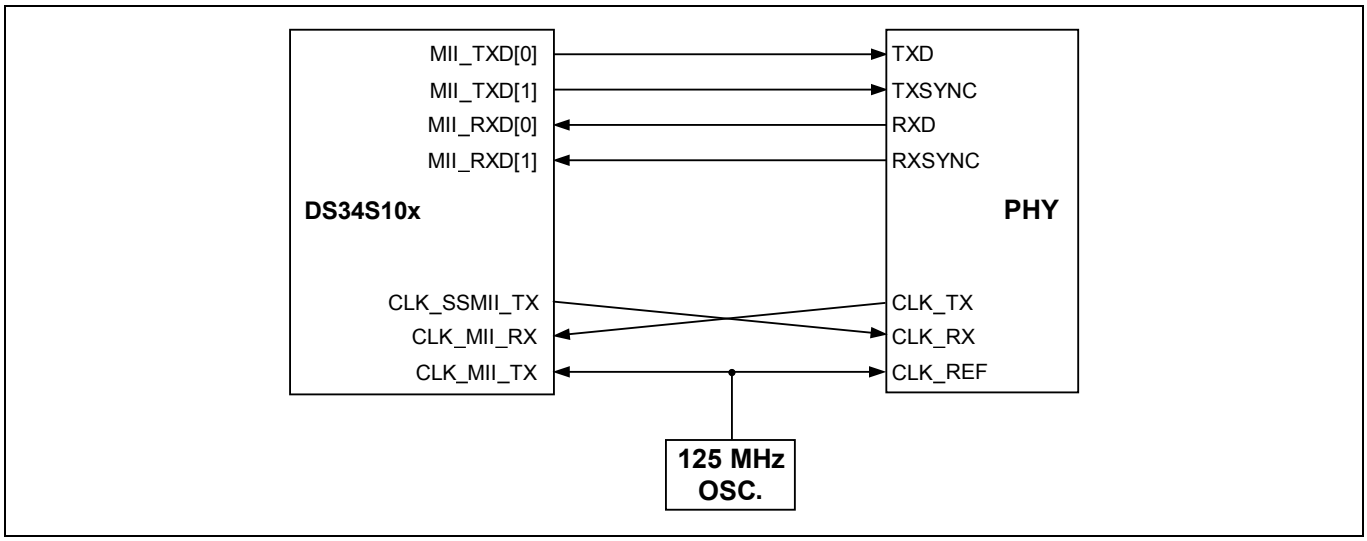
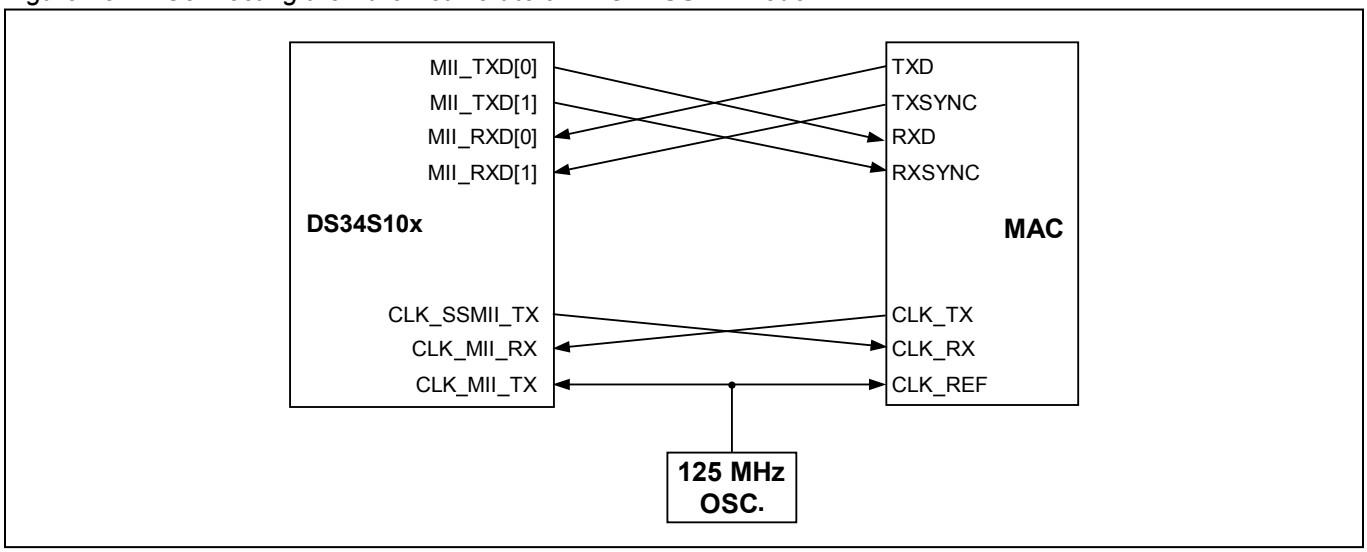



*Figure 15-7. Connecting the Ethernet Port to a MAC in SSMII Mode*



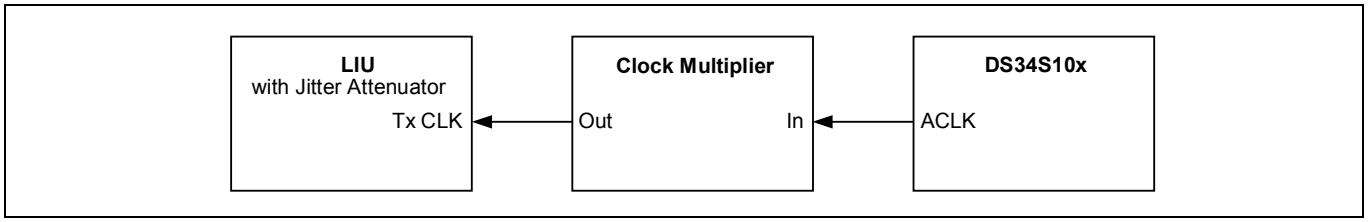
For the applications above, apply the following layout considerations:

- Provide termination on all high-speed interface signals and clock lines.
- Provide impedance matching on long traces to prevent reflections.
- Keep the clock traces away from all other signals to minimize mutual interference.
- In RMII mode, a very low skew clock buffer/driver is recommended to maximize the timing budget. In this mode it is recommended to keep all traces as short as possible.
- In SSMII mode there are two clock signals, one for each direction (Rx and Tx), routed together with the sync and data signals. Since the delay between the clock and these signals is lower, the designer can apply a longer trace delay in this mode. Keep data/sync traces and clock traces at the same length to maximize the timing budget.

## 15.3 Implementing Clock Recovery in High Speed Applications

For the high-speed interface (up to 51.84 MHz), an external clock multiplier and jitter attenuator are needed. Clock recovery in high-speed applications is depicted below:

*Figure 15-8. External Clock Multiplier for High Speed Applications*




The clock multiplier converts the low speed clock at ACLK to a clock at the frequency of the emulated high-speed circuit. The multiplication factor in the external clock multiplier must be 12 for an E3 or T3 interface and 10 for an STS-1 interface. The clock multiplier should be tuned to add minimal jitter. The jitter attenuator can be part of the LIU or an independent component.

## 15.4 Connecting a Motorola MPC860 Processor

The device is easily connected to a Motorola MPC860 processor by means of the MPC860 GPCM (General Purpose Chip Select Machine) module.

### 15.4.1 Connecting the Bus Signals

Since the MPC860 address bus MSb is always 0 while the DS34S10x address bus LSb is always 0, the signal order can be reversed as shown in the following figures.

*Figure 15-9. 32-Bit CPU Bus Connections*

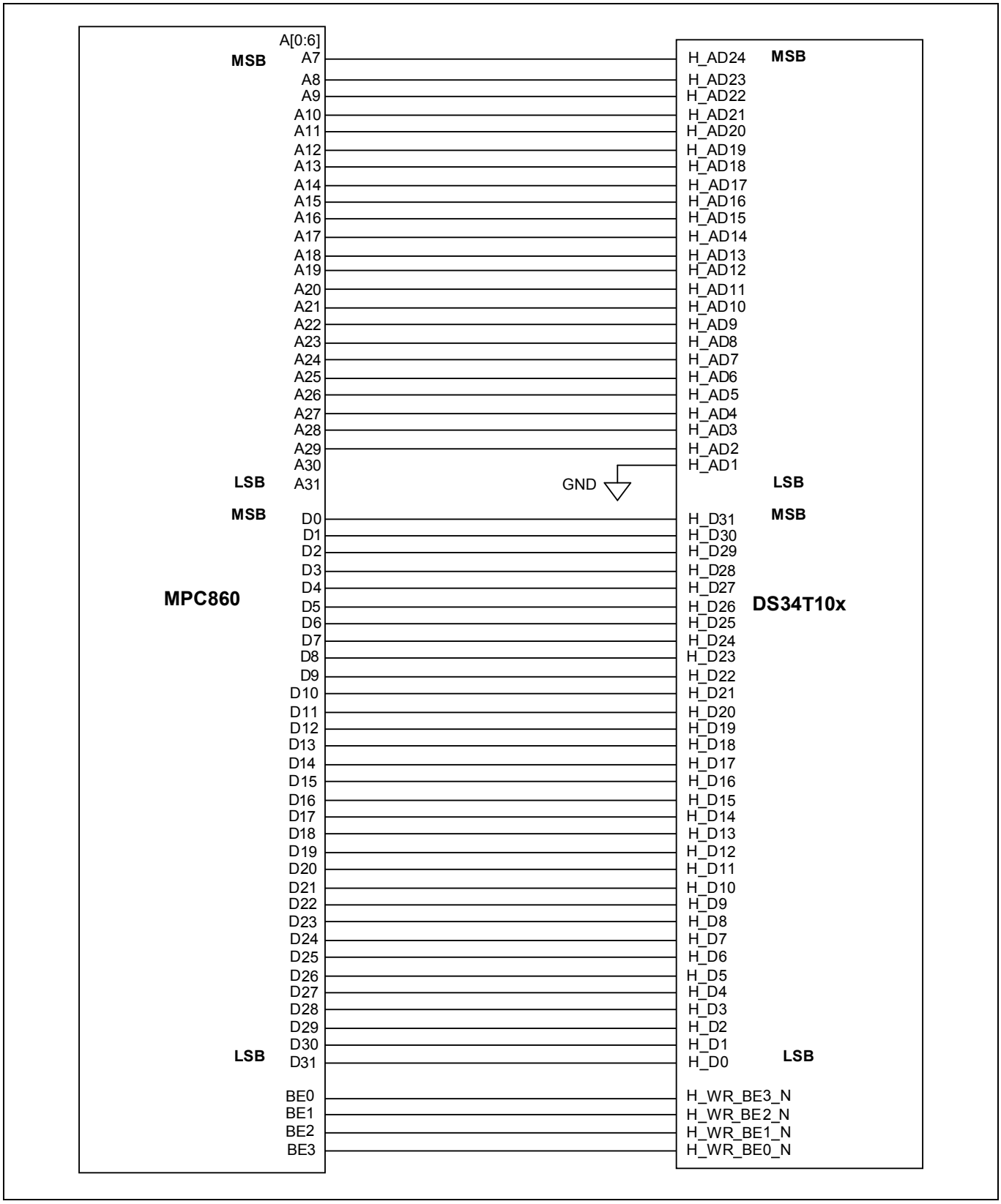
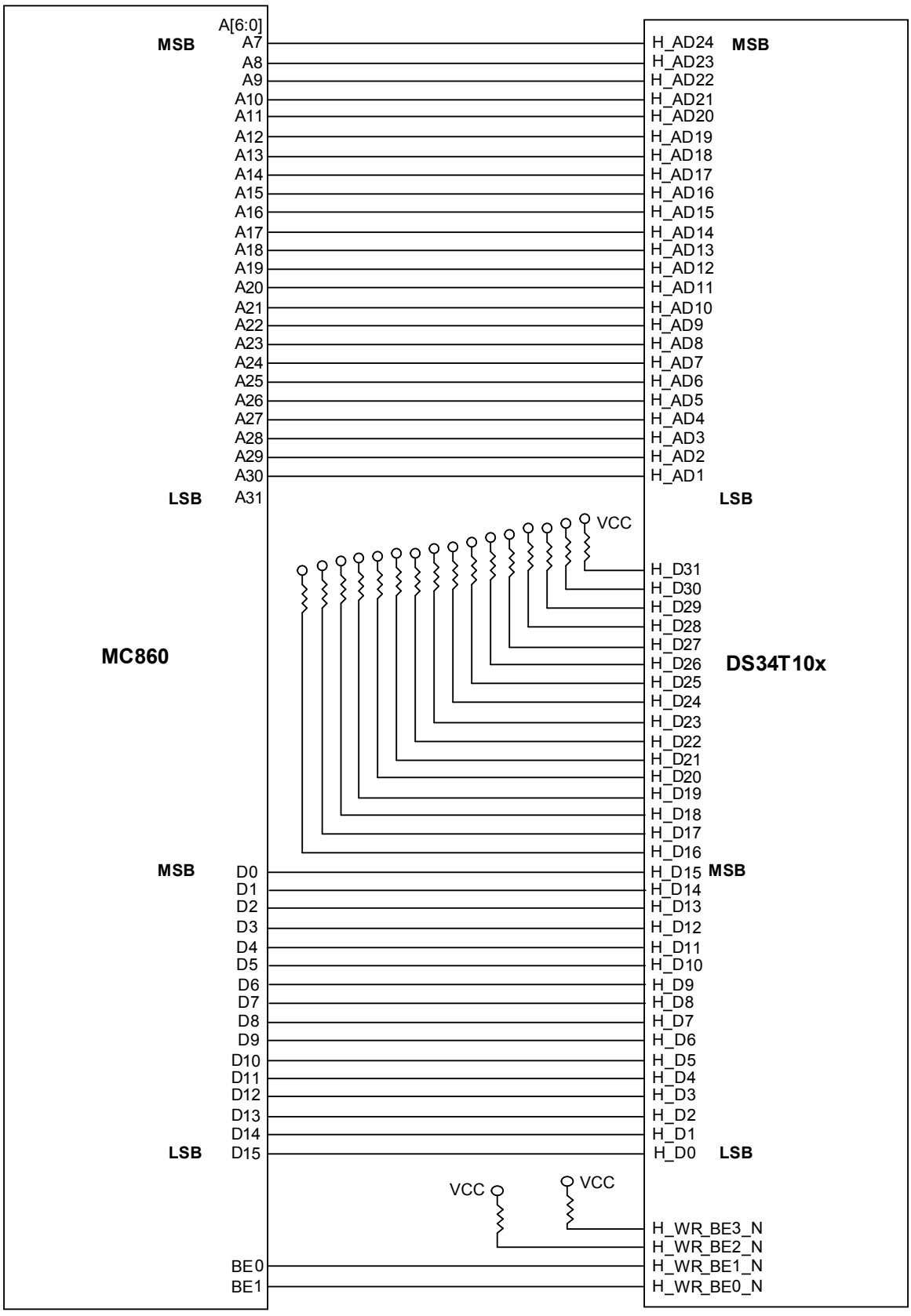


*Figure 15-10. 16-Bit CPU Bus Connections*



### 15.4.2 Connecting the H_READY_N Signal

The H_READY_N output should be connected to the MPC860 TA input. The CPU bus operates asynchronously. The TA of the MPC860 is a synchronous input (i.e., needs to meet set-up and hold times). The designer should synchronize H_READY_N to the MPC860 clock by means of a CPLD, which uses the MPC860 reference clock. The internal logic in the CPLD also uses the MPC860 CS (chip select) output. Both the H_READY_N output and the MPC860 TA input should have a 1kΩ pull-up resistor.

*Figure 15-11. Connecting the H_READY_N Signal to the MPC860 TA Pin*

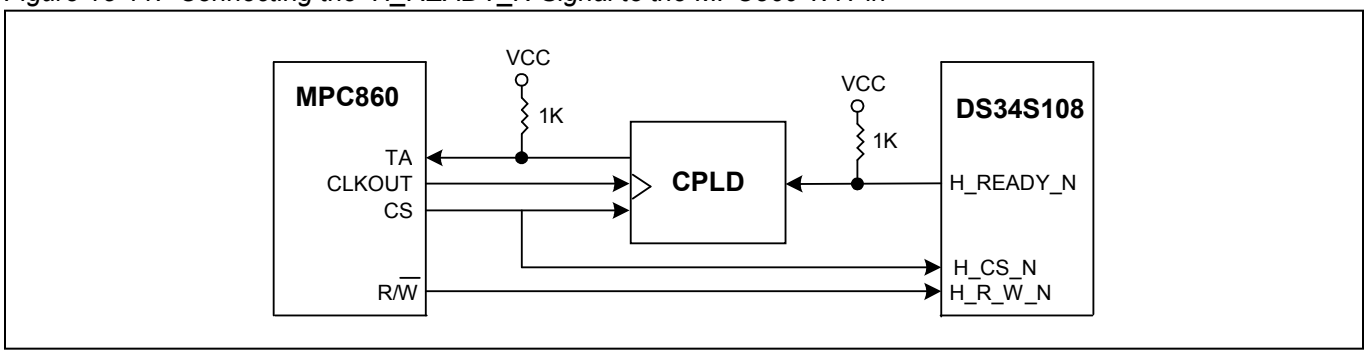
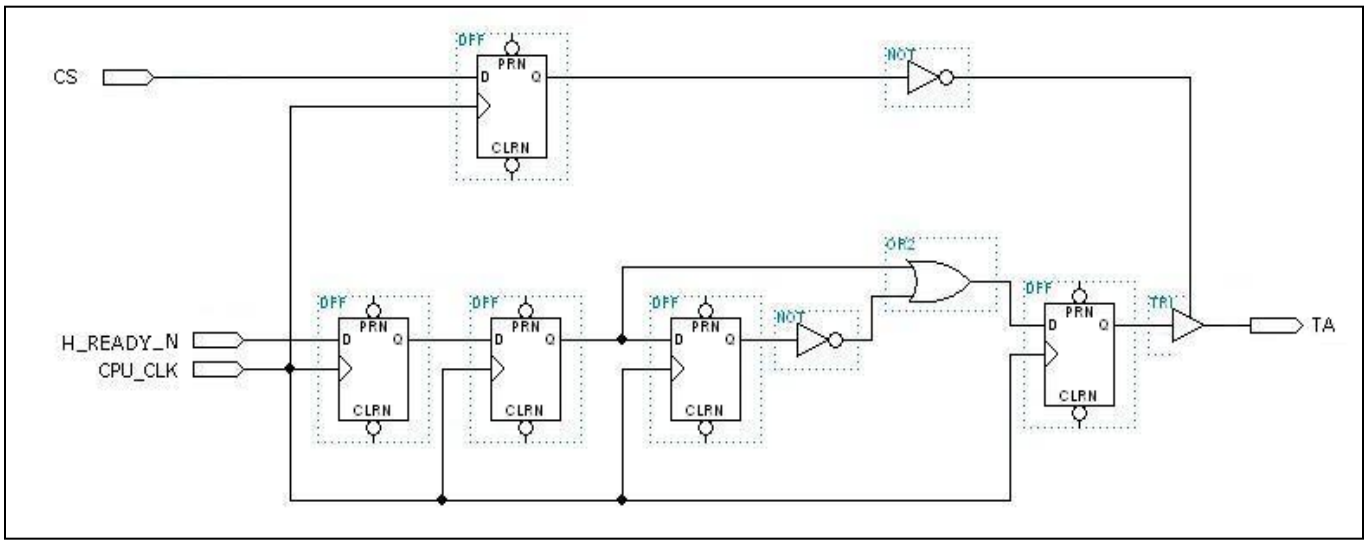



*Figure 15-12. Internal CPLD Logic to Synchronize H_READY_N to the MPC860 Clock*




Another alternative for connecting the H_READY_N signal is using the MPC860 UPM. In this option the H_READY_N output should be connected to the MPC860 UPWAIT (GPL4) signal, and no external timing adjustment is needed. The H_READY_N output should have a 1kΩ pull-up resistor. Refer to the MPC860 user manual for additional details.

## 15.5 Working in SPI Mode

The following table shows the I/O connections for operating in SPI mode.

**Table 15-1. SPI Mode I/O Connections**

| Signal name | Connect to | Comments |
|---|---|---|
| H_CPU_SPI_N | VSS (logic 0) | Selects SPI mode. |
| DAT_32_16_N | DVDDIO or DVSS | Ignored in SPI mode. |
| H_CS_N | DVDDIO or DVSS | Ignored in SPI mode. |
| H_AD[24:1] | DVDDIO or DVSS | Ignored in SPI mode. |
| H_D[31:1] | DVDDIO or DVSS | Ignored in SPI mode. |
| H_D[0] / SPI_MISO | Master MISO | |
| H_WR_BE0_N / SPI_CLK | Master SPI clock | |
| H_WR_BE1_N / SPI_MOSI | Master MOSI | |
| H_WR_BE2_N / SPI_SEL_N | Master SPI select | |
| H_WR_BE3_N / SPI_CI | DVDDIO (logic 1) or DVSS (logic 0) | According to required SPI mode |
| H_R_W_N / SPI_CP | DVDDIO (logic 1) or DVSS (logic 0) | According to required SPI mode |

## 15.6 Connecting SDRAM Devices

The following table lists suggested SDRAM devices to use in conjunction with the DS34S10x devices.

**Table 15-2. List of Suggested SDRAM Devices**

| Vendor | 64 Mb Device | 128 Mb Device |
|---|---|---|
| Micron | MT48LC2M32B2TG-6 | MT48LC4M32B2TG-6 |
| Samsung | K4S643232H-TC/L60 | K4S283232E-TC/L60 |
| Hynix | HY57V653220BTC-6 or HY57V643220CT-6 | HY57V283220T-6 |
| Elpida | N/A | EDS1232AATA-60 |
| Winbond | W986432DH-6 | N/A |
| ICSI | IC42S32200/L-6T or IC42S32200/L-6TI | N/A |
| ISSI | IS42S32200C1-6T | IS42S32400B-6T |

When connecting the device to an external SDRAM, it is advised to connect SD_CLK through a serial termination resistor.

When connecting the device to a 64 Mb external SDRAM, it is advised to connect SD_A[11] through a serial resistor to the SDRAM “NC” pin that is used for address pin A11 for a 128 Mb SDRAM. In this way, the 64Mb SDRAM could be replaced by a 128 Mb SDRAM later, if needed.

# 16. PIN ASSIGNMENTS

## 16.1 Board Design for Multiple DS34S101/2/4 Devices

The DS34S101, DS34S102 and DS34S104 require the same footprint on the board. It is recommended that boards be design to support the use of higher port-count devices in a lower port-count socket. If this is done, unused inputs, input/outputs, and outputs must be biased appropriately. Generally, unused inputs are tied directly to the ground plane, unused outputs are not connected, and unused input/outputs are tied to ground through a 10kΩ resistor. Unused inputs with internal pull-ups or pull-downs are not connected. Table 16-1 designates how each ball on the package should be connected to implement a common board design. Shading indicates balls for the unused inputs, input/outputs, and outputs of higher port-count devices.

If a common board design is not done, the balls for the unused inputs, input/outputs, and outputs need not be connected, and the stuffing of higher port-count devices into a lower port-count socket is not recommended.

Note: When a higher port-count device is used in a socket, the BSDL file of the higher port-count device must be used. BSDL files are available from the factory upon request.

**Table 16-1. Common Board Design Connections for DS34S101/2/4 (Sorted by Signal Name)**

| Ball | DS34S104 Socket | DS34S102 Socket | DS34S101 Socket |
|---|---|---|---|
| R8 | ACVDD1 | ACVDD1 | ACVDD1 |
| T8 | ACVDD2 | ACVDD2 | ACVDD2 |
| P8 | ACVSS1 | ACVSS1 | ACVSS1 |
| T9 | ACVSS2 | ACVSS2 | ACVSS2 |
| M14 | CLK_CMN | CLK_CMN | CLK_CMN |
| P9 | CLK_HIGH | CLK_HIGH | CLK_HIGH |
| A16 | CLK_MII_RX | CLK_MII_RX | CLK_MII_RX |
| D15 | CLK_MII_TX | CLK_MII_TX | CLK_MII_TX |
| E15 | CLK_SSMII_TX | CLK_SSMII_TX | CLK_SSMII_TX |
| T12 | CLK_SYS | CLK_SYS/SCCLK | CLK_SYS/SCCLK |
| R9 | CLK_SYS_S | CLK_SYS_S | CLK_SYS_S |
| M4 | DAT_32_16_N | DAT_32_16_N | DAT_32_16_N |
| F10 | DVDDC | DVDDC | DVDDC |
| F11 | DVDDC | DVDDC | DVDDC |
| F6 | DVDDC | DVDDC | DVDDC |
| F7 | DVDDC | DVDDC | DVDDC |
| F8 | DVDDC | DVDDC | DVDDC |
| F9 | DVDDC | DVDDC | DVDDC |
| M10 | DVDDC | DVDDC | DVDDC |
| M11 | DVDDC | DVDDC | DVDDC |
| M6 | DVDDC | DVDDC | DVDDC |
| M7 | DVDDC | DVDDC | DVDDC |
| M8 | DVDDC | DVDDC | DVDDC |
| M9 | DVDDC | DVDDC | DVDDC |
| G12 | DVDDIO | DVDDIO | DVDDIO |
| H12 | DVDDIO | DVDDIO | DVDDIO |
| J12 | DVDDIO | DVDDIO | DVDDIO |
| J5 | DVDDIO | DVDDIO | DVDDIO |
| K12 | DVDDIO | DVDDIO | DVDDIO |
| K5 | DVDDIO | DVDDIO | DVDDIO |

| Ball | DS34S104 Socket | DS34S102 Socket | DS34S101 Socket |
|---|---|---|---|
| L12 | DVDDIO | DVDDIO | DVDDIO |
| L5 | DVDDIO | DVDDIO | DVDDIO |
| M5 | DVDDIO | DVDDIO | DVDDIO |
| N10 | DVDDIO | DVDDIO | DVDDIO |
| N7 | DVDDIO | DVDDIO | DVDDIO |
| N8 | DVDDIO | DVDDIO | DVDDIO |
| N9 | DVDDIO | DVDDIO | DVDDIO |
| G10 | DVSS | DVSS | DVSS |
| G11 | DVSS | DVSS | DVSS |
| G6 | DVSS | DVSS | DVSS |
| G7 | DVSS | DVSS | DVSS |
| G8 | DVSS | DVSS | DVSS |
| G9 | DVSS | DVSS | DVSS |
| H10 | DVSS | DVSS | DVSS |
| H11 | DVSS | DVSS | DVSS |
| H5 | DVSS | DVSS | DVSS |
| H6 | DVSS | DVSS | DVSS |
| H7 | DVSS | DVSS | DVSS |
| H8 | DVSS | DVSS | DVSS |
| H9 | DVSS | DVSS | DVSS |
| J10 | DVSS | DVSS | DVSS |
| J11 | DVSS | DVSS | DVSS |
| J6 | DVSS | DVSS | DVSS |
| J7 | DVSS | DVSS | DVSS |
| J8 | DVSS | DVSS | DVSS |
| J9 | DVSS | DVSS | DVSS |
| K10 | DVSS | DVSS | DVSS |
| K11 | DVSS | DVSS | DVSS |
| K6 | DVSS | DVSS | DVSS |
| K7 | DVSS | DVSS | DVSS |
| K8 | DVSS | DVSS | DVSS |
| K9 | DVSS | DVSS | DVSS |
| L10 | DVSS | DVSS | DVSS |
| L11 | DVSS | DVSS | DVSS |
| L6 | DVSS | DVSS | DVSS |
| L7 | DVSS | DVSS | DVSS |
| L8 | DVSS | DVSS | DVSS |
| L9 | DVSS | DVSS | DVSS |
| R13 | H_AD[1] | H_AD[1] | H_AD[1] |
| C10 | H_AD[10] | H_AD[10] | H_AD[10] |
| E12 | H_AD[11] | H_AD[11] | H_AD[11] |
| A12 | H_AD[12] | H_AD[12] | H_AD[12] |
| T15 | H_AD[13] | H_AD[13] | H_AD[13] |
| C12 | H_AD[14] | H_AD[14] | H_AD[14] |
| D12 | H_AD[15] | H_AD[15] | H_AD[15] |
| T16 | H_AD[16] | H_AD[16] | H_AD[16] |
| B13 | H_AD[17] | H_AD[17] | H_AD[17] |
| R14 | H_AD[18] | H_AD[18] | H_AD[18] |

| Ball | DS34S104 Socket | DS34S102 Socket | DS34S101 Socket |
|---|---|---|---|
| B10 | H_AD[19] | H_AD[19] | H_AD[19] |
| D13 | H_AD[2] | H_AD[2] | H_AD[2] |
| P14 | H_AD[20] | H_AD[20] | H_AD[20] |
| A10 | H_AD[21] | H_AD[21] | H_AD[21] |
| B11 | H_AD[22] | H_AD[22] | H_AD[22] |
| N14 | H_AD[23] | H_AD[23] | H_AD[23] |
| A11 | H_AD[24] | H_AD[24] | H_AD[24] |
| P13 | H_AD[3] | H_AD[3] | H_AD[3] |
| D10 | H_AD[4] | H_AD[4] | H_AD[4] |
| E13 | H_AD[5] | H_AD[5] | H_AD[5] |
| D11 | H_AD[6] | H_AD[6] | H_AD[6] |
| N13 | H_AD[7] | H_AD[7] | H_AD[7] |
| A13 | H_AD[8] | H_AD[8] | H_AD[8] |
| T14 | H_AD[9] | H_AD[9] | H_AD[9] |
| L4 | H_CPU_SPI_N | H_CPU_SPI_N | H_CPU_SPI_N |
| E11 | H_CS_N | H_CS_N | H_CS_N |
| K13 | H_D[0]/SPI_MISO | H_D[0]/SPI_MISO | H_D[0]/SPI_MISO |
| M12 | H_D[1] | H_D[1] | H_D[1] |
| M13 | H_D[10] | H_D[10] | H_D[10] |
| P16 | H_D[11] | H_D[11] | H_D[11] |
| K14 | H_D[12] | H_D[12] | H_D[12] |
| M15 | H_D[13] | H_D[13] | H_D[13] |
| J14 | H_D[14] | H_D[14] | H_D[14] |
| M16 | H_D[15] | H_D[15] | H_D[15] |
| L14 | H_D[16] | H_D[16] | H_D[16] |
| L16 | H_D[17] | H_D[17] | H_D[17] |
| J15 | H_D[18] | H_D[18] | H_D[18] |
| K16 | H_D[19] | H_D[19] | H_D[19] |
| R15 | H_D[2] | H_D[2] | H_D[2] |
| N16 | H_D[20] | H_D[20] | H_D[20] |
| B12 | H_D[21] | H_D[21] | H_D[21] |
| J16 | H_D[22] | H_D[22] | H_D[22] |
| F12 | H_D[23] | H_D[23] | H_D[23] |
| F13 | H_D[24] | H_D[24] | H_D[24] |
| G13 | H_D[25] | H_D[25] | H_D[25] |
| H13 | H_D[26] | H_D[26] | H_D[26] |
| F14 | H_D[27] | H_D[27] | H_D[27] |
| G14 | H_D[28] | H_D[28] | H_D[28] |
| H14 | H_D[29] | H_D[29] | H_D[29] |
| L13 | H_D[3] | H_D[3] | H_D[3] |
| H16 | H_D[30] | H_D[30] | H_D[30] |
| H15 | H_D[31] | H_D[31] | H_D[31] |
| K15 | H_D[4] | H_D[4] | H_D[4] |
| P15 | H_D[5] | H_D[5] | H_D[5] |
| J13 | H_D[6] | H_D[6] | H_D[6] |
| N15 | H_D[7] | H_D[7] | H_D[7] |
| L15 | H_D[8] | H_D[8] | H_D[8] |
| R16 | H_D[9] | H_D[9] | H_D[9] |

| Ball | DS34S104 Socket | DS34S102 Socket | DS34S101 Socket |
|---|---|---|---|
| T13 | H_INT[0] | H_INT[0] | H_INT[0] |
| N12 | H_R_W_N/SPI_CP | H_R_W_N/SPI_CP | H_R_W_N/SPI_CP |
| R12 | H_READY_N | H_READY_N | H_READY_N |
| C13 | H_WR_BE0_N/SPI_CLK | H_WR_BE0_N/SPI_CLK | H_WR_BE0_N/SPI_CLK |
| P12 | H_WR_BE1_N/SPI_MOSI | H_WR_BE1_N/SPI_MOSI | H_WR_BE1_N/SPI_MOSI |
| E10 | H_WR_BE2_N/SPI_SEL_N | H_WR_BE2_N/SPI_SEL_N | H_WR_BE2_N/SPI_SEL_N |
| C11 | H_WR_BE3_N/SPI_CI | H_WR_BE3_N/SPI_CI | H_WR_BE3_N/SPI_CI |
| N11 | HIZ_N | HIZ_N | HIZ_N |
| R10 | JTCLK | JTCLK | JTCLK |
| P10 | JTDI | JTDI | JTDI |
| P11 | JTDO | JTDO | JTDO |
| T10 | JTMS | JTMS | JTMS |
| T11 | JTRST_N | JTRST_N | JTRST_N |
| N5 | MBIST_DONE | MBIST_DONE | MBIST_DONE |
| N6 | MBIST_EN | MBIST_EN | MBIST_EN |
| R4 | MBIST_FAIL | MBIST_FAIL | MBIST_FAIL |
| G16 | MDC | MDC | MDC |
| G15 | MDIO | MDIO | MDIO |
| B15 | MII_COL | MII_COL | MII_COL |
| C15 | MII_CRS | MII_CRS | MII_CRS |
| F16 | MII_RX_DV | MII_RX_DV | MII_RX_DV |
| A15 | MII_RX_ERR | MII_RX_ERR | MII_RX_ERR |
| B16 | MII_RXD[0] | MII_RXD[0] | MII_RXD[0] |
| C16 | MII_RXD[1] | MII_RXD[1] | MII_RXD[1] |
| D16 | MII_RXD[2] | MII_RXD[2] | MII_RXD[2] |
| E16 | MII_RXD[3] | MII_RXD[3] | MII_RXD[3] |
| D14 | MII_TX_EN | MII_TX_EN | MII_TX_EN |
| E14 | MII_TX_ERR | MII_TX_ERR | MII_TX_ERR |
| F15 | MII_TXD[0] | MII_TXD[0] | MII_TXD[0] |
| A14 | MII_TXD[1] | MII_TXD[1] | MII_TXD[1] |
| B14 | MII_TXD[2] | MII_TXD[2] | MII_TXD[2] |
| C14 | MII_TXD[3] | MII_TXD[3] | MII_TXD[3] |
| R11 | RST_SYS_N | RST_SYS_N | RST_SYS_N |
| J4 | SCEN | SCEN | SCEN |
| D2 | SD_A[0] | SD_A[0] | SD_A[0] |
| C3 | SD_A[1] | SD_A[1] | SD_A[1] |
| A4 | SD_A[10] | SD_A[10] | SD_A[10] |
| F2 | SD_A[11] | SD_A[11] | SD_A[11] |
| E1 | SD_A[2] | SD_A[2] | SD_A[2] |
| A3 | SD_A[3] | SD_A[3] | SD_A[3] |
| E2 | SD_A[4] | SD_A[4] | SD_A[4] |
| B3 | SD_A[5] | SD_A[5] | SD_A[5] |
| E3 | SD_A[6] | SD_A[6] | SD_A[6] |
| C4 | SD_A[7] | SD_A[7] | SD_A[7] |
| B4 | SD_A[8] | SD_A[8] | SD_A[8] |
| D3 | SD_A[9] | SD_A[9] | SD_A[9] |
| D1 | SD_BA[0] | SD_BA[0] | SD_BA[0] |
| A2 | SD_BA[1] | SD_BA[1] | SD_BA[1] |

| Ball | DS34S104 Socket | DS34S102 Socket | DS34S101 Socket |
|---|---|---|---|
| C1 | SD_CAS_N | SD_CAS_N | SD_CAS_N |
| B2 | SD_CLK | SD_CLK | SD_CLK |
| C2 | SD_CS_N | SD_CS_N | SD_CS_N |
| E4 | SD_D[0] | SD_D[0] | SD_D[0] |
| B5 | SD_D[1] | SD_D[1] | SD_D[1] |
| B6 | SD_D[10] | SD_D[10] | SD_D[10] |
| F3 | SD_D[11] | SD_D[11] | SD_D[11] |
| A6 | SD_D[12] | SD_D[12] | SD_D[12] |
| F4 | SD_D[13] | SD_D[13] | SD_D[13] |
| B8 | SD_D[14] | SD_D[14] | SD_D[14] |
| D7 | SD_D[15] | SD_D[15] | SD_D[15] |
| F5 | SD_D[16] | SD_D[16] | SD_D[16] |
| C7 | SD_D[17] | SD_D[17] | SD_D[17] |
| A7 | SD_D[18] | SD_D[18] | SD_D[18] |
| D8 | SD_D[19] | SD_D[19] | SD_D[19] |
| F1 | SD_D[2] | SD_D[2] | SD_D[2] |
| G2 | SD_D[20] | SD_D[20] | SD_D[20] |
| E8 | SD_D[21] | SD_D[21] | SD_D[21] |
| G3 | SD_D[22] | SD_D[22] | SD_D[22] |
| A8 | SD_D[23] | SD_D[23] | SD_D[23] |
| B9 | SD_D[24] | SD_D[24] | SD_D[24] |
| G4 | SD_D[25] | SD_D[25] | SD_D[25] |
| E9 | SD_D[26] | SD_D[26] | SD_D[26] |
| G5 | SD_D[27] | SD_D[27] | SD_D[27] |
| A9 | SD_D[28] | SD_D[28] | SD_D[28] |
| D9 | SD_D[29] | SD_D[29] | SD_D[29] |
| C6 | SD_D[3] | SD_D[3] | SD_D[3] |
| C9 | SD_D[30] | SD_D[30] | SD_D[30] |
| C8 | SD_D[31] | SD_D[31] | SD_D[31] |
| G1 | SD_D[4] | SD_D[4] | SD_D[4] |
| B7 | SD_D[5] | SD_D[5] | SD_D[5] |
| D5 | SD_D[6] | SD_D[6] | SD_D[6] |
| D6 | SD_D[7] | SD_D[7] | SD_D[7] |
| E7 | SD_D[8] | SD_D[8] | SD_D[8] |
| E5 | SD_D[9] | SD_D[9] | SD_D[9] |
| C5 | SD_DQM[0] | SD_DQM[0] | SD_DQM[0] |
| D4 | SD_DQM[1] | SD_DQM[1] | SD_DQM[1] |
| E6 | SD_DQM[2] | SD_DQM[2] | SD_DQM[2] |
| A5 | SD_DQM[3] | SD_DQM[3] | SD_DQM[3] |
| A1 | SD_RAS_N | SD_RAS_N | SD_RAS_N |
| B1 | SD_WE_N | SD_WE_N | SD_WE_N |
| H4 | STMD | STMD | STMD |
| R5 | TDM1_ACLK | TDM1_ACLK | TDM1_ACLK |
| T7 | TDM1_RCLK | TDM1_RCLK | TDM1_RCLK |
| P7 | TDM1_RSIG_RTS | TDM1_RSIG_RTS | TDM1_RSIG_RTS |
| T5 | TDM1_RX | TDM1_RX | TDM1_RX |
| P5 | TDM1_RX_SYNC | TDM1_RX_SYNC | TDM1_RX_SYNC |
| T6 | TDM1_TCLK | TDM1_TCLK | TDM1_TCLK |

| Ball | DS34S104 Socket | DS34S102 Socket | DS34S101 Socket |
|---|---|---|---|
| P6 | TDM1_TSIG_CTS | TDM1_TSIG_CTS | TDM1_TSIG_CTS |
| T4 | TDM1_TX | TDM1_TX | TDM1_TX |
| R7 | TDM1_TX_MF_CD | TDM1_TX_MF_CD | TDM1_TX_MF_CD |
| R6 | TDM1_TX_SYNC | TDM1_TX_SYNC | TDM1_TX_SYNC |
| R1 | TDM2_ACLK | TDM2_ACLK | NC |
| P4 | TDM2_RCLK | TDM2_RCLK | NC |
| T3 | TDM2_RSIG_RTS | TDM2_RSIG_RTS | NC |
| P2 | TDM2_RX | TDM2_RX | NC |
| T1 | TDM2_RX_SYNC | TDM2_RX_SYNC | NC |
| P3 | TDM2_TCLK | TDM2_TCLK | NC |
| T2 | TDM2_TSIG_CTS | TDM2_TSIG_CTS | NC |
| P1 | TDM2_TX | TDM2_TX | NC |
| R3 | TDM2_TX_MF_CD | TDM2_TX_MF_CD | NC |
| R2 | TDM2_TX_SYNC | TDM2_TX_SYNC | NC |
| M2 | TDM3_ACLK | NC | NC |
| M1 | TDM3_RCLK | NC | NC |
| N4 | TDM3_RSIG_RTS | NC | NC |
| L2 | TDM3_RX | NC | NC |
| N2 | TDM3_RX_SYNC | NC | NC |
| L3 | TDM3_TCLK | NC | NC |
| N3 | TDM3_TSIG_CTS | NC | NC |
| L1 | TDM3_TX | NC | NC |
| N1 | TDM3_TX_MF_CD | NC | NC |
| M3 | TDM3_TX_SYNC | NC | NC |
| J2 | TDM4_ACLK | NC | NC |
| J1 | TDM4_RCLK | NC | NC |
| K4 | TDM4_RSIG_RTS | NC | NC |
| H2 | TDM4_RX | NC | NC |
| K2 | TDM4_RX_SYNC | NC | NC |
| H3 | TDM4_TCLK | NC | NC |
| K3 | TDM4_TSIG_CTS | NC | NC |
| H1 | TDM4_TX | NC | NC |
| K1 | TDM4_TX_MF_CD | NC | NC |
| J3 | TDM4_TX_SYNC | NC | NC |

## 16.2 DS34S101 Pin Assignment

*Figure 16-1. DS34S101 Pin Assignment (TE-CSBGA Package)*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| A | SD_RAS_N | SD_BA[1] | SD_A[3] | SD_A[10] | SD_DQM[3] | SD_D[12] | SD_D[18] | SD_D[23] |
| B | SD_WE_N | SD_CLK | SD_A[5] | SD_A[8] | SD_D[1] | SD_D[10] | SD_D[5] | SD_D[14] |
| C | SD_CAS_N | SD_CS_N | SD_A[1] | SD_A[7] | SD_DQM[0] | SD_D[3] | SD_D[17] | SD_D[31] |
| D | SD_BA[0] | SD_A[0] | SD_A[9] | SD_DQM[1] | SD_D[6] | SD_D[7] | SD_D[15] | SD_D[19] |
| E | SD_A[2] | SD_A[4] | SD_A[6] | SD_D[0] | SD_D[9] | SD_DQM[2] | SD_D[8] | SD_D[21] |
| F | SD_D[2] | SD_A[11] | SD_D[11] | SD_D[13] | SD_D[16] | DVDDC | DVDDC | DVDDC |
| G | SD_D[4] | SD_D[20] | SD_D[22] | SD_D[25] | SD_D[27] | DVSS | DVSS | DVSS |
| H | NC | NC | NC | STMD | DVSS | DVSS | DVSS | DVSS |
| J | NC | NC | NC | SCAN_EN | DVDDIO | DVSS | DVSS | DVSS |
| K | NC | NC | NC | NC | DVDDIO | DVSS | DVSS | DVSS |
| L | NC | NC | NC | H_CPU_SPI_N | DVDDIO | DVSS | DVSS | DVSS |
| M | NC | NC | NC | DAT_32_16_N | DVDDIO | DVDDC | DVDDC | DVDDC |
| N | NC | NC | NC | NC | MBIST_DONE | MBIST_EN | DVDDIO | DVDDIO |
| P | NC | NC | NC | NC | TDM1_RX_SYNC | TDM1_TSIG_CTS | TDM1_RSIG_RTS | ACVSS1 |
| R | NC | NC | NC | MBIST_FAIL | TDM1_ACLK | TDM1_TX_SYNC | TDM1_TX_MF_CD | ACVDD1 |
| T | NC | NC | NC | TDM1_TX | TDM1_RX | TDM1_TCLK | TDM1_RCLK | ACVDD2 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |
|---|---|---|---|---|---|---|---|---|
| SD_D[28] | H_AD[21] | H_AD[24] | H_AD[12] | H_AD[8] | MII_TXD[1] | MII_RX_ERR | CLK_MII_RX | A |
| SD_D[24] | H_AD[19] | H_AD[22] | H_D[21] | H_AD[17] | MII_TXD[2] | MII_COL | MII_RXD[0] | B |
| SD_D[30] | H_AD[10] | H_WR_BE3_N/ SPI_CI | H_AD[14] | H_WR_BE0_N/ SPI_CLK | MII_TXD[3] | MII_CRS | MII_RXD[1] | C |
| SD_D[29] | H_AD[4] | H_AD[6] | H_AD[15] | H_AD[2] | MII_TX_EN | CLK_MII_TX | MII_RXD[2] | D |
| SD_D[26] | H_WR_BE2_N/ SPI_SEL_N | H_CS_N | H_AD[11] | H_AD[5] | MII_TX_ERR | CLK_SSMII_TX | MII_RXD[3] | E |
| DVDDC | DVDDC | DVDDC | H_D[23] | H_D[24] | H_D[27] | MII_TXD[0] | MII_RX_DV | F |
| DVSS | DVSS | DVSS | DVDDIO | H_D[25] | H_D[28] | MDIO | MDC | G |
| DVSS | DVSS | DVSS | DVDDIO | H_D[26] | H_D[29] | H_D[31] | H_D[30] | H |
| DVSS | DVSS | DVSS | DVDDIO | H_D[6] | H_D[14] | H_D[18] | H_D[22] | J |
| DVSS | DVSS | DVSS | DVDDIO | H_D[0]/SPI_MISO | H_D[12] | H_D[4] | H_D[19] | K |
| DVSS | DVSS | DVSS | DVDDIO | H_D[3] | H_D[16] | H_D[8] | H_D[17] | L |
| DVDDC | DVDDC | DVDDC | H_D[1] | H_D[10] | CLK_CMN | H_D[13] | H_D[15] | M |
| DVDDIO | DVDDIO | HiZ_N | H_R_W_N/SPI_CP | H_AD[7] | H_AD[23] | H_D[7] | H_D[20] | N |
| CLK_HIGH | JTDI | JTDO | H_WR_BE1_N/ SPI_MOSI | H_AD[3] | H_AD[20] | H_D[5] | H_D[11] | P |
| CLK_SYS_S | JTCLK | RST_SYS_N | H_READY_N | H_AD[1] | H_AD[18] | H_D[2] | H_D[9] | R |
| ACVSS2 | JTMS | JTRST_N | CLK_SYS/SCCLK | H_INT[0] | H_AD[9] | H_AD[13] | H_AD[16] | T |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

## 16.3 DS34S102 Pin Assignment

*Figure 16-2. DS34S102 Pin Assignment (TE-CSBGA Package)*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| A | SD_RAS_N | SD_BA[1] | SD_A[3] | SD_A[10] | SD_DQM[3] | SD_D[12] | SD_D[18] | SD_D[23] |
| B | SD_WE_N | SD_CLK | SD_A[5] | SD_A[8] | SD_D[1] | SD_D[10] | SD_D[5] | SD_D[14] |
| C | SD_CAS_N | SD_CS_N | SD_A[1] | SD_A[7] | SD_DQM[0] | SD_D[3] | SD_D[17] | SD_D[31] |
| D | SD_BA[0] | SD_A[0] | SD_A[9] | SD_DQM[1] | SD_D[6] | SD_D[7] | SD_D[15] | SD_D[19] |
| E | SD_A[2] | SD_A[4] | SD_A[6] | SD_D[0] | SD_D[9] | SD_DQM[2] | SD_D[8] | SD_D[21] |
| F | SD_D[2] | SD_A[11] | SD_D[11] | SD_D[13] | SD_D[16] | DVDDC | DVDDC | DVDDC |
| G | SD_D[4] | SD_D[20] | SD_D[22] | SD_D[25] | SD_D[27] | DVSS | DVSS | DVSS |
| H | NC | NC | NC | STMD | DVSS | DVSS | DVSS | DVSS |
| J | NC | NC | NC | SCAN_EN | DVDDIO | DVSS | DVSS | DVSS |
| K | NC | NC | NC | NC | DVDDIO | DVSS | DVSS | DVSS |
| L | NC | NC | NC | H_CPU_SPI_N | DVDDIO | DVSS | DVSS | DVSS |
| M | NC | NC | NC | DAT_32_16_N | DVDDIO | DVDDC | DVDDC | DVDDC |
| N | NC | NC | NC | NC | MBIST_DONE | MBIST_EN | DVDDIO | DVDDIO |
| P | TDM2_TX | TDM2_RX | TDM2_TCLK | TDM2_RCLK | TDM1_RX_SYNC | TDM1_TSIG_CTS | TDM1_RSIG_RTS | ACVSS1 |
| R | TDM2_ACLK | TDM2_TX_SYNC | TDM2_TX_MF_CD | MBIST_FAIL | TDM1_ACLK | TDM1_TX_SYNC | TDM1_TX_MF_CD | ACVDD1 |
| T | TDM2_RX_SYNC | TDM2_TSIG_CTS | TDM2_RSIG_RTS | TDM1_TX | TDM1_RX | TDM1_TCLK | TDM1_RCLK | ACVDD2 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |
|---|---|---|---|---|---|---|---|---|
| SD_D[28] | H_AD[21] | H_AD[24] | H_AD[12] | H_AD[8] | MII_TXD[1] | MII_RX_ERR | CLK_MII_RX | A |
| SD_D[24] | H_AD[19] | H_AD[22] | H_D[21] | H_AD[17] | MII_TXD[2] | MII_COL | MII_RXD[0] | B |
| SD_D[30] | H_AD[10] | H_WR_BE3_N/ SPI_CI | H_AD[14] | H_WR_BE0_N/ SPI_CLK | MII_TXD[3] | MII_CRS | MII_RXD[1] | C |
| SD_D[29] | H_AD[4] | H_AD[6] | H_AD[15] | H_AD[2] | MII_TX_EN | CLK_MII_TX | MII_RXD[2] | D |
| SD_D[26] | H_WR_BE2_N/ SPI_SEL_N | H_CS_N | H_AD[11] | H_AD[5] | MII_TX_ERR | CLK_SSMII_TX | MII_RXD[3] | E |
| DVDDC | DVDDC | DVDDC | H_D[23] | H_D[24] | H_D[27] | MII_TXD[0] | MII_RX_DV | F |
| DVSS | DVSS | DVSS | DVDDIO | H_D[25] | H_D[28] | MDIO | MDC | G |
| DVSS | DVSS | DVSS | DVDDIO | H_D[26] | H_D[29] | H_D[31] | H_D[30] | H |
| DVSS | DVSS | DVSS | DVDDIO | H_D[6] | H_D[14] | H_D[18] | H_D[22] | J |
| DVSS | DVSS | DVSS | DVDDIO | H_D[0]/SPI_MISO | H_D[12] | H_D[4] | H_D[19] | K |
| DVSS | DVSS | DVSS | DVDDIO | H_D[3] | H_D[16] | H_D[8] | H_D[17] | L |
| DVDDC | DVDDC | DVDDC | H_D[1] | H_D[10] | CLK_CMN | H_D[13] | H_D[15] | M |
| DVDDIO | DVDDIO | HiZ_N | H_R_W_N/SPI_CP | H_AD[7] | H_AD[23] | H_D[7] | H_D[20] | N |
| CLK_HIGH | JTDI | JTDO | H_WR_BE1_N/ SPI_MOSI | H_AD[3] | H_AD[20] | H_D[5] | H_D[11] | P |
| CLK_SYS_S | JTCLK | RST_SYS_N | H_READY_N | H_AD[1] | H_AD[18] | H_D[2] | H_D[9] | R |
| ACVSS2 | JTMS | JTRST_N | CLK_SYS/SCCLK | H_INT[0] | H_AD[9] | H_AD[13] | H_AD[16] | T |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

## 16.4 DS34S104 Pin Assignment

*Figure 16-3. DS34S104 Pin Assignment (TE-CSBGA Package)*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| A | SD_RAS_N | SD_BA[1] | SD_A[3] | SD_A[10] | SD_DQM[3] | SD_D[12] | SD_D[18] | SD_D[23] |
| B | SD_WE_N | SD_CLK | SD_A[5] | SD_A[8] | SD_D[1] | SD_D[10] | SD_D[5] | SD_D[14] |
| C | SD_CAS_N | SD_CS_N | SD_A[1] | SD_A[7] | SD_DQM[0] | SD_D[3] | SD_D[17] | SD_D[31] |
| D | SD_BA[0] | SD_A[0] | SD_A[9] | SD_DQM[1] | SD_D[6] | SD_D[7] | SD_D[15] | SD_D[19] |
| E | SD_A[2] | SD_A[4] | SD_A[6] | SD_D[0] | SD_D[9] | SD_DQM[2] | SD_D[8] | SD_D[21] |
| F | SD_D[2] | SD_A[11] | SD_D[11] | SD_D[13] | SD_D[16] | DVDDC | DVDDC | DVDDC |
| G | SD_D[4] | SD_D[20] | SD_D[22] | SD_D[25] | SD_D[27] | DVSS | DVSS | DVSS |
| H | TDM4_TX | TDM4_RX | TDM4_TCLK | STMD | DVSS | DVSS | DVSS | DVSS |
| J | TDM4_RCLK | TDM4_ACLK | TDM4_TX_SYNC | SCAN_EN | DVDDIO | DVSS | DVSS | DVSS |
| K | TDM4_TX_MF_CD | TDM4_RX_SYNC | TDM4_TSIG_CTS | TDM4_RSIG_RTS | DVDDIO | DVSS | DVSS | DVSS |
| L | TDM3_TX | TDM3_RX | TDM3_TCLK | H_CPU_SPI_N | DVDDIO | DVSS | DVSS | DVSS |
| M | TDM3_RCLK | TDM3_ACLK | TDM3_TX_SYNC | DAT_32_16_N | DVDDIO | DVDDC | DVDDC | DVDDC |
| N | TDM3_TX_MF_CD | TDM3_RX_SYNC | TDM3_TSIG_CTS | TDM3_RSIG_RTS | MBIST_DONE | MBIST_EN | DVDDIO | DVDDIO |
| P | TDM2_TX | TDM2_RX | TDM2_TCLK | TDM2_RCLK | TDM1_RX_SYNC | TDM1_TSIG_CTS | TDM1_RSIG_RTS | ACVSS1 |
| R | TDM2_ACLK | TDM2_TX_SYNC | TDM2_TX_MF_CD | MBIST_FAIL | TDM1_ACLK | TDM1_TX_SYNC | TDM1_TX_MF_CD | ACVDD1 |
| T | TDM2_RX_SYNC | TDM2_TSIG_CTS | TDM2_RSIG_RTS | TDM1_TX | TDM1_RX | TDM1_TCLK | TDM1_RCLK | ACVDD2 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |
|---|---|---|---|---|---|---|---|---|
| SD_D[28] | H_AD[21] | H_AD[24] | H_AD[12] | H_AD[8] | MII_TXD[1] | MII_RX_ERR | CLK_MII_RX | A |
| SD_D[24] | H_AD[19] | H_AD[22] | H_D[21] | H_AD[17] | MII_TXD[2] | MII_COL | MII_RXD[0] | B |
| SD_D[30] | H_AD[10] | H_WR_BE3_N/ SPI_CI | H_AD[14] | H_WR_BE0_N/ SPI_CLK | MII_TXD[3] | MII_CRS | MII_RXD[1] | C |
| SD_D[29] | H_AD[4] | H_AD[6] | H_AD[15] | H_AD[2] | MII_TX_EN | CLK_MII_TX | MII_RXD[2] | D |
| SD_D[26] | H_WR_BE2_N/ SPI_SEL_N | H_CS_N | H_AD[11] | H_AD[5] | MII_TX_ERR | CLK_SSMII_TX | MII_RXD[3] | E |
| DVDDC | DVDDC | DVDDC | H_D[23] | H_D[24] | H_D[27] | MII_TXD[0] | MII_RX_DV | F |
| DVSS | DVSS | DVSS | DVDDIO | H_D[25] | H_D[28] | MDIO | MDC | G |
| DVSS | DVSS | DVSS | DVDDIO | H_D[26] | H_D[29] | H_D[31] | H_D[30] | H |
| DVSS | DVSS | DVSS | DVDDIO | H_D[6] | H_D[14] | H_D[18] | H_D[22] | J |
| DVSS | DVSS | DVSS | DVDDIO | H_D[0]/SPI_MISO | H_D[12] | H_D[4] | H_D[19] | K |
| DVSS | DVSS | DVSS | DVDDIO | H_D[3] | H_D[16] | H_D[8] | H_D[17] | L |
| DVDDC | DVDDC | DVDDC | H_D[1] | H_D[10] | CLK_CMN | H_D[13] | H_D[15] | M |
| DVDDIO | DVDDIO | HiZ_N | H_R_W_N/SPI_CP | H_AD[7] | H_AD[23] | H_D[7] | H_D[20] | N |
| CLK_HIGH | JTDI | JTDO | H_WR_BE1_N/ SPI_MOSI | H_AD[3] | H_AD[20] | H_D[5] | H_D[11] | P |
| CLK_SYS_S | JTCLK | RST_SYS_N | H_READY_N | H_AD[1] | H_AD[18] | H_D[2] | H_D[9] | R |
| ACVSS2 | JTMS | JTRST_N | CLK_SYS/SCCLK | H_INT[0] | H_AD[9] | H_AD[13] | H_AD[16] | T |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

## 16.5 DS34S108 Pin Assignment

**Table 16-2. DS34S108 Pin Assignment (Sorted by Signal Name)**

| Signal Name | Ball | Signal Name | Ball | Signal Name | Ball | Signal Name | Ball |
|---|---|---|---|---|---|---|---|
| ACVDD1 | M2 | H_AD[18] | M19 | NC | C1 | SD_D[1] | F21 |
| ACVDD2 | K2 | H_AD[19] | N21 | NC | C10 | SD_D[10] | B22 |
| ACVSS1 | M1 | H_AD[2] | M21 | NC | C2 | SD_D[11] | H20 |
| ACVSS2 | K1 | H_AD[20] | M17 | NC | C5 | SD_D[12] | C21 |
| CLK_CMN | P1 | H_AD[21] | P20 | NC | C6 | SD_D[13] | H18 |
| CLK_HIGH | L1 | H_AD[22] | R22 | NC | C7 | SD_D[14] | C22 |
| CLK_MII_RX | V16 | H_AD[23] | N17 | NC | C8 | SD_D[15] | D21 |
| CLK_MII_TX | AA18 | H_AD[24] | T21 | NC | C9 | SD_D[16] | G20 |
| CLK_SSMII_TX | Y19 | H_AD[3] | K16 | NC | D3 | SD_D[17] | D22 |
| CLK_SYS/SCCLK | J1 | H_AD[4] | M22 | NC | D5 | SD_D[18] | J20 |
| CLK_SYS_S | J2 | H_AD[5] | T20 | NC | D6 | SD_D[19] | G21 |
| DAT_32_16_N | L21 | H_AD[6] | M18 | NC | D7 | SD_D[2] | G19 |
| DVDDC | A12 | H_AD[7] | M16 | NC | D8 | SD_D[20] | J21 |
| DVDDC | B11 | H_AD[8] | M20 | NC | D9 | SD_D[21] | E22 |
| DVDDC | C20 | H_AD[9] | L16 | NC | E1 | SD_D[22] | J19 |
| DVDDC | C4 | H_CPU_SPI_N | K19 | NC | E2 | SD_D[23] | H21 |
| DVDDC | E18 | H_CS_N | L17 | NC | E4 | SD_D[24] | F22 |
| DVDDC | E20 | H_D[0]/SPI_MISO | T22 | NC | E6 | SD_D[25] | K21 |
| DVDDC | E5 | H_D[1] | U21 | NC | E7 | SD_D[26] | G22 |
| DVDDC | G18 | H_D[10] | V22 | NC | E8 | SD_D[27] | K20 |
| DVDDC | G5 | H_D[11] | P18 | NC | F3 | SD_D[28] | H22 |
| DVDDC | L2 | H_D[12] | W22 | NC | F4 | SD_D[29] | G16 |
| DVDDC | T18 | H_D[13] | Y21 | NC | F5 | SD_D[3] | A21 |
| DVDDC | T5 | H_D[14] | P19 | NC | F7 | SD_D[30] | K22 |
| DVDDC | V18 | H_D[15] | Y22 | NC | F8 | SD_D[31] | J22 |
| DVDDC | V20 | H_D[16] | AA21 | NC | G1 | SD_D[4] | C16 |
| DVDDC | V5 | H_D[17] | AA22 | NC | G2 | SD_D[5] | A22 |
| DVDDC | Y10 | H_D[18] | AB21 | NC | G4 | SD_D[6] | A18 |
| DVDDC | Y20 | H_D[19] | U20 | NC | G6 | SD_D[7] | B21 |
| DVDDIO | AA11 | H_D[2] | N18 | NC | G7 | SD_D[8] | E21 |
| DVDDIO | AA13 | H_D[20] | R19 | NC | G8 | SD_D[9] | H19 |
| DVDDIO | AA15 | H_D[21] | AB22 | NC | H4 | SD_DQM[0] | A20 |
| DVDDIO | AA2 | H_D[22] | P17 | NC | H5 | SD_DQM[1] | E19 |
| DVDDIO | AA9 | H_D[23] | V21 | NC | H6 | SD_DQM[2] | B20 |
| DVDDIO | B10 | H_D[24] | R17 | NC | H7 | SD_DQM[3] | D20 |
| DVDDIO | B14 | H_D[25] | V19 | NC | J4 | SD_RAS_N | D16 |
| DVDDIO | B16 | H_D[26] | T19 | NC | J5 | SD_WE_N | C17 |
| DVDDIO | B2 | H_D[27] | W21 | NC | J6 | STMD | K15 |
| DVDDIO | B8 | H_D[28] | U16 | NC | J7 | TDM1_ACLK | E10 |
| DVDDIO | C3 | H_D[29] | R18 | NC | J8 | TDM1_RCLK | D12 |
| DVDDIO | D1 | H_D[3] | R20 | NC | K4 | TDM1_RSIG_RTS | C11 |
| DVDDIO | F2 | H_D[30] | W20 | NC | K5 | TDM1_RX | D10 |
| DVDDIO | H2 | H_D[31] | U19 | NC | K6 | TDM1_RX_SYNC | D11 |
| DVDDIO | J10 | H_D[4] | T17 | NC | K7 | TDM1_TCLK | F12 |
| DVDDIO | J11 | H_D[5] | P16 | NC | K8 | TDM1_TSIG_CTS | E11 |
| DVDDIO | J12 | H_D[6] | U18 | NC | L22 | TDM1_TX | C12 |
| DVDDIO | J13 | H_D[7] | R16 | NC | L4 | TDM1_TX_MF_CD | F13 |
| DVDDIO | K14 | H_D[8] | U22 | NC | L5 | TDM1_TX_SYNC | E13 |
| DVDDIO | K9 | H_D[9] | T16 | NC | L6 | TDM2_ACLK | E9 |
| DVDDIO | L14 | H_INT[0] | J17 | NC | L7 | TDM2_RCLK | E12 |

| Signal Name | Ball | Signal Name | Ball | Signal Name | Ball | Signal Name | Ball |
|---|---|---|---|---|---|---|---|
| DVDDIO | L9 | H_R_W_N/SPI_CP | K17 | NC | L8 | TDM2_RSIG_RTS | C14 |
| DVDDIO | M14 | H_READY_N | K18 | NC | M4 | TDM2_RX | D13 |
| DVDDIO | M9 | H_WR_BE0_N/SPI_CLK | L19 | NC | M5 | TDM2_RX_SYNC | C13 |
| DVDDIO | N14 | H_WR_BE1_N/SPI_MOSI | J16 | NC | M6 | TDM2_TCLK | G10 |
| DVDDIO | N9 | H_WR_BE2_N/SPI_SEL_N | J18 | NC | M7 | TDM2_TSIG_CTS | F11 |
| DVDDIO | P10 | H_WR_BE3_N/SPI_CI | L20 | NC | M8 | TDM2_TX | G11 |
| DVDDIO | P11 | HiZ_N | T3 | NC | N4 | TDM2_TX_MF_CD | F10 |
| DVDDIO | P12 | JTCLK | L3 | NC | N5 | TDM2_TX_SYNC | E14 |
| DVDDIO | P13 | JTDI | M3 | NC | N6 | TDM3_ACLK | G14 |
| DVDDIO | R2 | JTDO | N3 | NC | N7 | TDM3_RCLK | C15 |
| DVDDIO | U2 | JTMS | K3 | NC | N8 | TDM3_RSIG_RTS | G13 |
| DVDDIO | V3 | JTRST_N | P3 | NC | P4 | TDM3_RX | D15 |
| DVDDIO | W1 | MBIST_DONE | M15 | NC | P5 | TDM3_RX_SYNC | D14 |
| DVSS | A10 | MBIST_EN | P15 | NC | P6 | TDM3_TCLK | G9 |
| DVSS | A14 | MBIST_FAIL | N15 | NC | P7 | TDM3_TSIG_CTS | G12 |
| DVSS | A16 | MCLK | N1 | NC | P8 | TDM3_TX | E15 |
| DVSS | A8 | MDC | AB17 | NC | P9 | TDM3_TX_MF_CD | F9 |
| DVSS | AA1 | MDIO | AA20 | NC | R3 | TDM3_TX_SYNC | F14 |
| DVSS | AB11 | MII_COL | AA17 | NC | R4 | TDM4_ACLK | H12 |
| DVSS | AB13 | MII_CRS | Y18 | NC | R5 | TDM4_RCLK | J14 |
| DVSS | AB15 | MII_RX_DV | Y17 | NC | R6 | TDM4_RSIG_RTS | F15 |
| DVSS | AB9 | MII_RX_ERR | V17 | NC | R7 | TDM4_RX | H9 |
| DVSS | B1 | MII_RXD[0] | AA16 | NC | T1 | TDM4_RX_SYNC | H14 |
| DVSS | B12 | MII_RXD[1] | W16 | NC | T2 | TDM4_TCLK | H11 |
| DVSS | D19 | MII_RXD[2] | AB16 | NC | T4 | TDM4_TSIG_CTS | G15 |
| DVSS | D2 | MII_RXD[3] | Y16 | NC | T6 | TDM4_TX | J9 |
| DVSS | D4 | MII_TX_EN | W17 | NC | T7 | TDM4_TX_MF_CD | H13 |
| DVSS | E3 | MII_TX_ERR | AB20 | NC | T8 | TDM4_TX_SYNC | H10 |
| DVSS | F1 | MII_TXD[0] | AB18 | NC | U3 | TDM5_ACLK | V11 |
| DVSS | F17 | MII_TXD[1] | W18 | NC | U4 | TDM5_RCLK | V9 |
| DVSS | F6 | MII_TXD[2] | AA19 | NC | U5 | TDM5_RSIG_RTS | T9 |
| DVSS | H1 | MII_TXD[3] | AB19 | NC | U7 | TDM5_RX | R11 |
| DVSS | H15 | NC | A1 | NC | U8 | TDM5_RX_SYNC | U14 |
| DVSS | H8 | NC | A11 | NC | V1 | TDM5_TCLK | T13 |
| DVSS | K10 | NC | A13 | NC | V2 | TDM5_TSIG_CTS | P14 |
| DVSS | K11 | NC | A15 | NC | V4 | TDM5_TX | R12 |
| DVSS | K12 | NC | A2 | NC | V6 | TDM5_TX_MF_CD | R10 |
| DVSS | K13 | NC | A3 | NC | V7 | TDM5_TX_SYNC | R14 |
| DVSS | L10 | NC | A4 | NC | V8 | TDM6_ACLK | W14 |
| DVSS | L11 | NC | A5 | NC | W3 | TDM6_RCLK | T12 |
| DVSS | L12 | NC | A6 | NC | W5 | TDM6_RSIG_RTS | R9 |
| DVSS | L13 | NC | A7 | NC | W6 | TDM6_RX | V12 |
| DVSS | M10 | NC | A9 | NC | W7 | TDM6_RX_SYNC | T15 |
| DVSS | M11 | NC | AA10 | NC | W8 | TDM6_TCLK | V15 |
| DVSS | M12 | NC | AA12 | NC | Y1 | TDM6_TSIG_CTS | V13 |
| DVSS | M13 | NC | AA14 | NC | Y2 | TDM6_TX | W15 |
| DVSS | N10 | NC | AA3 | NC | Y4 | TDM6_TX_MF_CD | U15 |
| DVSS | N11 | NC | AA4 | NC | Y5 | TDM6_TX_SYNC | T10 |
| DVSS | N12 | NC | AA5 | NC | Y6 | TDM7_ACLK | V14 |
| DVSS | N13 | NC | AA6 | NC | Y7 | TDM7_RCLK | U13 |
| DVSS | N2 | NC | AA7 | NC | Y8 | TDM7_RSIG_RTS | T14 |
| DVSS | R1 | NC | AA8 | RST_SYS_N | P2 | TDM7_RX | U12 |
| DVSS | R15 | NC | AB1 | SCEN | J15 | TDM7_RX_SYNC | R13 |
| DVSS | R8 | NC | AB10 | SD_A[0] | A17 | TDM7_TCLK | Y11 |
| DVSS | U1 | NC | AB12 | SD_A[1] | F18 | TDM7_TSIG_CTS | W9 |

| Signal Name | Ball | Signal Name | Ball | Signal Name | Ball | Signal Name | Ball |
|---|---|---|---|---|---|---|---|
| DVSS | U17 | NC | AB14 | SD_A[10] | B19 | TDM7_TX | W12 |
| DVSS | U6 | NC | AB2 | SD_A[11] | D17 | TDM7_TX_MF_CD | Y15 |
| DVSS | W19 | NC | AB3 | SD_A[2] | F16 | TDM7_TX_SYNC | U11 |
| DVSS | W2 | NC | AB4 | SD_A[3] | B18 | TDM8_ACLK | Y13 |
| DVSS | W4 | NC | AB5 | SD_A[4] | E17 | TDM8_RCLK | U9 |
| DVSS | Y12 | NC | AB6 | SD_A[5] | A19 | TDM8_RSIG_RTS | Y9 |
| DVSS | Y3 | NC | AB7 | SD_A[6] | H17 | TDM8_RX | V10 |
| H_AD[1] | L18 | NC | AB8 | SD_A[7] | F19 | TDM8_RX_SYNC | T11 |
| H_AD[10] | N22 | NC | B13 | SD_A[8] | F20 | TDM8_TCLK | Y14 |
| H_AD[11] | L15 | NC | B15 | SD_A[9] | D18 | TDM8_TSIG_CTS | W11 |
| H_AD[12] | P21 | NC | B3 | SD_BA[0] | G17 | TDM8_TX | W10 |
| H_AD[13] | N16 | NC | B4 | SD_BA[1] | C19 | TDM8_TX_MF_CD | W13 |
| H_AD[14] | N20 | NC | B5 | SD_CAS_N | E16 | TDM8_TX_SYNC | U10 |
| H_AD[15] | P22 | NC | B6 | SD_CLK | H16 | TEST_CLK | J3 |
| H_AD[16] | N19 | NC | B7 | SD_CS_N | B17 | TST_CLD | G3 |
| H_AD[17] | R21 | NC | B9 | SD_D[0] | C18 | NC | H3 |

*Figure 16-4. DS34S108 Pin Assignment (HSBGA Package)*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | NC | NC | NC | NC | NC | NC | NC | DVSS | NC | DVSS | NC |
| B | DVSS | DVDDIO | NC | NC | NC | NC | NC | DVDDIO | NC | DVDDIO | DVDDC |
| C | NC | NC | DVDDIO | DVDDC | NC | NC | NC | NC | NC | NC | TDM1_RSIG_RTS |
| D | DVDDIO | DVSS | NC | DVSS | NC | NC | NC | NC | NC | TDM1_RX | TDM1_RX_SYNC |
| E | NC | NC | DVSS | NC | DVDDC | NC | NC | NC | TDM2_ACLK | TDM1_ACLK | TDM1_TSIG_CTS |
| F | DVSS | DVDDIO | NC | NC | NC | DVSS | NC | NC | TDM3_TX_MF_CD | TDM2_TX_MF_CD | TDM2_TSIG_CTS |
| G | NC | NC | TST_CLD | NC | DVDDC | NC | NC | NC | TDM3_TCLK | TDM2_TCLK | TDM2_TX |
| H | DVSS | DVDDIO | NC | NC | NC | NC | NC | DVSS | TDM4_RX | TDM4_TX_SYNC | TDM4_TCLK |
| J | CLK_SYS/SCCLK | CLK_SYS_S | TEST_CLK | NC | NC | NC | NC | NC | TDM4_TX | DVDDIO | DVDDIO |
| K | ACVSS2 | ACVDD2 | JTMS | NC | NC | NC | NC | NC | DVDDIO | DVSS | DVSS |
| L | CLK_HIGH | DVDDC | JTCLK | NC | NC | NC | NC | NC | DVDDIO | DVSS | DVSS |
| M | ACVSS1 | ACVDD1 | JTDI | NC | NC | NC | NC | NC | DVDDIO | DVSS | DVSS |
| N | MCLK | DVSS | JTDO | NC | NC | NC | NC | NC | DVDDIO | DVSS | DVSS |
| P | CLK_CMN | RST_SYS_N | JTRST_N | NC | NC | NC | NC | NC | NC | DVDDIO | DVDDIO |
| R | DVSS | DVDDIO | NC | NC | NC | NC | NC | DVSS | TDM6_RSIG_RTS | TDM5_TX_MF_CD | TDM5_RX |
| T | NC | NC | HiZ_N | NC | DVDDC | NC | NC | NC | TDM5_RSIG_RTS | TDM6_TX_SYNC | TDM8_RX_SYNC |
| U | DVSS | DVDDIO | NC | NC | NC | DVSS | NC | NC | TDM8_RCLK | TDM8_TX_SYNC | TDM7_TX_SYNC |
| V | NC | NC | DVDDIO | NC | DVDDC | NC | NC | NC | TDM5_RCLK | TDM8_RX | TDM5_ACLK |
| W | DVDDIO | DVSS | NC | DVSS | NC | NC | NC | NC | TDM7_TSIG_CTS | TDM8_TX | TDM8_TSIG_CTS |
| Y | NC | NC | DVSS | NC | NC | NC | NC | NC | TDM8_RSIG_RTS | DVDDC | TDM7_TCLK |
| AA | DVSS | DVDDIO | NC | NC | NC | NC | NC | NC | DVDDIO | NC | DVDDIO |
| AB | NC | NC | NC | NC | NC | NC | NC | NC | DVSS | NC | DVSS |

| | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | DVDDC | NC | DVSS | NC | DVSS | SD_A[0] | SD_D[6] | SD_A[5] | SD_DQM[0] | SD_D[3] | SD_D[5] |
| B | DVSS | NC | DVDDIO | NC | DVDDIO | SD_CS_N | SD_A[3] | SD_A[10] | SD_DQM[2] | SD_D[7] | SD_D[10] |
| C | TDM1_TX | TDM2_RX_SYNC | TDM2_RSIG_RTS | TDM3_RCLK | SD_D[4] | SD_WE_N | SD_D[0] | SD_BA[1] | DVDDC | SD_D[12] | SD_D[14] |
| D | TDM1_RCLK | TDM2_RX | TDM3_RX_SYNC | TDM3_RX | SD_RAS_N | SD_A[11] | SD_A[9] | DVSS | SD_DQM[3] | SD_D[15] | SD_D[17] |
| E | TDM2_RCLK | TDM1_TX_SYNC | TDM2_TX_SYNC | TDM3_TX | SD_CAS_N | SD_A[4] | DVDDC | SD_DQM[1] | DVDDC | SD_D[8] | SD_D[21] |
| F | TDM1_TCLK | TDM1_TX_MF_CD | TDM3_TX_SYNC | TDM4_RSIG_RTS | SD_A[2] | DVSS | SD_A[1] | SD_A[7] | SD_A[8] | SD_D[1] | SD_D[24] |
| G | TDM3_TSIG_CTS | TDM3_RSIG_RTS | TDM3_ACLK | TDM4_TSIG_CTS | SD_D[29] | SD_BA[0] | DVDDC | SD_D[2] | SD_D[16] | SD_D[19] | SD_D[26] |
| H | TDM4_ACLK | TDM4_TX_MF_CD | TDM4_RX_SYNC | DVSS | SD_CLK | SD_A[6] | SD_D[13] | SD_D[9] | SD_D[11] | SD_D[23] | SD_D[28] |
| J | DVDDIO | DVDDIO | TDM4_RCLK | SCEN | H_WR_BE1_N/SPI_MOS | H_INT[0] | H_WR_BE2_N/SPI_SEL_ | SD_D[22] | SD_D[18] | SD_D[20] | SD_D[31] |
| K | DVSS | DVSS | DVDDIO | STMD | H_AD[3] | H_R_W_N/SPI_CP | H_READY_N | H_CPU_SPI_N | SD_D[27] | SD_D[25] | SD_D[30] |
| L | DVSS | DVSS | DVDDIO | H_AD[11] | H_AD[9] | H_CS_N | H_AD[1] | H_WR_BE0_N/SPI_CLK | H_WR_BE3_N/SPI_CI | DAT_32_16_N | NC |
| M | DVSS | DVSS | DVDDIO | MBIST_DONE | H_AD[7] | H_AD[20] | H_AD[6] | H_AD[18] | H_AD[8] | H_AD[2] | H_AD[4] |
| N | DVSS | DVSS | DVDDIO | MBIST_FAIL | H_AD[13] | H_AD[23] | H_D[2] | H_AD[16] | H_AD[14] | H_AD[19] | H_AD[10] |
| P | DVDDIO | DVDDIO | TDM5_TSIG_CTS | MBIST_EN | H_D[5] | H_D[22] | H_D[11] | H_D[14] | H_AD[21] | H_AD[12] | H_AD[15] |
| R | TDM5_TX | TDM7_RX_SYNC | TDM5_TX_SYNC | DVSS | H_D[7] | H_D[24] | H_D[29] | H_D[20] | H_D[3] | H_AD[17] | H_AD[22] |
| T | TDM6_RCLK | TDM5_TCLK | TDM7_RSIG_RTS | TDM6_RX_SYNC | H_D[9] | H_D[4] | DVDDC | H_D[26] | H_AD[5] | H_AD[24] | H_D[0]/SPI_MISO |
| U | TDM7_RX | TDM7_RCLK | TDM5_RX_SYNC | TDM6_TX_MF_CD | H_D[28] | DVSS | H_D[6] | H_D[31] | H_D[19] | H_D[1] | H_D[8] |
| V | TDM6_RX | TDM6_TSIG_CTS | TDM7_ACLK | TDM6_TCLK | CLK_MII_RX | MII_RX_ERR | DVDDC | H_D[25] | DVDDC | H_D[23] | H_D[10] |
| W | TDM7_TX | TDM8_TX_MF_CD | TDM6_ACLK | TDM6_TX | MII_RXD[1] | MII_TX_EN | MII_TXD[1] | DVSS | H_D[30] | H_D[27] | H_D[12] |
| Y | DVSS | TDM8_ACLK | TDM8_TCLK | TDM7_TX_MF_CD | MII_RXD[3] | MII_RX_DV | MII_CRS | CLK_SSMII_TX | DVDDC | H_D[13] | H_D[15] |
| AA | NC | DVDDIO | NC | DVDDIO | MII_RXD[0] | MII_COL | CLK_MII_TX | MII_TXD[2] | MDIO | H_D[16] | H_D[17] |
| AB | NC | DVSS | NC | DVSS | MII_RXD[2] | MDC | MII_TXD[0] | MII_TXD[3] | MII_TX_ERR | H_D[18] | H_D[21] |

## 17. Package Information

For the latest package outline information and land patterns, go to **www.maxim-ic.com/packages**.

DS34S101, DS34S102 and DS34S108 have a 256-lead thermally enhanced chip-scale ball grid array (TECSBGA) package. The TECSBGA package dimensions are shown in Maxim document 21-0353.

DS34S108 has a 484-lead thermally enhanced ball grid array (TEBGA) package. The TEBGA package dimensions are shown in Maxim document 21-0365.

## 18. Thermal Information

| Parameter | TECSBGA-256<br>DS34S101<br>DS34S102<br>DS34S104 | TEBGA-484<br>DS34S108 |
|---|---|---|
| Target Ambient Temperature Range | -40 to 85°C | -40 to 85°C |
| Die Junction Temperature Range | -40 to 125°C | -40 to 125°C |
| Theta Jc (junction to top of case) | 3.7 °C/W | 4.5 °C/W |
| Theta Jb (junction to bottom pins) | 13.1 °C/W | 7.1 °C/W |
| Theta Ja, Still Air (Note 1) | 26.2 °C/W | 15.0 °C/W |

**Note 1**: These numbers are estimates using JEDEC standard PCB and enclosure dimensions.